ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2018
T. Beeler and N. Thuerey
(Guest Editors)

Volume 37 (2018), Number 8

FTP-SC: Fuzzy Topology Preserving Stroke Correspondence

Wenwu Yang[†1], Hock-Soon Seah[2], Quan Chen[2], Hong-Ze Liew[3], Daniel Sýkora[4]

[1]Zhejiang Gongshang University, School of Computer and Information Engineering, China
[2]Nanyang Technological University, School of Computer Science and Engineering, Singapore
[3]CACANi Private Limited, Singapore
[4]Czech Technical University in Prague, Faculty of Electrical Engineering, Czech Republic

Abstract

Stroke correspondence construction is a precondition for vectorized 2D animation inbetweening and remains a challenging problem. This paper introduces the FTP-SC, a fuzzy topology preserving stroke correspondence technique, which is accurate and provides the user more effective control on the correspondence result than previous matching approaches. The method employs a two-stage scheme to progressively establish the stroke correspondence construction between the keyframes. In the first stage, the stroke correspondences with high confidence are constructed by enforcing the preservation of the so-called "fuzzy topology" which encodes intrinsic connectivity among the neighboring strokes. Starting with the high-confidence correspondences, the second stage performs a greedy matching algorithm to generate a full correspondence between the strokes. Experimental results show that the FTP-SC outperforms the existing approaches and can establish the stroke correspondence with a reasonable amount of user interaction even for keyframes with large geometric and spatial variations between strokes.

CCS Concepts

•***Computing methodologies*** → *Animation; Parametric curve and surface models;*

1. Introduction

The generation of inbetween frames that interpolate a given set of key frames is a major component in the traditional 2D animation process [JT95, IBiA09]. In the production of a 2D feature animation, drawing the inbetween frames is both tedious and time-consuming, requiring many hours of intensive labor by skilled professionals. Therefore, automation of the inbetweening process would not only allow the artist to concentrate on the more creative work of drawing the key frames, but it would also effectively increase productivity for 2D animation [BW76, Dur91].

However, automatic inbetweening has proven to be a formidable task [Cat78, FBC*95]. This is in part because the geometric information is inconsistent between the key frames, e.g., due to changing occlusion, which makes it near impossible to automatically match two successive key frames to each other, especially in cases of complex motion and large topology variation [DRvdP15]. Therefore, most recent work focuses on computer aided inbetweening where the artists are kept into the loop and are allowed to guide or edit the results [Kor02, WNS*10, Yan18]. The success of such systems comes from automating as much as possible while effortlessly deferring to user control whenever needed. In these systems, the core of automatic process consists of identifying a correspondence between the strokes of consecutive key frames, known as the stroke correspondence construction problem, and interpolating the positions of the corresponding strokes along predetermined paths, known as the stroke interpolation problem.

Most of the existing computer aided inbetweening techniques are mainly concerned with the stroke interpolation problem, and several practical solutions to the problem have been proposed, such as the logarithmic spiral trajectory in [WNS*10] and the context-aware interpolation of [Yan18]. For the stroke correspondence construction problem, topology ambiguities, i.e., a feature may be drawn with different number of strokes between successive key frames or a part in some keyframe may disappear in the next keyframe due to occlusion, make it infeasible to construct a full stroke correspondence between the successive key frames by using the fully automatic local or global optimization strategies [NSC*11, BZBM*16].

In [LCY*11, YBS*12], a stroke reconstruction algorithm is proposed to resolve the topology ambiguities. However, such algorithm often fails in practice, since it is akin to a difficult artificial intelligence problem to infer the occluded parts from the key drawings or to associate the stroke(s) that correspond to a common feature between the successive key frames. Experiments show that a feasible solution to the stroke correspondence construction problem would be to keep the artist into the loop [WNS*10, Yan18, Lim18], e.g., drawing occluded lines or merging/spliting the strokes, ulti-

† E-mail: wwyang@zjgsu.edu.cn

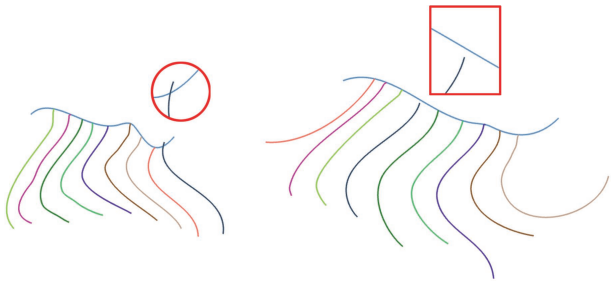

Figure 1: *A simple case is used to show the problems of the one-to-one stroke correspondence using the approaches proposed in [WNS*10] and [Yan18]. The correspondence result between the left and right key frames is automatically constructed using [Yan18], where the same color indicates a correspondence. On the other hand, the gap and imprecise intersection in the key frames make [WNS*10] unsuitable for this case.*

mately making possible the one-to-one correspondence between the strokes on two successive key frames. Furthermore, to make this solution practical and efficient, an effective matching technique for one-to-one stroke correspondence is necessary, such that the loop could be automated as much as possible.

Intuitively, establishing the one-to-one stroke correspondence between two keyframes with the same number of strokes seems technically simple. However, we find that this problem could be very challenging especially in the scenario of 2D character animations where complex and exaggerated motions are often involved, such that large geometric and spatial variations between strokes are common, e.g., two disjoined strokes may become intersected with each other in the next keyframe. As a result, the existing techniques either rely on time-consuming manual intervention [Ree81], or are unstable and error-prone, making them difficult to use in practice.

Figure 1 shows that even the state-of-the-art correspondence techniques may fail in this simple case. For example, the system of BetweenIT [WNS*10] builds the one-to-one stroke correspondence upon the compatibility in the intersection and connectivity between the strokes of the key frames. It needs to operate on very clean drawings with well-connected strokes, because gaps and imprecise intersections tend to cause incompatibility in the connectivity and intersection, as shown in Figure 1. Furthermore, BetweenIT is also sensitive to the spatial variation between the strokes, since such variation may destroy the compatibility in the strokes' intersection and connectivity; thus, it is more suitable for tight pairs of key frames, i.e., the involved motions are small or simple. Instead of the connectivity-compatibility constraints, the CACAI [Yan18] uses geometric shape of the strokes and their neighborhood information to establish the one-to-one stroke correspondence between the key frames. Obviously, this algorithm tends to generate mismatches when large geometric variations occur between the corresponding strokes, as shown in Figure 1.

On the other hand, when mismatches occur, a straightforward approach is to correct them one by one, as in [WNS*10]. However, this tends to be tedious and somewhat laborious, since the number of strokes in the key frames is typically large and the strokes may be very close to each other, which makes it hard to distinguish them. The CACAI [Yan18] improves the user interaction by incorporating it into the matching process, such that a user correction can have more control over the correspondence results. However, we find that in practice such incorporation may become unstable, e.g., a correction may introduce new mismatches, such that the user correction cannot be performed in a moving-forward way which is important in actual productions. A vivid example, corresponding to Figure 1, is shown in the accompanying video.

In this paper, we present the FTP-SC, a fuzzy topology preserving stroke correspondence technique. The key observation is that while the geometric shape and the spatial relationships between the strokes may often vary between successive key frames, "intrinsic" connectivity between the neighboring strokes is usually maintained in many parts of the drawing, manifesting their visual features. Throughout this paper, we use intrinsic connectivity to indicate the (*virtual*) connectivity between the strokes that may not meet at end points but can be considered to be connected at the feature level. For example, this is the case in Figure 1, where most of the corresponding strokes have a very different geometric shape, but the intrinsic connectivity between the horizontal stroke and the others remains unchanged. Intuitively, the intrinsic connectivity is free from the impact of the gaps and imprecise intersections between the strokes, which is different from the geometric connectivity between the strokes.

The main implication of this observation is that it is possible to design a robust and accurate algorithm for the stroke correspondence construction between successive key frames. At the core of our algorithm is a two-stage scheme where the high-confidence stroke correspondences are derived in the first stage and are then used in the second stage to guide the generation of a full correspondence between the strokes. In order to exploit the characteristic that the intrinsic connectivity tends to be maintained between the successive key frames, we introduce the so-called "fuzzy topology" which encodes the intrinsic connectivity among the neighboring strokes. By enforcing the strict preservation of the fuzzy topology, we are able to obtain the stroke correspondences with high confidence. Furthermore, these high-confidence stroke correspondences can be effectively employed to derive the correspondences between the remaining strokes by using the neighborhood information.

The main contributions of this paper are:

- a simple but practical correspondence technique which uses fuzzy topology to restrict the stroke association and thus is accurate even when large geometric and spatial variations occur;
- an effective user correction scheme which exhibits user-friendly 'moving forward' property;
- a data-dependent stroke interpolation strategy which generates natural inbetweening sequences for cases where locally large and inconsistent motions are involved.

2. System Workflow

The FTP-SC focuses on vectorized 2D animation inbetweening, where our prototype system is similar to the computer aided inbetweening systems such as [Kor02], BetweenIT [WNS*10], [Lim18], and CACAI [Yan18]. In the system, the artist draws the

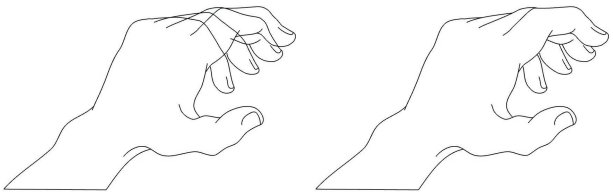

Figure 2: *The occlusion between the strokes in (a) is automatically handled by the system, as shown in (b).*

clean drawings using strokes where a stroke, possibly with variable width [SWT*05], is generated by vectorizing the pen's trajectory as the artist draws and is represented as a piecewise linear curve with uniformly sampled points. The system does not set up constraints, e.g., in the same drawing order, so that the artist can focus efforts on the creative aspects when s/he draws the key frames.

Given a pair of successive key frames, called the initial and the target key frame, respectively, the FTP-SC aims to construct a one-to-one correspondence between the strokes of the key frames, which serves as the basis for inbetweening and coloring in 2D computer-assisted animation production [QSST*05,SDC09]. In the following, we start by describing the two-stage FTP-SC technique (Section 3) and then evaluate its effectiveness and practicality with real production data (Section 5).

Like [WNS*10,Yan18,Lim18], the artist is kept into the loop for resolving topology ambiguities in the correspondence in order to establish the one-to-one stroke correspondence between the initial and target key frames. As shown in Figure 2, the artist is allowed to draw the occluded lines quickly in a layering-like scheme, and the system will automatically toggle their visibility by using the occlusion handling tools described in [WNS*10] and [Yan18]. In addition, simple operations of stroke merging and splitting are provided to adjust the number of strokes; in practice, usually only a small number (≤ 6) of adjustments are necessary.

3. FTP-SC: Two-Stage Correspondence Construction

Let $\{S_i\}$ be the set of strokes in the initial key frame and $\{T_j\}$ be the set of strokes in the target key frame. A general approach to the stroke correspondence construction between $\{S_i\}$ and $\{T_j\}$ is to define a metric for measuring the matching degree between strokes and, then, to find the corresponding strokes which have maximum matching degree between each other. Formally speaking, it is equivalent to the following optimization problem:

$$\underset{\{(i,j)\}}{\arg\max} \sum_{(i,j)} MD(S_i, T_j), \tag{1}$$

where each (i, j) corresponds to a pair of corresponding strokes S_i and T_j, and $MD(\bullet)$ returns the matching degree between a pair of strokes. However, in real 2D animation production, the geometric shape and the spatial relationships between the strokes may often vary between the successive key frames, especially in cases of complex or exaggerated motions. Therefore, no metric is perfect for all cases, i.e., a metric may work well in some situations but fails in

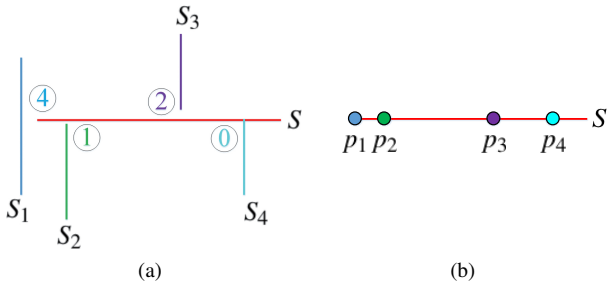

Figure 3: *The concept of $\alpha-$connectivity and fuzzy topology. For clarity and simplicity, we represent a stroke by a line segment. (a) With a threshold $\alpha = 5$, the stroke S has the intrinsic connectivity with the strokes S_1, S_2, S_3 and S_4, respectively, since $\mu_S(S_2) = 1$, $\mu_S(S_3) = 2$, $\mu_S(S_4) = 0$, and $\mu_{S_1}(S) = 4$, all of them being less than α; (b) The fuzzy topology of S is the ordered set $\{p_1, p_2, p_3, p_4\}$ where p_i is associated with the stroke S_i, $i = 1, 2, 3, 4$.*

others. Ultimately, we have to add various considerations into the metric with respect to different cases, which will make the metric very complex and quite difficult to maintain.

Our solution is to find certain common characteristics between successive key frames which may not be general enough for all strokes, but can be used to derive some correspondences with high confidence. Based on this strategy, the key idea of the two-stage FTP-SC technique is to generate the correspondences with high confidence at first. Then, remaining correspondences are obtained under the guidance of the high-confidence ones and, thus, are more probably correct. An overview is shown in Figure 4 and the accompanying video.

3.1. Preliminary definitions

In order to exploit the common characteristics between successive key frames that we have observed, we begin with the definitions of intrinsic connectivity and fuzzy topology, as illustrated in Figure 3.

Definition 1 Let S be a stroke in the initial key frame. The intrinsic connectivity between S and the other strokes in $\{S_i\}$ is characterized by a connectivity function which associates with each S_i its "grade of having the intrinsic connectivity with S," $\mu_S(S_i)$, in $\{S_i\}$.

Intuitively, the function $\mu_S(S_i)$ is similar to the membership (characteristic) function in the concept of fuzzy sets [Zad65], and describes how closely a stroke S_i approaches to the stroke S. Since the connectivity between the strokes is mainly manifested by their endpoint connections, we compute $\mu_S(S_i)$ as the minimum of distances from the end points of S_i to the stroke S.

Then, given a threshold α, for any pair of strokes S_i and S_j in the initial key frame, we say "there is the intrinsic connectivity with respect to α between S_i and S_j when $\mu_{S_i}(S_j) \leq \alpha$ or $\mu_{S_j}(S_i) \leq \alpha$, thus also called $\alpha-$connectivity".

Definition 2 Let S be a stroke in the initial key frame. The fuzzy topology of S is an ordered set of points $F_S = \{p_1, p_2, \ldots, p_k, \ldots\}$ which has the following properties:

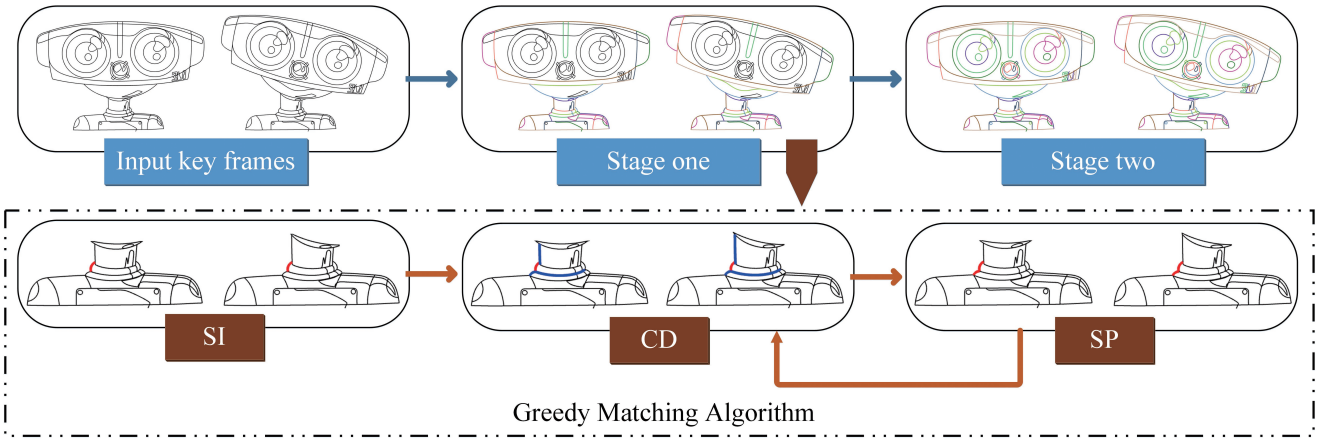

Figure 4: *Top row: an overview of the two-stage correspondence construction where the stroke correspondences established at each stage are indicated using the same non-black color; Bottom row: the greedy algorithm working for the stage one. At this stage, the SI-component automatically detects an initial seed, i.e., the red strokes; Given a seed, the CD-component finds the unmatched strokes that have intrinsic connectivity with the seed strokes, i.e., the blue strokes, and derives a list of candidate corresponding pairs between them if the seed strokes have a compatible α−topology; the SP-component selects one from all of the candidate corresponding pairs and establishes a correspondence between the strokes in the pair, obtaining a new seed. Please see the accompanying video for a live example.*

(a) Each p_k is associated with a stroke in $\{S_i\}$ that has the α−connectivity with S, and no two (p_k)s correspond to the same stroke,
(b) Let p_k be associated with the stroke S_k. The point p_k is on S, and is coincident with an endpoint of S when $\mu_{S_k}(S) \leq \alpha$; or, is the projection point of an endpoint of S_k to the stroke S when $\mu_S(S_k) \leq \alpha$,
(c) The order of p_k in the set is determined by its position on S when traversing the stroke from its beginning to the end.

Note that (a) means when multiple (p_k)s correspond to a same stroke, only the one with smallest grade is chosen. Since the fuzzy topology of a stroke is dependent on its α−connectivity with other strokes, we also call it "α−topology". Both of these concepts can be straightforwardly applied to the strokes of the target key frame.

3.2. A greedy algorithm for two-stage correspondence

In the two-stage FTP-SC technique, a greedy matching algorithm is employed in each stage to find the correspondences between the strokes of the key frames. The algorithm consists of three basic components: seed initializing (SI), candidate deriving (CD), and seed picking (SP), and can be described as follows:

An overview of the greedy matching algorithm is shown in the bottom row of Figure 4. In the two-stage FTP-SC technique, the SP-component remains fixed, while the concrete implementations of the SI-component and the CD-component differ between stage one and stage two.

Matching degree. In the greedy matching algorithm, we need to measure the matching degree between two strokes. Many existing metrics can be used for this purpose such as the differences in area and arc-length [WNS*10], visually-based composition [Yan18], shape context [SKK03], or perceptually-based metric [LWZ*04]. In our implementation, we use the similarity transformation to align two strokes and, then, measure their matching degree as the shape difference between them. Such a measure is usually robust and accurate and the computation is fast since it has an analytical form [CG97, YF09].

Algorithm 1: Greedy Matching Algorithm

```
1  h: a heap of storing the candidate pairs of corresponding
   strokes, which is keyed on matching degree between the strokes
2  Perform SI-component to obtain one or multiple pairs of corresponding
   strokes, {(S_i, T_j)}, as the initial seed(s);
3  For each initial seed, i.e., a pair of corresponding strokes, CD-
   component is performed to derive from it a list of candidate
   stroke pairs which are pushed into the heap h;
4  while h is not empty do
5      // Perform SP-component:
6      Pop from the heap h the candidate stroke pair with maximum matching degree;
7      if the strokes in the candidate pair are yet unmatched then
8          A correspondence is constructed between the strokes in
           the pair, leading to a new seed;
9          Perform CD-component to derive from the new seed a
           list of candidate stroke pairs which are pushed into the
           heap h.
10     end
11 end
```

3.3. Stage one: fuzzy topology preservation

In this stage, we restrict the correspondence between the strokes by preserving their fuzzy topology, such that the correspondences with high confidence are obtained (see Figure 4). This can be done by performing the greedy matching algorithm proposed in Section 3.2. In its SI-component, each of strokes in the initial key frame is matched to each of strokes in the target key frame, where the pair with maximum matching degree is chosen as the initial seed. Then, in the CD-component, we realize the constraint of the fuzzy topology preservation as follows.

Given a seed, i.e., a pair of corresponding strokes (S_i, T_j), the CD-component is performed to derive from it a list of candidate stroke pairs. Note that the directions of strokes S_i and T_j are made consistent by aligning their vertices (see Section 4). Let $F_{S_i} = \{p_1, p_2, \ldots, p_k, \ldots\}$ be the $\alpha-$topology of the stroke S_i and $F_{T_j} = \{q_1, q_2, \ldots, q_k, \ldots\}$ be the $\alpha-$topology of the stroke T_j. We say strokes S_i and T_j have a compatible $\alpha-$topology if and only if F_{S_i} and F_{T_j} are in one-to-one correspondence, i.e., F_{S_i} and F_{T_j} have the same number of elements. In practice, the value of α is dependent on the resolution and the size of the drawings. Our solution is to pre-define an initial value of α and then decrease $\alpha = \alpha - 1$ until $\alpha = 0$ or when F_{S_i} and F_{T_j} are compatible in fuzzy topology with respect to current α. In our experiments, we use $\alpha = 5$ as the initial value for all examples and find that it works well for imprecise placement of strokes such as gaps and imperfect intersections.

In order to strictly preserve the local fuzzy topology of the strokes S_i and T_j, the CD-component derives the candidate stroke pairs for S_i and T_j only when F_{S_i} and F_{T_j} are compatible in $\alpha-$topology. To do so, we successively pick out each element p_k from F_{S_i} and its counterpart q_k from F_{T_j}, and then create a candidate stroke pair by matching the stroke associated with p_k to the stroke associated with q_k. An illustration is given in Figure 5.

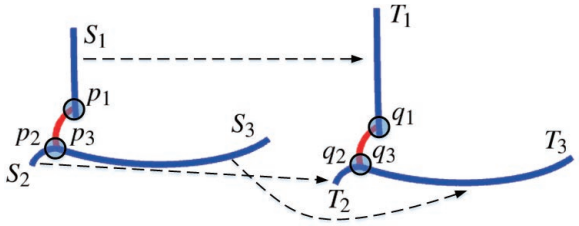

Figure 5: *Fuzzy topology preservation is realized by the CD-component in stage one. The red strokes are a pair of corresponding strokes and $\{p_1, p_2, p_3\}$ and $\{q_1, q_2, q_3\}$ are their $\alpha-$topology which are compatible. The CD creates three candidate pairs by matching each S_k, which is associated to p_k, to the stroke T_k, which is associated to q_k, where $k = 1, 2, 3$. Note that in this special case q_2 and q_3 are coincident so we need to re-determine their order in the $\alpha-$topology. We achieve this by using the matching degree between the strokes, e.g., S_2 is matched to T_2 but not to T_3, because the matching degree between S_2 and T_2 is higher.*

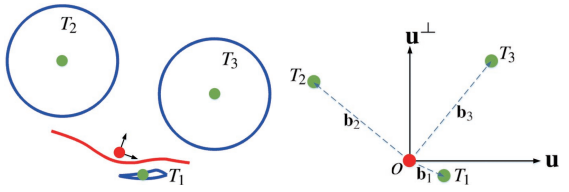

Figure 6: *Extraction of neighborhood information. (a) A target stroke (the red one) and its neighborhood (the blue ones: T_1, T_2, T_3) where the stroke barycenters are shown in circular dots and a local coordinate system is derived from the red stroke; (b) The relative position information of T_1 (or T_2, T_3) with respect to the red stroke is represented by the vector $\boldsymbol{b}_1$ (or $\boldsymbol{b}_2$, $\boldsymbol{b}_3$).*

3.4. Stage two: neighborhood competition

In this stage, the greedy matching algorithm is performed once again to generate a full correspondence between the strokes, as shown in Figure 4. In its SI-component, the initial seeds are obtained from the high-confidence correspondences that are generated in stage one. Given a seed, i.e., a pair of corresponding strokes (S_i, T_j), the CD-component will use their neighborhood information to derive a list of candidate stroke pairs in the following way.

For each stroke, we obtain its neighborhood information from k strokes that are nearest to it, where the distance between two strokes is defined as the minimum vertex distance between them. Let $N(S_i)$ and $N(T_j)$ be the strokes in the neighborhood of S_i and T_j, respectively. With respect to S_i, we extract the relative position information for each stroke in $N(S_i)$ as follows. First, the principal component analysis [Shl14] is applied to the vertex positions of S_i, obtaining a major eigenvector $\mathbf{u}$ whose direction manifests the main orientation of the stroke S_i. Then, a local coordinate system is formed as: $L = (o, \mathbf{u}, \mathbf{u}^\perp)$, where o is the barycenter of S_i and $\perp$ is a 2D operation such that $(x, y)^\perp = (-y, x)$. Finally, for each stroke in $N(S_i)$, its barycenter is transformed into the local system L and, then, its relative position information with respect to S_i is determined by the vector which is from o to the transformed barycenter in L. Similarly, the relative position information with respect to T_j can be extracted for each stroke in $N(T_j)$, as shown in Figure 6.

Given a stroke in $N(S_i)$ and a stroke in $N(T_j)$, assuming their relative positions with respect to S_i (or T_j) are described by the vectors $\mathbf{a}$ and $\mathbf{b}$, respectively, we consider their relative positions to be similar when $\Theta(\mathbf{a}, \mathbf{b}) \leq \theta$, where $\Theta(\bullet, \bullet)$ returns the angle between two vectors. To derive the candidate stroke pairs, we associate each stroke of $N(S_i)$ to each stroke of $N(T_j)$ if they have similar relative positions with respect to S_i (or T_j). In our experiments, we use $k = 6$ and $\theta = \frac{\pi}{4}$ for all examples.

3.5. Additional details

The general algorithm of the two-stage FTP-SC technique is given so far, but we still need to consider several issues in practice.

User intervention. While the two-stage FTP-SC technique is able to effectively improve the robustness and accuracy of the stroke correspondence construction, mismatches may still occur in practice. For example, when the geometric shape of the corresponding strokes differ significantly between the initial and target frames,

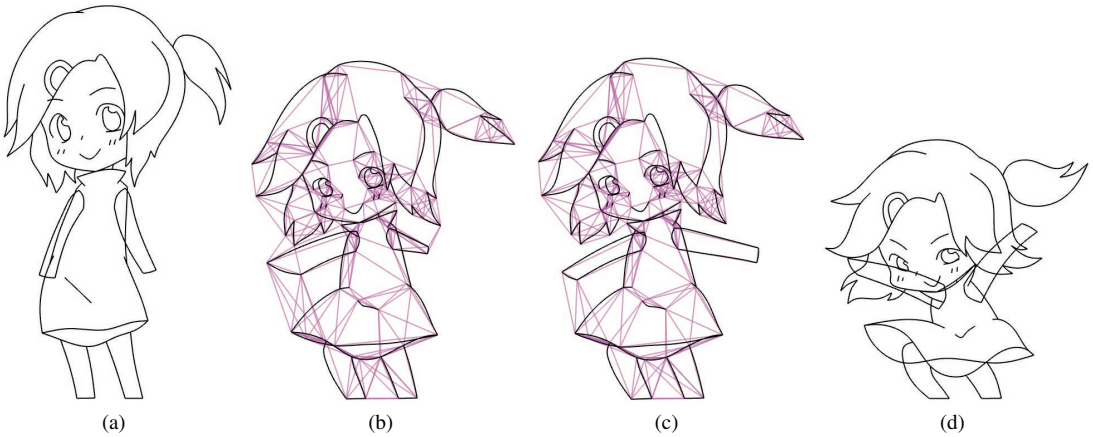

Figure 7: *An inbetweening with large and inconsistent motions. (a, d) The initial and target key frames; (b, c) Interpolation result at $t = 0.5$ where the context mesh at this interpolating time is shown in purple line. In (b), the edge connections of the context mesh are derived using the approach of [Yan18], while the invalid edge connections are removed in (c) by using our data-dependent strategy. Note that in (c) four edge connections between the left arm and lower skirt are not removed under the threshold D, since the motions of the left arm and the lower skirt are visually consistent (see the live example in the accompanying video).*

the SI-component in the stage one may detect a wrong correspondence as its initial seed, possibly leading to a poor correspondence accuracy, as shown in the example 'A' of Figure 11. Fortunately, the FTP-SC technique enables us to incorporate the user correction in an easy way, i.e., simply replacing the initial seed in the SI-component of stage one with the user-specified one(s). Such incorporation makes the user intervention very efficient, since it allows one user correction to have more influence and control over the correspondence results. As shown in our experiments, such user correction scheme also exhibits user-friendly 'moving forward' property which is important in actual productions.

Extensibility. Note that the proposed two-stage FTP-SC technique is independent of the practical implementations of the stage one and stage two. The new strategies can be easily incorporated into the two-stage scheme in a 'plug-in' and 'plug-out' manner. For example, if the key frames correspond to very clean drawings and have similar spatial relationships between the strokes, the strategy of [WNS*10] can be employed in the stage one to generate the stroke correspondences with high confidence, instead of the fuzzy topology preservation. In the stage two, when the spatial distribution of the strokes is uniform in the key frames, we can also obtain the neighborhood information of a stroke from the strokes whose distance to the stroke is within a threshold.

4. Generating Inbetweening Sequences

In order to generate the inbetweening sequences between the initial and target key frames, vertex correspondence between each pair of corresponding strokes is necessary. A straightforward approach is to align the vertices of the strokes using the proportional arc-length principle, as adopted in [WNS*10]. Many more complex and elegant algorithms have also been proposed [SG92,CEBY97,SKK03]. We use the feature-based approach introduced in [LWZ*04, YF09], where salient points (endpoints and curvature extrema) are detected on the strokes and then associated by the dynamic programming technique. This approach can guarantee that the visually salient features are associated between the strokes. It is also very fast, since it operates on the salient points whose number is much less than the number of the strokes' vertices.

4.1. Data-dependent stroke interpolation strategy

For the generation of inbetweening sequences, we use the context-aware technique [Yan18] to interpolate the corresponding strokes between the key frames. This technique preserves the structure between the strokes and is able to incorporate the logarithmic spiral trajectory [WNS*10] by specifying the path points. For brevity, we will only recall the core idea of the approach and refer the reader to [Yan18]. The core idea is to build a pair of the so-called compatible context meshes to represent the spatial structure between the strokes of the initial and target key frames. A context mesh consists of vertices and edge connections between vertices, as shown in Figure 7. During interpolation, it is the edge connections that take the responsibility of preserving the spatial structure between the strokes.

In practice, we find that the edge connections may introduce unacceptable distortions when locally large and inconsistent motions are involved. An example is shown in Figure 7, where the motion of the girl's arms is large and is obviously inconsistent with that of her hair; however, the edge connections between the arms and hair will tie them together during the interpolation, ultimately leading to distortions, as shown in 7(b). In our solution, we detect such edge connection by analyzing its vertex motion. For an edge connection, let (p_1, p_2) and (q_1, q_2) be its vertex positions in the initial and target key frames, respectively. Then, the displacements for the two vertices of the edge connection are: $q_1 - p_1$ and $q_2 - p_2$, denoted as d_1 and d_2, respectively. We denote DIF as the difference between the two displacements and measure it by: $DIF = \| d_1 - d_2 \|$.

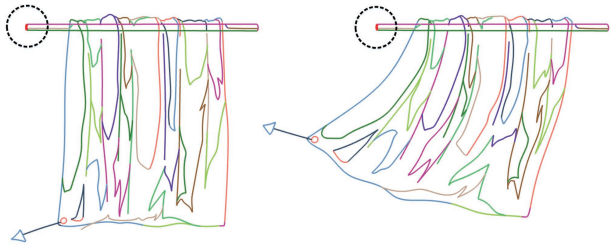

Figure 8: *Correspondence result of a simple case where the automatically detected initial (red) seed is in the dashed circles.*

Given a threshold D, we consider an edge connection to be invalid when $DIF > D$, and remove it from the compatible context meshes, as shown in Figure 7(c).

However, it is difficult to pre-define a default value for the threshold D. If the D is too small, it may remove the desired connections; on the other side, the invalid connections may not be detected when the D is too large. To tackle this problem, we propose a data-dependent strategy to determine D. Given a pair of compatible context meshes, let $\{DIF_1, DIF_2, \ldots, DIF_k, \ldots\}$ be the set of displacement differences between vertices of their edge connections. It is reasonable to assume that these vertex displacement differences follow a normal distribution. We compute the mean and standard deviation of the data set, and denote them as μ and σ, respectively. Since the locally large and inconsistent motions are usually few, the invalid edge connections would correspond to a small probability event, i.e., the number of DIF_k that is greater than D is small. Therefore, we can determine D as: $D = \mu + 3\sigma$. In experiments, we find that this strategy works very well for the detection of invalid edge connections, as shown in Figure 7(c).

5. Results

In this section, we present the results of experiments with the two-stage FTP-SC technique. Our goal is to evaluate the effectiveness of the proposed technique against practical requirements, including accuracy, user interaction, stability, and efficiency. All experiments were run on a 3.6GHz Intel Core(TM)i7 processor (32GB RAM) with single thread. The proposed technique is very fast. For examples shown in the paper, the time for stroke correspondence is about $12 \sim 49$ ms, while the interpolation rate is about $90 \sim 322$ FPS.

Accuracy: In this experiment, we tested the FTP-SC on examples of different complexity. To evaluate the accuracy, the FTP-SC, except specifically stated, was run automatically without user intervention. We start with easy cases where the strokes are similar in both geometric shape and spatial relationships. Figures 4 and 8 show the correspondence results. In these cases, all of the source strokes are correctly matched to their corresponding target ones. It shows that it is sufficient for the FTP-SC to use the information on fuzzy topology, geometric shape and neighborhood to construct a full stroke correspondence. In Figure 9, the input data are harder than the previous ones because the geometric shape changes significantly between some of the corresponding strokes. we can see that all of the corresponding strokes are matched to each other even

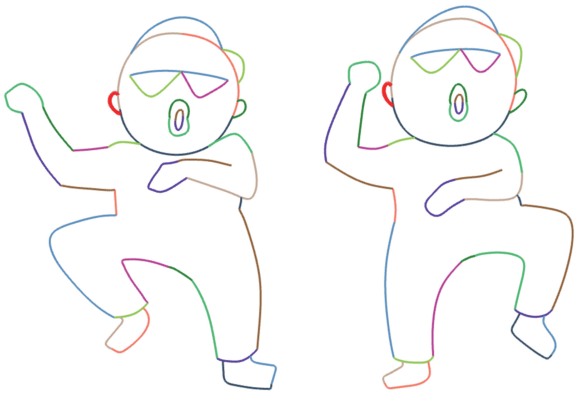

Figure 9: *Correspondence results of key frames with large geometric variation where the initial seed is shown in red thick line.*

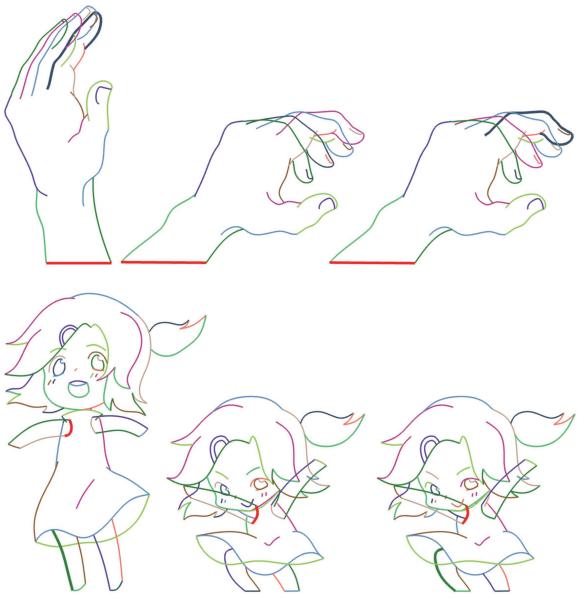

Figure 10: *Correspondence results of key frames with large spatial variation. At each row, the left and middle are the automatically-generated result. In top row, the right shows the correct result where one user correction is sufficient. Moreover, as shown in the right of the bottom row, when one user correction is applied on the girl's left leg, i.e., the strokes in green thick line, all of the mismatches on the girl's lower body are corrected. Note that the user corrections are shown in thick line.*

when some of them have very different shape, e.g., the strokes on the lower body, which shows that the two-stage FTP-SC can effectively accommodate the geometric variations.

Figure 10 shows results of particularly hard case where large geometric and spatial variations between strokes occur. The example in the top row is relatively simple because the number of the strokes is small (20 strokes at each keyframe). For this example, 4 mismatches are generated (80% accuracy) and only one user-correction

Table 1: *Statistical information of User Testing*

Fig. 11	#Mismatch (Accuracy)	Number of user corrections				
		User 1	2	3	4	5
A	31 (0%)	2	3	2	3	1
B	37 (81%)	7	8	8	7	8
C	33 (76%)	8	13	12	12	11
D	4 (93%)	2	2	2	2	2

is needed for a fully correct correspondence. In the bottom row, the key frames have many large shape changes and spatial variations, and thus are more challenging for the correspondence algorithm. For this case, the automatic FTP-SC generates 31 mismatches (58% accuracy) and the fully correct correspondence result can be obtained with 9 user corrections, as shown in Figure 13.

Comparison. It is very difficult for the state-of-the-art techniques of [WNS*10, Yan18] to handle all of the cases mentioned above. For example, it is impossible for the approach of [WNS*10] to construct the stroke correspondences in Figure 10, since the spatial relationships between the strokes have a significant change such that the stroke intersection and connectivity between the keyframes are totally incompatible in some portions. In addition, due to the large shape variation in Figure 9 and at the bottom row of Figure 10, the approach of [Yan18] is unsuitable for these cases, as demonstrated in the following experiment.

User Interaction and Stability: In 2D animation, complex and exaggerated motions tend to cause large geometric or spatial variations, such that mismatches are unavoidable for the automatic correspondence algorithms. Therefore, user intervention is necessary. Our FTP-SC technique provides an efficient user correction scheme and exhibits user-friendly 'moving forward' property which is important in actual productions. In this experiment, we evaluate these properties for the FTP-SC technique by conducting the following user study.

Stage I: User Testing. We invited 5 users, 3 men and 2 women, to evaluate the user interaction of the FTP-SC. One of the participants is an animator and the others are graduate students. Each of participants was given 4 examples which have different complexity, as shown in Figure 11. For each example, mismatches occur in the first automatic run of the FTP-SC due to the geometric and spatial variations, as shown in the second column of Table 1. Then, each user was asked to correct the results using the FTP-SC where the current mismatch information after each user correction would be recorded by the system. Table 1 lists the total number of user corrections which are made by each participant in order to achieve a fully correct correspondence result for each example. The dynamic mismatch information after each user correction is illustrated in Figure 12. We can conclude that though the process of user interaction may change with the different user correction behavior and example complexity, the FTP-SC generally enables the user to correct mismatches in an efficient and 'moving forward' way.

Stage II: Comparison. To further verify that the FTP-SC is more accurate and allows a more effective user interaction, we carried out a second stage of the user study. We gave the animator two examples of different complexity, i.e., that of Figure 9 and the bottom

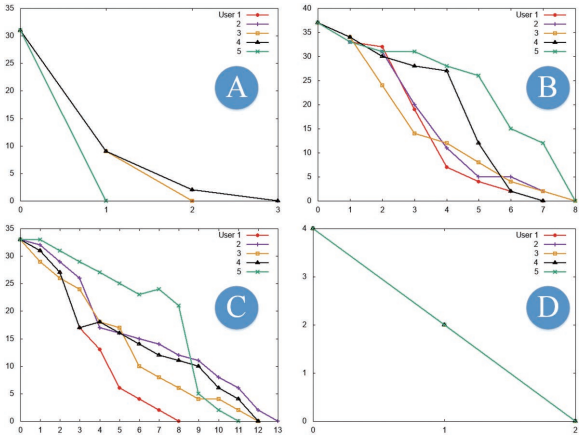

Figure 12: *User Testing: dynamic mismatch information of each example during the user interaction process of each participant. For each chart, the x-axis represents the number of user corrections, while the y-axis shows the number of mismatches in the current correspondence result.*

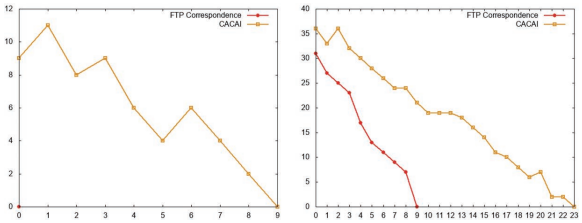

Figure 13: *Comparison between the FTP-SC and the correspondence approach of the CACAI [Yan18]. The left chart corresponds to the example in Figure 9 and the right chart corresponds to the example at the bottom row of Figure 10, where x-y axis is the same as in Figure 12. Note that the curve of FTP-SC in the left chart corresponds to a point at (0,0) since no mismatch occurs in FTP-SC for this example.*

row of Figure 10, and ask him to establish the stroke correspondence using the correspondence method of the CACAI [Yan18] and the FTP-SC, respectively. The dynamic correspondence information is shown in Figure 13. It is shown that with the FTP-SC the animator needs 0 and 9 user corrections respectively to establish the correspondence for the given examples, while he needs 9 and 23 user corrections respectively by using the approach of [Yan18]. Additionally, it is also shown that the user interaction in the approach of [Yan18] may often 'turn back' but not 'move forward', i.e., a user correction may introduce the new mismatches, as illustrated in the left chart of Figure 13. Therefore, we can conclude that the FTP-SC is more accurate and stable than the method of [Yan18], and thus is more efficient and easy to use in practice.

Efficiency and System Evaluation: As stated in Section 2, we embed the FTP-SC technique into a computer-aided inbetweening prototype system. An inbetweening result is demonstrated in Figure 14, and more results are shown in the accompanying video.

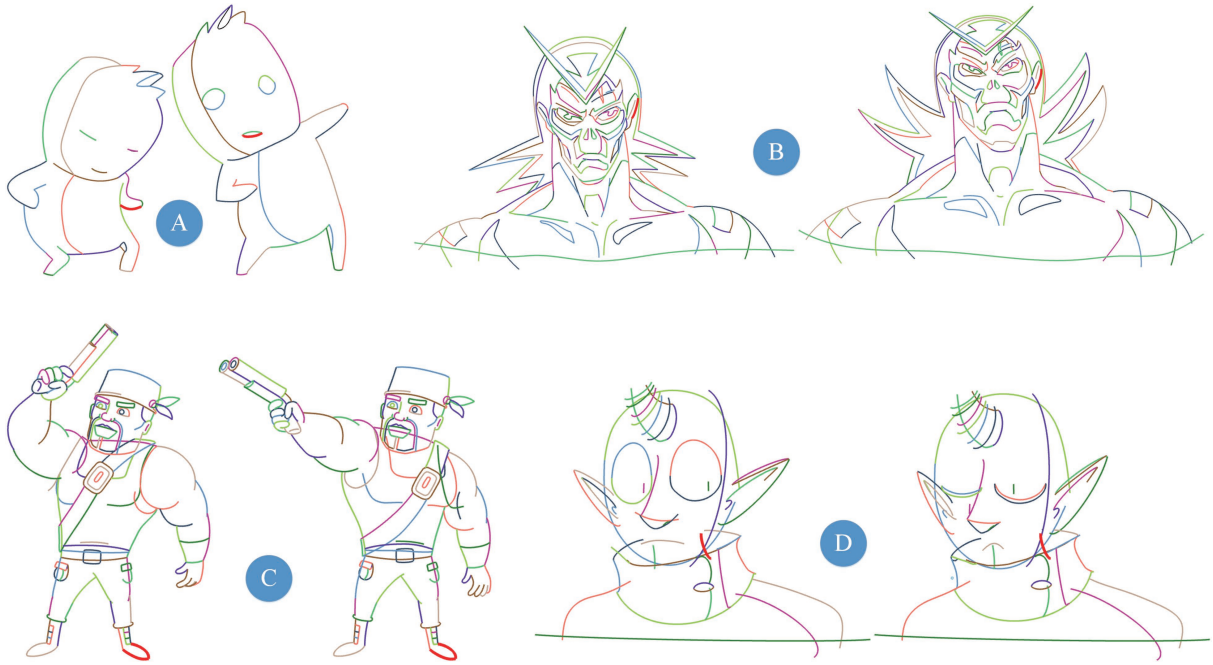

Figure 11: *Examples used in the user testing. For each example, the automatically-generated result is shown. Note that the FTP-SC detects a wrong correspondence as the initial seed for the example 'A', i.e., the red thick strokes, leading to a poor accuracy (0%).*

Table 2: *Efficiency Gain*

Figure	#Inb. Strk	Interactions	%Eff. Gain
8 (Cloth)	220	10	96
10 (Hand)	300	16	95
14 (Gunman)	414	39	91

In actual productions, it has been proven that the practical solution to the stroke correspondence construction problem would be to keep the artist into the loop [WNS*10, Yan18, Lim18]. Though drawing the occluded lines or merging/splitting strokes will induce some extra workload in the key frames, there is still significant efficiency gain. It comes from that (a) from the animators' feedback it is convenient and easy for them to draw the occluded parts in the key frames if necessary, (b) the operation of the stroke merging/splitting is simple, with just a mouse click, and (c) the system automates as much as possible, e.g., it automatically toggles the visibility of the occluded parts by using the occlusion handling tools. Using our prototype system, the inbetweening examples shown in the paper and the accompanying video, which are from real production, were completed in roughly from 6 to 43 minutes, not including the drawing time of the key frames. However, the hand-drawn version of the corresponding inbetween frames in real production took approximately from 90 minutes to 16 hours.

In Table 2, we provide a quantitative measure of efficiency gain of using the system. In the system, the user interaction is not only used in the FTP-SC but also used for adjusting the number of strokes and handling occlusion. We assume that each user interaction has the same workload as drawing a stroke though it is typically much easier than drawing a precise stroke in the final frame. Then, a rough measure of efficiency gain can be taken as one minus the ratio between the interaction count and the total number of strokes in the inbetween frames.

6. Conclusion and Discussion

In this paper, we have presented the FTP-SC, a fuzzy-topology preserving correspondence technique for the stroke correspondence construction between a pair of key frames. A novel two-stage scheme is designed to progressively construct the stroke correspondences. This scheme allows us to control and guarantee the correspondence quality by strictly preserving the so-called fuzzy topology of the strokes in the first stage. A full stroke correspondence, more probably correct, is thus able to be generated in the second stage under the guidance of the results from the first stage. Our method also supports the easy and effective incorporation of the user involvement. Both results and user studies indicate that the FTP-SC is accurate and efficient even for the keyframes with large geometric and spatial variations between the strokes. Furthermore, a data-dependent stroke interpolation strategy has been proposed which generates natural inbetweening sequences for cases where the locally large and inconsistent motions are involved.

Limitations and future work. When the initial seeds are far away from the other strokes, the FTP-SC may not detect enough high-confidence correspondences by preserving the fuzzy topology; thanks to the effective user interaction the high-confidence correspondences can be increased by manually specifying more seeds. The FTP-SC may fail in the very complex key frames, as shown

Figure 14: *An inbetweeing result where the occlusion between the strokes is automatically handled by the system.*

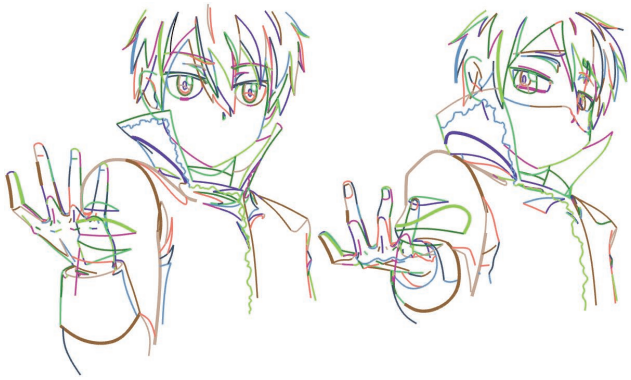

Figure 15: *A failure case where the correspondence result after 11 user corrections is shown. For this case, it's impossible to establish the correspondence with a reasonable amount of user interaction.*

in Figure 15. In this example, the key frames have a large number of strokes where it is even difficult for a human to determine which strokes should be associated with each other between some portions of the key frames, due to the significantly changing geometric shape and spatial relationships between the strokes. One possible solution to such complex cases is to manually decompose the key frames into several relatively simple parts and then establish the stroke correspondence between each pair of the corresponding parts. Moreover, in the current system, we let the user split/merge strokes to eliminate the difference of the strokes' number between the corresponding features. To further reduce the user interaction, (semi-)automatically performing this task is one direction of the future work. To reach this goal, it seems necessary to segment the strokes into small pieces, and then to analyze and match the strokes on the basis of their sub-segments.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. Many thanks to the artists for their lovely drawings, where Fig. 1, 8, 9 is [Courtesy of Liew Hongze], Fig. 2, 4 is [Courtesy of Yovanny Ramirez], Fig. 7 is [Courtesy of Low Zirong], Fig. 10 is [Courtesy of Yovanny Ramirez and Low Zirong, respectively], Fig. 11 is [Courtesy of Antonio Chiu, Sergio Cajazeiras, Eugene Babich, and Martina McKenzie, respectively], Fig. 14 is [Courtesy of Eugene Babich], and Fig. 15 is [Courtesy of Haili Animation]. This research was partially funded by the Science and Technology Agency projects of Zhejiang Province (No. 2016C33171), National Science Foundation of China (No. 61003189), the Technology Agency of the Czech Republic under research program TE01020415 (V3C -- Visual Computing Competence Center), Research Center for Informatics (No. CZ.02.1.01/0.0/0.0/16_019/0000765), and the Fulbright Commission in the Czech Republic.

References

[BW76] BURTNYK N., WEIN M.: Interactive skeleton techniques for enhancing motion dynamics in key frame animation. *Commun. ACM 19*, 10 (1976), 564–569. 1

[BZBM*16] BEN-ZVI N., BENTO J., MAHLER M., HODGINS J., SHAMIR A.: Line-drawing video stylization. *Computer Graphics Forum 35*, 6 (2016), 18–32. 1

[Cat78] CATMULL E.: The problems of computer-assisted animation. *SIGGRAPH Comput. Graph. 12*, 3 (1978), 348–353. 1

[CEBY97] COHEN S., ELBER G., BAR-YEHUDA R.: Matching of freeform curves. *Computer-Aided Design 29*, 5 (1997), 369 – 378. 6

[CG97] COHEN S. D., GUIBAS L. J.: Partial matching of planar polylines under similarity transformations. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms* (1997), SODA '97, pp. 777–786. 4

[DRvdP15] DALSTEIN B., RONFARD R., VAN DE PANNE M.: Vector graphics animation with time-varying topology. *TOG 34*, 4 (2015), 145:1–145:12. 1

[Dur91] DURAND C. X.: The toon project: Requirements for a computerized 2d animation system. *Computers and Graphics 15*, 2 (1991), 285 – 293. 1

[FBC*95] FEKETE J.-D., BIZOUARN E., COURNARIE E., GALAS T., TAILLEFER F.: Tictactoon: A paperless system for professional 2d animation. SIGGRAPH '95, pp. 79–90. 1

[IBiA09] III W. V. B., BARLA P., I. ANJYO K.: Compatible embedding for 2d shape animation. *TVCG 15*, 5 (2009), 867–879. 1

[JT95] JOHNSTON O., THOMAS F.: *The Illusion of Life: Disney Animation*. Disney Editions, 1995. 1

[Kor02] KORT A.: Computer aided inbetweening. NPAR '02, pp. 125–132. 1, 2

[LCY*11] LIU D., CHEN Q., YU J., GU H., TAO D., SEAH H. S.: Stroke correspondence construction using manifold learning. *CGF 30*, 8 (2011), 2194–2207. 1

[Lim18] LIMITED C. P.: Cacani: 2d animation and inbetween software. https://cacani.sg/, 2018. 1, 2, 3, 9

[LWZ*04] LIU L., WANG G., ZHANG B., GUO B., SHUM H.-Y.: Perceptually based approach for planar shape morphing. In *PG 2004* (2004), pp. 111–120. 4, 6

[NSC*11] NORIS G., SÝKORA D., COROS S., WHITED B., SIMMONS M., HORNUNG A., GROSS M., SUMNER R. W.: Temporal noise control for sketchy animation. NPAR '11, pp. 93–98. 1

[QSST*05] QIU J., SOON SEAH H., TIAN F., CHEN Q., WU Z.: Enhanced auto coloring with hierarchical region matching. *CAVW 16*, 3-4 (2005), 463–473. 3

[Ree81] REEVES W. T.: Inbetweening for computer animation utilizing moving point constraints. *SIGGRAPH Comput. Graph. 15*, 3 (1981), 263–269. 2

[SDC09] SÝKORA D., DINGLIANA J., COLLINS S.: LazyBrush: Flexible painting tool for hand-drawn cartoons. *Computer Graphics Forum 28*, 2 (2009), 599–608. 3

[SG92] SEDERBERG T. W., GREENWOOD E.: A physically based approach to 2D shape blending. *ACM Computer Graphics 26*, 2 (1992), 25–34. 6

[Shl14] SHLENS J.: A Tutorial on Principal Component Analysis. *ArXiv e-prints* (2014). arXiv:1404.1100. 5

[SKK03] SEBASTIAN T. B., KLEIN P. N., KIMIA B. B.: On aligning curves. *PAMI 25*, 1 (2003), 116–125. 4, 6

[SWT*05] SEAH H. S., WU Z., TIAN F., XIAO X., XIE B.: Artistic brushstroke representation and animation with disk b-spline curve. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology* (2005), ACE '05, pp. 88–93. 3

[WNS*10] WHITED B., NORIS G., SIMMONS M., SUMNER R., GROSS M., ROSSIGNAC J.: Betweenit: An interactive tool for tight inbetweening. *CGF 29*, 2 (2010), 605–614. 1, 2, 3, 4, 6, 8, 9

[Yan18] YANG W.: Context-aware computer aided inbetweening. *TVCG 24*, 2 (2018), 1049–1062. 1, 2, 3, 4, 6, 8, 9

[YBS*12] YU J., BIAN W., SONG M., CHENG J., TAO D.: Graph based transductive learning for cartoon correspondence construction. *Neurocomputing 79*, 0 (2012), 105 – 114. 1

[YF09] YANG W., FENG J.: 2d shape morphing via automatic feature matching and hierarchical interpolation. *Computers and Graphics 33* (2009), 414–423. 4, 6

[Zad65] ZADEH L.: Fuzzy sets. *Information and Control 8*, 3 (1965), 338 – 353. 3