Live Sketch: Video-driven Dynamic Deformation of Static Drawings

Qingkun Su[1], **Xue Bai**[3], **Hongbo Fu**[2], **Chiew-Lan Tai**[1], **Jue Wang**[3]
[1]Department of Computer Science and Engineering, HKUST
[2]School of Creative Media, City University of Hong Kong
[3]Megvii Inc. US

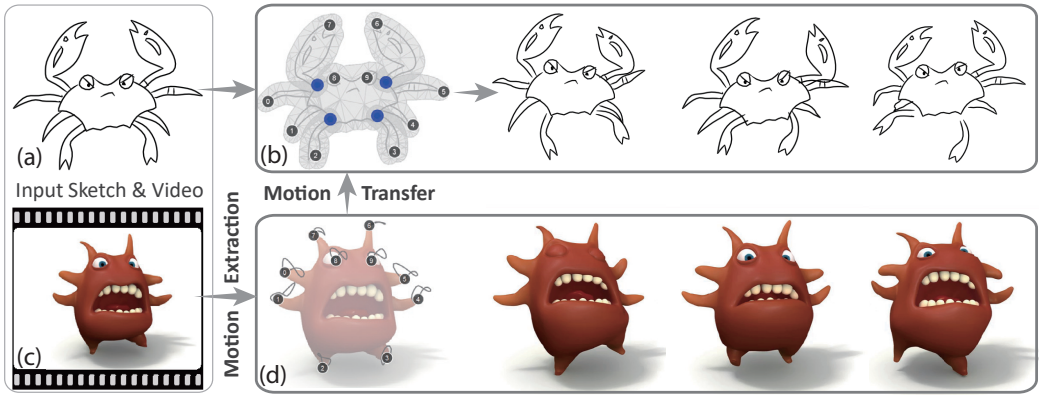

Figure 1. We present *Live Sketch*, an interactive system for users with no or little animation skills to create vivid dynamic deformation of a sketch (a) by first extracting object motion from a video sequence (bottom), and then transferring the motion to the sketch (top). Please see the animation in the accompanying video. Video credit: Albert Ziganshin.

ABSTRACT

Creating sketch animations using traditional tools requires special artistic skills, and is tedious even for trained professionals. To lower the barrier for creating sketch animations, we propose a new system, *Live Sketch*, which allows novice users to interactively bring static drawings to life by applying deformation-based animation effects that are extracted from video examples. Dynamic deformation is first extracted as a sparse set of moving control points from videos and then transferred to a static drawing. Our system addresses a few major technical challenges, such as motion extraction from video, video-to-sketch alignment, and many-to-one motion-driven sketch animation. While each of the sub-problems could be difficult to solve fully automatically, we present reliable solutions by combining new computational algorithms with intuitive user interactions. Our pilot study shows that our system allows both users with or without animation skills to easily add dynamic deformation to static drawings.

Author Keywords

Sketch animation; video examples; motion extraction; motion transfer; dynamic deformation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CHI 2018, April 21–26, 2018, Montreal, QC, Canada

© 2018 ACM. ISBN 978-1-4503-5620-6/18/04…$15.00
DOI: https://doi.org/10.1145/3173574.3174236

INTRODUCTION

In recent years, there has been a new wave of research on developing intelligent user interfaces to help novice users create artistically expressive sketches and drawings. Representative examples include *ShadowDraw* [31], *EZ-Sketch* [42], *ColorSketch* [32], and the repetition autocompletion system [49]. By combining intelligent computational algorithms with intuitive UI controls, these systems enable novice users with little or no formal artistic training to create static drawings of high quality with minimal user effort, which otherwise cannot be achieved using traditional tools.

Despite the fact that creating static drawings has been made easier by these tools, creating high-quality 2D animations remains both difficult and time-consuming. Traditional animation tools, such as *Adobe Flash* and *Toon Boom* software, require accurate motion keyframing, which is a tedious and labor-intensive process even for experienced artists. To avoid this, new interactive tools have been proposed to allow artists to specify motion in more creative ways, such as spatio-temporal sketching [20], or using specially-designed cursive gestures [44]. Still, using these tools requires not only intensive training, but also animation skills that novice users do not possess. It is very hard for novice users to mentally map a desired continuous object motion to static keyframe drawings that are discrete and sparse.

On the other hand, there exist many videos with rich motion information, which can be potentially extracted and used for animating 2D sketches. Proper video examples should contain mainly 2D motion that can be semantically mapped to

sketches. An ideal video-driven sketch animation workflow would be: 1) given an arbitrary sketch, search for proper video examples from a large dataset of videos; 2) extract and transfer the motion of the retrieved relevant videos to the sketch. While both the subproblems are interesting, we aim to address the second subproblem and focus on how to effectively reuse the motion of a given video example for animating sketches which might have different shapes from the video objects. Our work thus shares the same objective as Bregler et al. [10] in producing video-driven animation. However, their technique requires a series of user-prepared target key-shapes corresponding to the source key-shapes from a video, and transfers interpolation weights to achieve keyframe-based animation, while ours needs only a single sketch.

In this paper, we present *Live Sketch*, a new intelligent interface to assist non-professional users in animating a single sketch through expressive dynamic deformation using corresponding video example(s), as illustrated in Fig. 1. Our main idea is to extract the desired motion from an existing video as a sparse set of moving control points (Fig. 1 (d)), which are then used to drive the deformation of the mesh on which the sketch is embedded (Fig. 1 (b)). We propose new computational algorithms and combine them with intuitive user controls for robust and controllable motion extraction and transfer so that our system is friendly even for users with no or little animation skills.

In the motion extraction stage, we propose a new semi-automatic video tracking approach, which is able to track complex non-rigid object motion and is robust against occlusion and ambiguity. Our approach extends the existing techniques [2, 11] by introducing an additional component that allows our tracking method to handle the drifting problem due to ambiguity and topology changes. It takes a minimal set of user-specified, semantically-meaningful control points on the first frame, and automatically tracks their positions throughout the video. This approach can also incorporate sparse user edits as hard constraints to refine the tracking results in difficult cases.

In the motion transfer stage, we animate an input sketch by mesh-based deformation guided by the tracking results (Fig. 1(b)), while providing users a set of tools to fine-tune the deformation effects such as local rigidity of the motion. We further show how to interactively decompose a sketch into multiple layers to handle complex object motion patterns, such as self-occlusion and topology change.

We have conducted a pilot user study to evaluate the usability and expressiveness of our system, demonstrating very positive results. Our work presents the following main contributions:

- The first efficient and user-friendly 2D animation tool that animates a single drawing by reusing the motion information of a video example.
- A new sparse point tracking method that is robust against occlusion and ambiguity, and allows easy user control.
- A new stroke-preserving mesh deformation method for animating sketches, with constraints for better preserving the shape of user-specified strokes.

RELATED WORK

Digital sketching. There has been extensive research on how to assist artists or novice users in creating sketches. Some methods refine or guide users' sketching by analyzing a crowd-sourced set of images [31], low-quality drawings [19], or even a single image being traced over [42]. Xing et al. [49] present a painting system that auto-completes tedious repetitions. These methods have focused on assisting users in producing individual static drawings. Animations, on the other hand, rely heavily on users' sense of both space and time [41, 9]. Our system aims to relieve users from the tedious control required of motion authoring by making use of video examples.

Sketch-based animation. Several animation tools with intuitive interaction techniques have been proposed in recent years. A common objective is to develop tools that aid interactive specification of motion trajectories. Willett et al. [48] and Thorne et al. [44] present a set of transition motions or cursive gestures for sketching a significant variety of motions for 2D characters. Guay et al. [20] introduce a space-time sketching concept that enables animators to create a movement, including shape deformation over time, by sketching only a single stroke. Other techniques use predefined motion effects (particle, waves, smoke, etc.) to animate 2D static drawings [27, 50]. These methods focus on specific animation effects or a small number of animation styles. Other methods aim at interactively predicting what users might sketch next using temporal coherence [46, 1, 51]. But they still require many manual sketching operations for each keyframe. Unlike these methods, our *Live Sketch* is able to generate complex non-rigid deformation for a static sketch by retargeting motions from videos with a small amount of user assistance.

Animation tools using video. There have been existing tools that also utilize videos for animation creation, as videos provide abundant motion. One way is to interactively produce cutout-style animations [6, 22] by processing color or RGB-D videos. Another category is non-photorealistic rendering (NPR) techniques, which transform images or videos into artistically stylized renderings. For example, Ben-Zvi et al. [8] develop a data-driven method that automatically converts videos and CG animations to stylized animated line drawings. Please refer to [30] for an insightful survey on this topic. These methods focus on changing the rendering styles of input videos instead of extracting the underlying motion for animating new line drawings. Santosa et al. [38] provide a user interface to animate a sketch painted over the video using optical flow, and thus their system can animate only drawings of similar structure as input videos.

Animations can also be created by tracking a contour of an object in the video and using its temporally coherent information. For instance, Bregler et al. [10] track the motion from traditionally animated cartoons and retarget it onto 3D models or 2D drawings. It takes a keyframe-based approach and requires a series of corresponding input and output key shapes for deformation interpolation, leading to more expressive results than ours but at the cost of significantly additional user inputs. In contrast, we take a non-keyframe-based approach to transfer dynamic movement of control points to a single

static drawing, thus requiring significantly less user's input and lowering the barrier for novices. Their method further requires a sub-part decomposition procedure, which is relatively easy for cartoon characters but is hard for real video objects. Agarwala et al. [1] present a rotoscoping approach that uses computer vision techniques to reduce user interaction in the contour-tracking process. The extracted contours from this system can be used to drive user sketches for video stylization. Our system uses sparse control points as motion representation, which are much easier to track accurately than contours, and thus requires much less computation and user intervention in the tracking process.

Face and body driven animation. Our system is closely related to automated human-performance-driven animation. MoCap [29] is the industry standard for capturing 3D human pose for animation. It requires complex hardware setup that is not easily accessible to normal users. Fitting 3D face or body models to images and videos to extract human motion has been extensively studied [13, 15, 47, 17, 35, 43]. It has enabled new real-time face- or body-driven animation systems such as Adobe Character Animator. Park et al. [34] directly synthesize human motion in video by analyzing exemplars. However, these systems are limited to facial expression or body motion only. In contrast, our system focuses on extracting and transferring 2D dynamic deformation for a wide range of objects. Furthermore, it uses a regular video as motion source, which is relatively easy to capture and vastly available online.

Cross-modality motion retargeting. Many systems have been proposed to transfer motion across different modalities. For example, Yamane et al. [52] generate animations of non-humanoid characters from human motion capture data. Hornung et al. [23] transfer 3D motion data to 2D image objects, while Diener et al. [18] retarget 2D motion fields for the animation of 3D plant models. Zhou et al. [55] propose to use the volumetric graph Laplacian for large 3D mesh deformation, allowing motion to be transferred from exaggerated 2D animations to 3D meshes. Bernhard et al. [36] focus on the animation of 3D articulated creatures using video examples. Our system is a pure 2D system, and thus is applicable even when 3D models are not available.

Video tracking. Previous single object tracking methods [33, 5, 14, 12] often utilize weak local features of sampled patches to track a single object at the bounding box level. The box position is determined from all patches through a voting or statistical aggregation method. Our application however requires more than a bounding box: it requires tracking a sparse set of control points to capture major structural deformation.

Sparse feature tracking is a fundamental component for video analysis that has been extensively studied in the computer vision literature. Our tracking method is built upon well-known tracking techniques such as good-feature-to-track [40] and Struck [21]. However, most previous tracking methods track each feature point individually without considering their geometrical relationships. This is valid for most computer vision tasks where hundreds of features are being tracked on each frame, and tracking errors are eliminated by outlier removal methods such as RANSAC. However, since we only track a few semantically-meaningful control points, tracking failure of a single point would result in large distortion. This problem is alleviated by our joint-optimization over multiple points.

OUR SYSTEM

System Overview

Inputs. As illustrated in Fig. 1, our system takes as input a static sketch drawing represented as a *vector graphics*, and a video example that contains the desired motion of a moving object for animating the sketch. The moving object should be entirely visible throughout the video. The source motion should be approximately 2D and within the image plane, since deforming a 2D sketch in 3D can easily lead to (e.g., twisting) artifacts. Furthermore, since what we care about is the motion of the foreground objects, we assume that the camera of the input video is static. Existing camera motion prediction methods such as [54, 3] may help to remove camera motion.

Motion Representation. A key question for our motion transfer task is the representation of object motion. Given the complexity of natural object motion, it is difficult to describe it in parametric forms, as similarly done in [10]. For non-parametric representation, we could choose either dense or sparse motion flow. The former can be extracted by optical flow methods, while the latter be computed by using sparse feature tracking. We have experimented with both representations, and concluded that sparse feature point trajectories are more suitable for our task. The reason is that dense flow often contains tracking errors that could distort the sketch during animation, and these errors are hard to correct via a user interface, given that motion is defined on every pixel. On the contrary, sparse trajectories allow us to enforce strong spatial and temporal smoothness in motion, resulting in more stable animation. It further allows quick user editing by interactively moving the positions of these points. The downside of using sparse representation is that motion with finer granularity cannot be captured and transferred. This problem, however, can be partially alleviated by adjusting the spatial distribution of control points, as described in Section "Control Point Extraction".

User Interface

Our system uses a two-stage workflow to animate user's 2D static sketch: (1) *motion extraction* from the input video; and (2) *motion transfer* to the sketch. We first introduce the user interfaces for these two stages, as shown in Fig. 2.

Motion Extraction

We provide an intuitive user interface to extract the desired motion from the video. The user first defines a set of structural control points inside the object, which are automatically tracked through the entire video. The details about control point extraction are described in Section "Control Point Extraction" and our tracking method is described in Section "Motion Extraction". The trajectories of these control points represent the object motion, and will be applied to the corresponding control points on the sketch for animation. Given that automatic tracking is not always perfect, our interface also provides manual tools for fine-tuning the extracted motion by adding,

removing or repositioning the control points. The motion trajectories are updated in real time and visualized to the user during the editing process (see Fig. 2(left)).

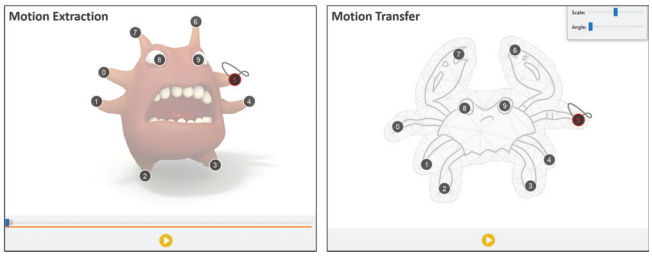

Figure 2. Our user interfaces. Left: motion extraction UI, where control points are defined and tracked in the input video. Right: motion transfer UI, where the position and motion of the control points are used to drive the sketch. The corresponding control points on the video and sketch are shown with the same numbers. Video credit: Albert Ziganshin.

Motion Transfer

The system automatically maps the control point trajectories to the user's sketch (Section "Control Point Transfer"). These mapped control points are used as constraints in a mesh deformation process to drive the animation, as detailed in Section "Motion Transfer". The user can immediately review the animation result without any additional manual operation.

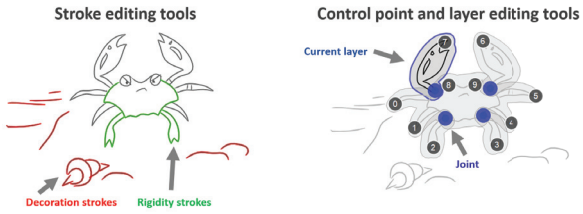

Figure 3. Interactive tools for motion transfer. Left: the user can label selected strokes as decoration strokes (i.e., no motion), or add strong shape rigidity constraints to strokes. Right: the user can group several strokes as a new layer to animate them independently from other strokes, or add/move automatically initiated joints in the layer intersection regions.

Our interface provides interactive tools for fine-tuning the automatically-generated sketch animation, as shown in Fig. 3. First, it allows the user to adjust the spatial positions of the control points on the sketch, whose initial positions are automatically inferred from the control points on the video object. Another useful tool is "Rigidity Stroke", which can be used to alleviate local shape distortion during the animation. The user can also use a "Decoration Stroke" tool to mark sketch components that are to remain stationary, like those for background decoration or a "Pin Point" tool to add stationary control points to the sketch (black points with no IDs in Fig. 10(a)).

One important feature of our system is its ability to decompose a sketch and animate it in multiple layers (see Fig. 3, Right). This is particularly useful for handling self-occlusions, which are common for video objects with complex motion. To create such layers, the user can use a "Layer Brush" to paint over strokes that should be grouped together, based on the semantics of the object and its motion. We also provide an interface for users to refine the automatically generated joints between layers to fine-tune the animation. Please refer to Section "Multi-layer Animation" for more details.

Control Point Extraction

In our system, object motion is represented by a set of sparse control points which are tracked throughout the video, and used as constraints for deforming the sketch for animation. We have done extensive studies on automatically computing these points. We have tried to use image features such as SIFT or SURF to find good points to track, and also explored determining the points based on the initial object shape. In the end we conclude that although these automatic approaches do work well in some cases, they cannot handle a wide variety of examples well. This is due to the following requirements on these control points:

R1. They should be sparsely distributed across the entire object shape to cover the animated region.

R2. Their distribution should reflect the degree of freedom of the underlying motion. For example, one control point could be enough for a flying baseball; however, more are needed for a fly bird to capture its wing motion.

R3. They should be easy to track in the video, such as image corners, which have large appearance dissimilarity from their neighborhood.

R4. The number of control points should be minimal for easy user adjustment.

R5. They should obey the semantics of the object, e.g., respecting semantic symmetry (such as the tentacles of the running virus in Fig. 1).

R1 suggests having a sufficient number of control points to capture the underlying motion of the object. On the other hand, R4 aims to minimize the number of control points. This suggests distributing the control points adaptively with respect to the underlying motion (R2), while making sure each control point is semantics-aware (R5) and located in regions with salient image features (R3). While the individual requirements might be satisfied using local image feature space, motion analysis and shape analysis, satisfying all the requirements in a balanced manner is more difficult and requires deep semantic understanding of the object and its motion. On the other hand, it is relatively easy for the user to preview the video and determine the semantics of the object and its motion. We thus resort to a manual procedure for specifying control points. We employ a robust tracking algorithm that jointly optimizes all tracking points, as detailed in the next section.

Discussion. The distribution of the control points determines the granularity of the motion we extract; fewer control points results in coarser motion, and vice versa. However, having too many control points could be problematic as the tracking failure might increase, and the motion field becomes less stable due to tracking errors. Given that we will enforce temporally-smooth motion fields for animation stability, we have found that a smaller set of control points (one or two dozens) is usually sufficient for a wide variety of examples.

Motion Extraction

After specifying the control points, we track their positions throughout the video. A naïve approach would be to track each control point individually using standard point tracking methods, e.g., KLT point tracker [53]. However, these methods may fail for some points that are hard to track, e.g., points specified in textureless regions, resulting in drifting. Used as hard constraints, these drifted points will later distort the sketch shape in the final animation. Given that automatic tracking failure may always occur no matter how robust the underlying algorithm is, we employ an interactive tracking procedure. We first adopt a dynamic-programming-based trajectory optimization framework [2, 11] to track individual points. When tracking drifting occurs for one point, the user only needs to modify the tracking position of the failed point in a particular frame, which is incorporated as a hard constraint to globally refine the whole trajectory of the failed point. Therefore, the user can quickly refine the tracking results by only editing a few frames. This interactive tracking method is quite efficient and is able to address drifting due to occlusion. To handle the second source of drifting due to ambiguity, we jointly optimize over multiple tracking trajectories.

Specifically, the tracking algorithm optimizes the following energy for each control point trajectory:

$$E_{tr}(\mathbf{x}) = \sum_t \left(\lambda_d d(x_t, x_{t+1}) + \lambda_u u(p_t, p_{t+1}) + \min_k (a(p_t, c_k)) \right), \tag{1}$$

where x_t represents the position of a control point $\mathbf{x}$ in frame t, and p_t denotes the feature descriptor (SIFT in our implementation) of the image patch (we use 9×9 patches) centered at x_t. The energy consists of three parts. The velocity term $d(\cdot) = \|x_t - x_{t+1}\|_2^2$ and update penalty term $u(\cdot) = \|p_t - p_{t+1}\|_2^2$ measure the respective motion intensity and appearance variation between two consecutive frames, to achieve a smooth motion and appearance transition. $a(\cdot) = \|p_t - q_k\|_2^2$ in the third term is a measure of the appearance deviation from the user-specified control point locations $\{c_k\}$, where q_k is the patch feature descriptor of c_k. The user may manually specify two or more control point positions at different frames. We measure the appearance energy of points at other frames using $\min_k(a(p_t, c_k))$, so that they should match at least one of these user-specified control points.

We minimize the energy function in Eq. 1, by finding the shortest path of a graph (Fig. 4). First, the user specifies the locations, $\{c_k\}$, of the tracking point at some frames (black circles). Then, for each remaining frame, we find a set of trajectory candidate positions (green circles), which have small appearance deviation from user-specified positions. Then we build a graph that takes these candidate points and the user-specified points as its nodes. We connect the points of each frame t with those of the next frame $t+1$ as the edges of the graph (gray lines). The weight of each node p_t is its appearance deviation $a(p_t, c_k)$. The weight of each edge is computed by

$$w_e(p_t, p_{t+1}) = \lambda_d d(x_t, x_{t+1}) + \lambda_u u(p_t, p_{t+1}).$$

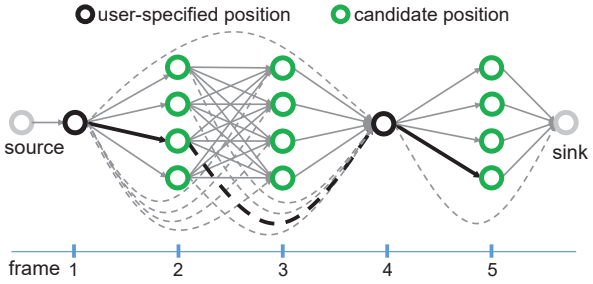

Figure 4. Illustration of our interactive motion extraction algorithm for each tracked point. The user first specifies the position of the associated control point (black circles) at some frames (1&4). For each of the remaining frames (2,3&5), our system then finds a set of candidate positions (green circles) of the tracking trajectory, builds a graph using these points and searches for the shortest path as the optimal tracking trajectory. By adding edges (dot lines) that skip frames when occlusion happens, this method can handle the occlusion problem. For example, the black trajectory is a possible solution when occlusion happens in frame 3.

Therefore, the total cost of the shortest path is the minimum value of $E_{tr}(\mathbf{x})$.

To handle the occlusion problem, we also fully connect the points of each frame with those of its next m frames (dot lines in Fig. 4, we set $m = 10$). The weight of the extra edge between two respective nodes p_t and $p_{t'}$ of frame t and t' ($t+1 < t' \leq t+m$) is

$$w_e(p_t, p_{t'}) = \lambda_o + \lambda_d d(x_t, x_{t'}) + (t'-t)\lambda_u u(p_t, p_{t'}).$$

The parameter values and more details of this method can be found in [2, 11].

Ambiguity Handling

Besides the occlusion problem, another source of drifting is ambiguity, i.e., when two or more tracking points with similar appearance get close to each other. An example is shown in Fig. 6(top). When the two legs of the tiger cross, the two control points become too close to each other, and they end up tracking the same image feature going forward, resulting in a collision.

To solve this problem, we propose a new energy minimization method to jointly optimize the tracking trajectories of multiple points (Fig 5). The main idea is to first generate a set of trajectories for each control point as candidate trajectories (Fig. 5(top)) and then employ a global optimization approach to select the best one for each point by jointly considering multiple points together.

Candidate Trajectories. Given all control points on a keyframe, we first compute the appearance similarity between any two points based on $u(\cdot)$ in Eq. 1. We then group points into several sets, each containing points that have similar appearance. Denote one such set as $S = \{\mathbf{x}^1, ..., \mathbf{x}^n\}$. For each point $\mathbf{x}^i$, we apply the method proposed in [16] to compute a set of candidate tracking trajectories. This method has two major parameters: (1) the maximum overlap between any two candidate trajectories, which we set as $T/4$, where T is the total number of frames; and (2) a tracking cost threshold (set to 4.0 times of the shortest path's cost).

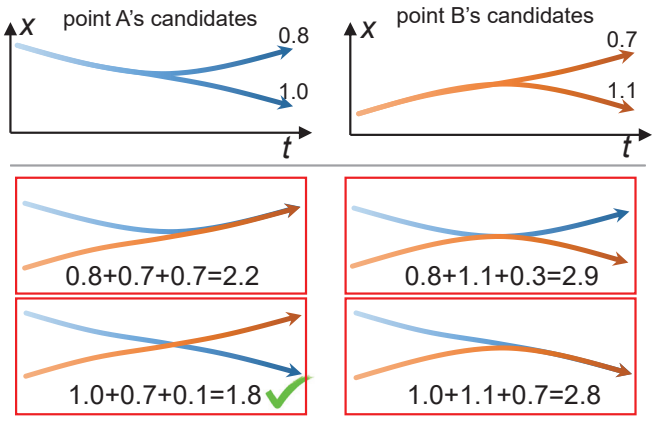

Figure 5. Ambiguity handling. We first generate a set of trajectories for each control point as candidates (top). The tracking cost E_{tr} of each candidate trajectory is shown. A global optimization approach is then applied to find the best combination (bottom) with the minimum sum of tracking energy (E_{tr}) and overlapping energy (E_o), i.e., $E = E_{tr}(\mathbf{x}^A) + E_{tr}(\mathbf{x}^B) + \beta E_o(\mathbf{x}^A, \mathbf{x}^B)$.

Trajectory Optimization. We jointly optimize over multiple tracking points to find the best trajectory for each point. This can be formulated as the following energy minimization problem:

$$\arg\min \sum_i E_{tr}(\mathbf{x}^i) + \beta \sum_{i \neq j} E_o(\mathbf{x}^i, \mathbf{x}^j), \qquad (2)$$

where the tracking energy $E_{tr}(\mathbf{x}^i)$ is measured by Eq. 1. $E_o(\mathbf{x}^i, \mathbf{x}^j)$ is defined as the normalized duration of the overlapping portion of the two trajectories. β is the weight parameter and set to 0.25 in our implementation. Intuitively, this term prevents two similar control points to collapse into a single one, in which case the overlapping portion of the two trajectories will be large.

We use a greedy algorithm to minimize this energy. We first initialize the solution by choosing the trajectory that has the minimal tracking energy for each point (according to Eq. 1). Starting from the point that has the largest total energy according to Eq. 2, we assign to it another candidate trajectory that best minimizes this energy. We repeat this process until the total energy cannot be further reduced.

Note that we do not make any assumption on the background of a video. One tracking result of a video with complex background is shown in Fig. 6 (bottom). In addition, our motion extraction method is a general feature tracking method, and thus is able to track objects involving 3D motion (Fig. 6(bottom) and Fig. 15(right)), though it might require additional user assistance. Such video examples are more useful for other applications, e.g., driving the motion of a 3D model instead of animating a 2D static drawing.

Control Point Transfer

To transfer the extracted motion to the input sketch, we first need to map the control points specified on the video object to the sketch. Because of possibly dissimilar appearance between the sketch content and the video object, existing methods [4, 39] based on local features of the content are not suitable for our scenario. We propose a semi-automatic mapping method based on their contours.

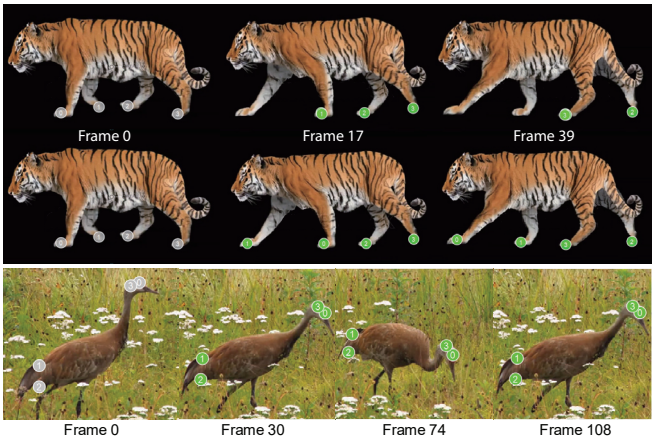

Figure 6. Top: automatically solving tracking ambiguity using multi-track optimization. 1st row: four control points defined on the four legs collide when the legs cross, due to appearance ambiguity; 2nd row: our new tracking method can effectively prevent such ambiguities. Bottom: tracking result for video with 3D motion and complex background with a small amount of user intervention. Video credits: AlphaNature; zontinela.

The contour of the sketch can be extracted using the Active Contour models [26]. The contour of the video object can be extracted by automatic *GrabCut* segmentation [37] with the bounding box enclosing all the tracking points as input. Note that thanks to the requirement R1, the bounding box of the tracking points can typically well cover the video object. This contour-extraction method works well when the background is clean. Otherwise, the user can quickly draw the correct outline using a pen tool (e.g., the example in Fig. 6(bottom)). Then we apply the Shape Context method [7] to build correspondence between the two contours. Next, for a control point inside the video object, we represent its location using the barycentric coordinate of a set of evenly sampled contour points, and compute its location using the same barycentric coordinate on the sketch image. We also provide an intuitive user interface to assist fine-tuning the mapping results (Section User Interface) when the object contour of sketched and video objects is very dissimilar like the example in Fig. 10(b).

Motion Transfer

We use the sparse control point trajectories to drive the final sketch animation. For this purpose, we first embed the sketch into a mesh and then use a mesh deformation method to warp the mesh for animation. The mesh is generated by triangulating an expanded region of the sketch (see Fig. 1). Our expectation for the deformation is twofold: (1) the deformed mesh should closely follow the guidance of the control points to preserve the extracted motion, thus requiring the resolution of the mesh to be high enough to afford the detailed motion; and (2) the original sketch strokes should not be distorted too much to avoid unnatural deformation artifacts.

Traditional controlled mesh deformation methods, such as the as-rigid-as-possible (ARAP) mesh deformation [24, 25], are designed to keep only the rigidity of the mesh triangles (see Fig.8), not necessarily preserving the shape of the embedded

strokes, especially those crossing multiple triangles. Moreover, it is desirable to have parts of the sketch being more rigid, while other parts being flexible to capture large motion. Fig. 7 shows such an example. With the original ARAP deformation, the body of the bird is undesirably distorted by the large motion of its wings. The user can apply a stronger rigidity constraint on the bird's body while applying a weaker rigidity to its wings to better capture the intensive flapping motion.

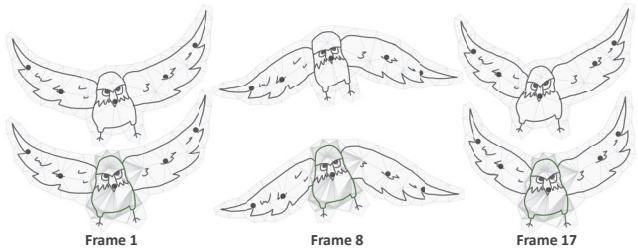

Figure 7. Sketch deformation results with (bottom) and without (top) applying our stroke-preserving ARAP mesh deformation method. The stroke in green is set to be more rigid.

We develop a *stroke-preserving ARAP* to attain user-specified local rigidity on the input strokes, Our method adds new mesh triangles derived from the input strokes to the original mesh. Denote the original mesh as $\mathscr{M}_0 = (\mathscr{V}_0, \mathscr{T}_0)$, as shown in gray in Fig. 8. We uniformly sample points from each user-specified input stroke, and construct a *sketch triangle* set $\mathscr{M}_s = (\mathscr{V}_s, \mathscr{T}_s)$ (purple triangles in Fig. 8) by connecting every three consecutive points on each stroke, excluding degenerate triangles on line segments. We then construct a *link triangle* set $\mathscr{T}_l$ by connecting each vertex in $\mathscr{V}_s$ with the three vertices of the triangle in $\mathscr{V}_0$ on which it falls.

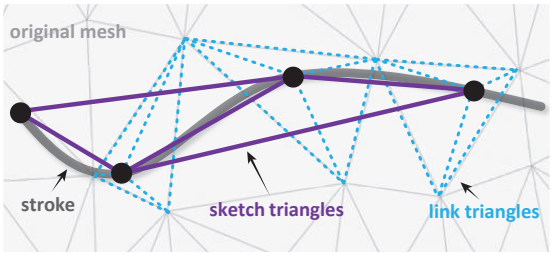

Figure 8. The modified mesh used for stroke-preserving mesh deformation.

Given the augmented mesh structure $\mathscr{M} = (\mathscr{V}_0 \cup \mathscr{V}_s, \mathscr{T}_0 \cup \mathscr{T}_s \cup \mathscr{T}_l)$, we formulate the stroke-preserving ARAP deformation as the following energy minimization problem:

$$\min_{\mathscr{M}} E_0 + \gamma_l E_{link} + \gamma_s E_{sketch}, \tag{3}$$

where E_0, E_{link} and E_{sketch} are the deformation energy terms of the three mesh triangles set $\mathscr{T}_0$, $\mathscr{T}_l$ and $\mathscr{T}_s$, respectively. Each energy term is defined as:

$$E(\mathscr{T}) = \sum_{t \in \mathscr{T}} \sum_{v_i, v_j \in t} \| \overrightarrow{v_i' v_j'} - H \overrightarrow{v_i v_j} \|^2, \tag{4}$$

which is based on [24]. It can also be formulated by other ARAP optimization methods, like [25]. H is a rigid and scale-irrelevant ARAP transformation matrix, which is achieved by a two-step optimization algorithm. The first step finds an appropriate rotation for each triangle and the second step adjusts its scale. Please refer to [24] for more details. Minimizing E_{sketch} keeps the shape of the strokes while minimizing E_{link} transfers the deformation of the original mesh to the strokes. In our system, we set the weight γ_l to a small value 0.01, allowing the triangles in the link triangle set to be distorted to balance between mesh deformation and stroke shape preserving. We set $\gamma_s = 1.0$ by default in order to better preserve the stroke shape. For all the strokes that need to be more rigid, the user can increase the corresponding γ_s value by a factor greater than 1.0 by moving a slider.

Multiple-Layer Animation

As discussed earlier, single layer, mesh-deformation-based animation cannot handle topology change, which is common for dynamic objects (see Fig. 9a). To handle such cases, our system allows the user to create animations with multiple layers. Using this tool, the user first divides the sketch strokes into several groups, each representing a layer (Fig. 9b). To avoid detaching different layers from each other during animation, our system then automatically detects the intersections of these layers, and adds one *joint point* at the center of each intersection region (Fig. 9c, blue circles) so that these layers are connected. Therefore, all layers can be deformed together during the motion transfer stage, as shown in Fig. 9d-e.

RESULTS

We have implemented our system in C++, and tested it on a wide variety of input sketch drawings and videos. For drawing aesthetics, we asked artists to create the sketches (all the examples in this paper except Fig. 13). Our main contribution is producing vivid animation of these static sketches. *Live Sketch* can handle complex non-rigid motions well, e.g., elastic motion (Fig. 10a), articulated motion (Fig. 10c) etc. By applying layered-animation, it can also produce high quality animation that involves self-occlusion (Fig. 10c). The motion extracted from one video example can be easily reused to drive different input sketches with different styles (Fig. 1; Fig. 10b; Fig. 11). Similarly, a single complex drawing can be animated by applying multiple motion examples to its different parts, as shown in Fig. 10d, where the final animation combines the extracted motion from the virus and kite videos. Please refer to the accompanying video for the resulting animations.

EVALUATIONS AND DISCUSSIONS

We conducted a pilot study to evaluate the efficiency and effectiveness of our system. Ideally we would like to compare our tool with existing animation tools. However, there is no existing tool that takes the same input as ours. As discussed previously, the closest work by Bregler et al. [10] requires a sequence of key shapes for interpolation, instead of a single sketch. Our tool would be faster as it does not need sketching additional key shapes. On the other hand, existing tools such as Adobe Flash and Toon Boom, which take more user inputs, might generate more detailed, expressive results, beyond the motion source. Thus such comparative experiments might not produce useful insights.

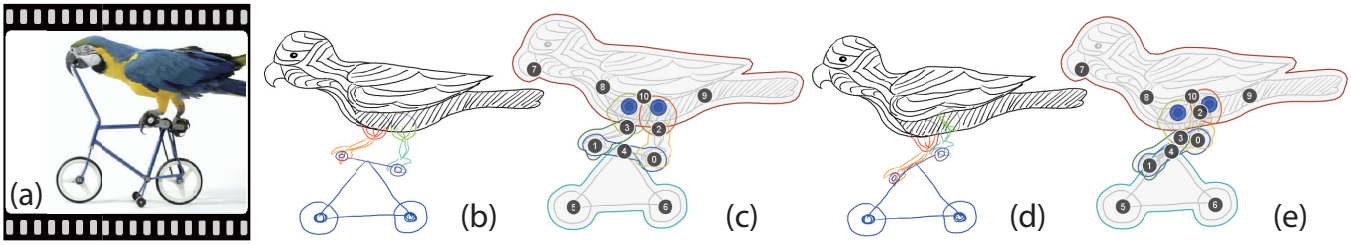

Figure 9. Multiple layer animation to handle self-occlusion and topology changes. When the two legs of the macaw cross, they introduce severe occlusion and topology change that cannot be handled by a single mesh layer. Our system allows the user to decompose the mesh into multiple layers (shown in different colors) to properly handle it. (a) Input video. (b) Grouping strokes into layers, visualized in different colors. (c) Control points and layer joints (blue circles). (d-e) Deformed sketch and new point layout in another frame. Video credit: Eric Isselee.

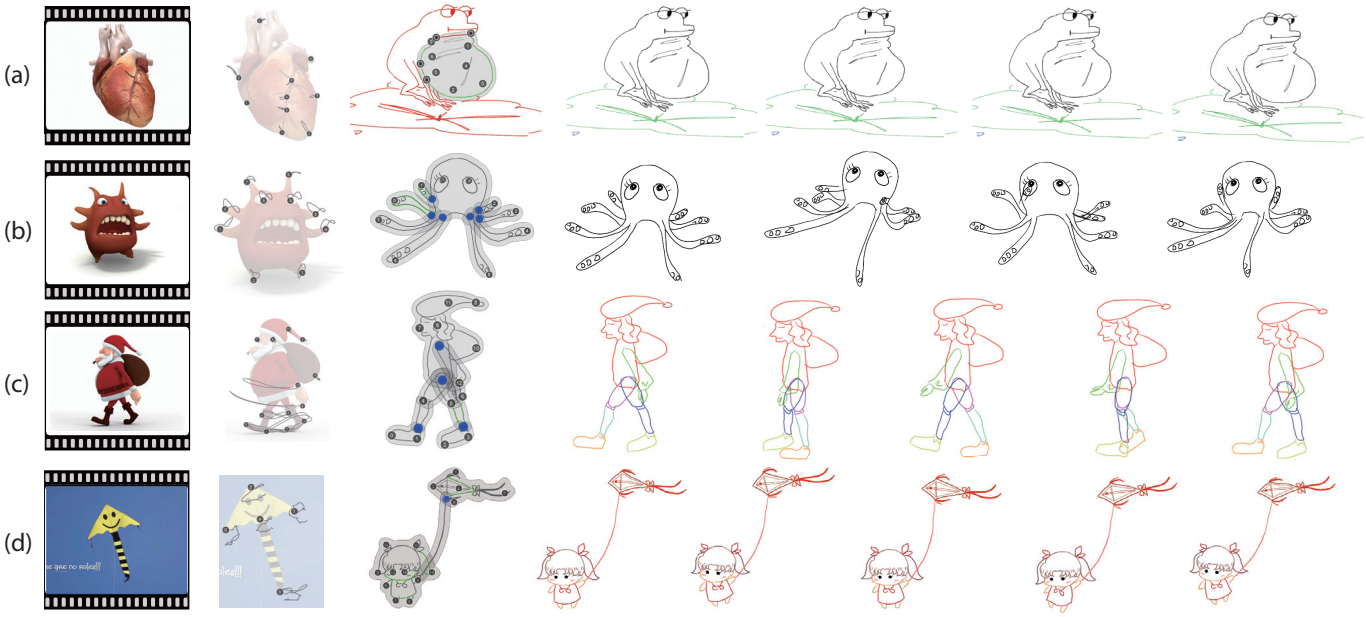

Figure 10. Some results produced using our system by artists. Column 1: the first frame of the video. Column 2: the extracted motion trajectories. Column 3: the corresponding animation control points on the sketches, with rigidity strokes in green and layer joints in blue. Other columns: several frames of the final animation. Video credits: Kjpargeter; Albert Ziganshin; brave rabbit; Jim Nicholls.

Instead of comparing with these existing tools, we were interested more in exploring whether our system can be used effectively by non-professional users. We also included users with adequate animation skills to see whether the performance of our tool depends on the complexity of the motion but not animation skills. In addition, we want to know if our system can fully support users' creativity to animate sketches by different users.

Participants and Apparatus. 12 university students (a1 to a12) were invited to participate in this study, including 6 male and 6 female. Half of the subjects (a1-a6) had good to professional drawing skills and 2D animation skills, while the other half (n1-n6) had no or very little animation experience. All subjects were enthusiastic about the idea of creating their own 2D animations. A touch-screen notebook (Surface Pro 4 with Intel(R) Core(TM) i5-6300U @2.40GHz 2.5GHz 8GB RAM) running Windows 10 was used as the testing device. Our system ran smoothly at an interactive rate on this device.

Design and Procedure. Each participant was first given a 20-minute training on how to use our system to draw sketches,

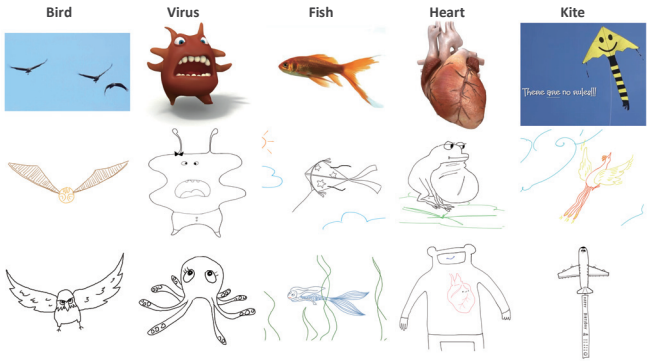

Figure 11. The trials used in our user study. Row 1: the input videos; Rows 2&3: the two sketches to be animated. Video credits: Sjoerd van der Wal; Albert Ziganshin; Artilopa ph; Kjpargeter; Jim Nicholls.

extract motion from a video and transfer motion to sketches, instructed by one of the authors. Our system provides only a basic sketching interface, since how to create good static drawings is not our focus. In this tutorial, one video was given for training and it was not used in the formal study.

The subjects were then given 10 minutes to practice using the same example. After the subjects were able to create animations for the sketches they drew and felt comfortable moving forward, they were asked to finish two tasks for each of 5 different video examples (namely, bird, virus, fish, heart, and kite, see Fig. 11 row 1): **Task 1:** animate two given sketches pre-drawn by artists (Fig. 11 row 2 and 3) and **Task 2:** draw two new sketches and animate them.

We used Task 1 to evaluate the usability of motion extraction and transfer, and Task 2 to evaluate the expressiveness of our tool. No time limit was set for these tasks, and one of the authors was always available to answer any questions the subjects might have. The subjects were asked to complete a questionnaire at the end of the study.

Figure 12. Selected sketches from Task 2, with each row corresponding to one artist (a1-a6). The columns correspond to the tasks in Fig. 11.

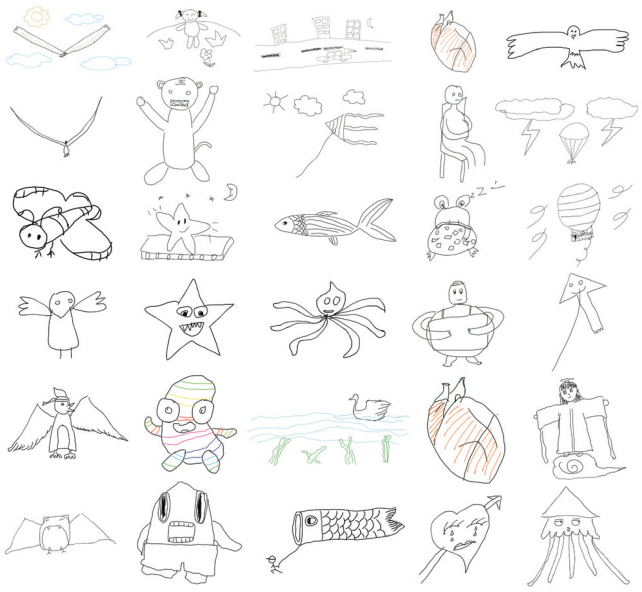

Figure 13. Selected sketches from Task 2, with each row corresponding to one novice (n1-n6). The columns correspond to the tasks in Fig. 11.

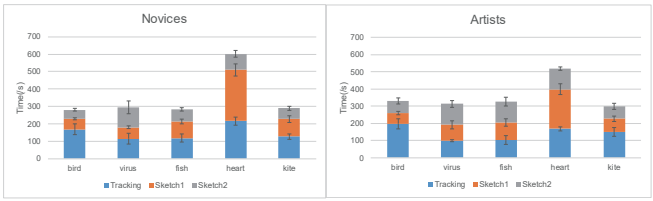

Figure 14. Finishing time of Task 1, for novice (left) and artist (right) users. Tracking: motion extraction time; Sketch1 and sketch2: motion transfer times for sketch1 and sketch2. There was no significant difference between novice and artist users.

Results. Our system recorded the timings of each subject, including the start and finish times of the sketching, the motion extraction and transfer steps. We also recorded all the user interactions: adding and adjusting tracking points; sketching; labeling decoration lines; applying local rigidity brushes, and adjusting motion scale and rotation, etc. The averaged completion time of Task 1 for novice and artist users is plotted in Fig. 14. The figure for the averaged number of manual operations is similar and thus not included in the paper. More detailed statistics can be found in Table 1. Some results of Task 2 are shown in Fig 12 and 13. Some of the animation results can be found in the supplementary video.

The results of Task 1 show that there was no significant difference between novice and artist users on the user time of motion extraction and transfer, when producing similar animation results. More specifically, on average, novice users and artists respectively spent 3.066 and 3.107 seconds per frame. Both groups of subjects spent the most time on the heart example, mainly because the specification of control points for motion extraction is less obvious for elastic motion of a single part. Artists applied slightly more manual operations (on average 0.655 operations per frame for artists and 0.510 operations per frame for novices). This is possibly because they preferred to have a finer control on the tracking and motion results to produce higher quality animation. Interestingly, their animation experience helped them operate faster, which is perhaps the reason why their finishing times were comparable to those of the novices.

Through the questionnaire, the participants indicated their satisfaction with our system. They commented that *Live Sketch* was easy to use, intuitive and efficient. More specifically,

n5: "Honestly, I have no experience in 2D sketch animation creation; this system provides a very intuitive method to transfer the realistic motion to the 2D sketch."

a2: "With this system, as we can just track an existing animation and apply it to other sketches, a simple animation can be easily created. With existing tools, we would need to create a whole new animation from scratch by ourselves."

Eleven out of twelve participants mentioned that the motion extraction tool was easy to learn and use. There were also ten users who liked the fact that they did not need to control the animation frame-by-frame. In addition, our statistics for Task 2 shows that the motion extraction and transfer parts took very short time compared with the sketching time. The respective average times for sketching, motion extraction and motion transfer were 306.80s, 149.86s, 108.62s for novice

Table 1. Statistics of Task1. "tracking", "sketch1", "sketch2" refer to the average tracking time and animation time for each input sketch in seconds, respectively. "#ctrl pts" and "#adjust." refer to the respective numbers of control points and the manual adjustments in tracking each video.

		Novices					Artists				
video	#frame	#ctrl pts	#adjust.	tracking(s)	sketch1(s)	sketch2(s)	#ctrl pts	#adjust.	tracking (s)	sketch1(s)	sketch2(s)
bird	45	4.2	9.05	169.60	59.19	52.06	4.7	18.1	198.65	62.11	69.89
virus	18	8.3	8.03	115.08	65.71	114.15	9.5	7.5	100.13	92.02	121.39
fish	118	6.7	4.13	118.76	94.94	69.52	8	4.33	103.32	101.29	121.17
heart	31	10	12	217.58	291.90	91.66	8	9.67	168.80	229.32	122.10
kite	129	5	9.33	128.27	99.90	61.72	4.8	14.53	150.23	77.24	71.85

users and 450.36s, 144.23s and 95.90s for artists. This means once motion is prepared, both artists and novices can animate a sketch quickly.

Motion extraction. The user interface of motion extraction impressed some users. They commented they could extract satisfactory motion with a small amount of user intervention. The frequency of tracking refinement operation is very low, more specifically, on average one operation for every 34 frames per control point.

n2: "I like the idea of transferring the motion from a tracked object to the drawn sketch, which makes the animation quite efficient. . . . Though the tracking results are not stable sometime, users can choose tools to manually update the control points, which for sure is a nice add-on."

a5: "I like the feature of motion capture which can save the time of animators. For 2D-hand drawing animators, they have to draw frame by frame (12-24 frames per second) in order to create a smooth animation. However, by using this system, animators just need to draw one sketch to create a short animation."

Motion transfer. Artists a1, a2, and a5 and the novices n1, and n3-5 also gave positive feedback on the motion transfer part. The stroke-preserving constraint and motion adjustment tool simplify the whole animation process, as confirmed by the participants.

n3: "The system also provides some useful tools that allow adding anchors, fixing background and adding constraints. . . . With them I can further improve the quality and customize my animations with several interactions."

Limitations. Some artists mentioned that the transferred animation lacks fine details of the video. This is mainly because the animation is driven by sparsely distributed control points. One possible solution is using dense tracking methods to extract the motion. However, existing dense tracking methods usually do not provide a good scheme to correct tracking errors. We will explore interactive dense tracking correction and precise dense points mapping methods in the future to achieve results with richer details (Fig. 15 (left)). Furthermore, because our system requires only a single sketch as input, it is difficult to introduce 3D animation by simply deforming a single sketch in 3D, though our motion extraction method can capture 3D motion (Fig. 15 (right)).

CONCLUSION AND FUTURE WORK

We have presented a new interactive system for animating sketch drawings using video examples. The key idea is to extract object motion presented in the video using sparse point tracking, and transfer it to the sketch image using controlled

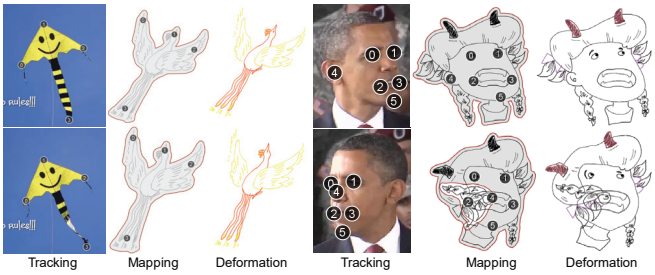

Figure 15. Failure cases of our system. Left: fine motion details of the tail are missing because of our sparse point tracking method; right: large sketch distortion after transferring 3D motion. Video credits: Jim Nicholls; Rick Ray.

mesh deformation. For motion extraction, we propose a new tracking method that is robust against occlusion and ambiguity, and further combines it with easy user controls for reliable tracking. For motion transfer, our system allows the user to fine control the animation using a set of interactive tools. We conducted a pilot study to show that given static sketches and the corresponding videos, non-professional users can turn them into vivid animations using the amount of time and efforts similar to that regard by the professional users.

We believe our current work has only scratched the surface of an exciting opportunity. As pointed out earlier, our system still has problems in handling smooth image regions such as deforming clouds, or stochastic motions such as ocean waves, whose motion is hard to extract using automatic or interactive ways. Expanding the system to handle more types of motions is desirable. The system currently can only transfer the raw motion extracted from videos, it would be interesting to use existing motion stylization or exaggeration approaches, such as the Animation Filter [45], to produce more lively motion trajectories by applying a simple filter. More stylizing animations may also be made by interactively applying Motion Amplifiers [28], Elemental Dynamics [50] to our animation results. In the future we would like to address the first subproblem as discussed in the Introduction section by building a large set of videos with different motion properties and developing a retrieval method to find proper video examples given an arbitrary sketch as input.

ACKNOWLEDGMENTS

We thank Lei Li for video making, Yilan Chen for preparing some of the examples, the reviewers for their constructive comments, the user study participants for their time and the ShutterStock users for providing their videos. This work was partially supported by grants from ACIM-SCM and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKUST 16201315, CityU 11237116 and CityU 11204014).

REFERENCES

1. Aseem Agarwala, Aaron Hertzmann, David H. Salesin, and Steven M. Seitz. 2004. Keyframe-based Tracking for Rotoscoping and Animation. *ACM Trans. Graph.* 23, 3 (2004), 584–591.
2. B. Amberg and T. Vetter. 2011. GraphTrack: Fast and globally optimal tracking in videos. In *CVPR 2011*. 1209–1216.
3. M. Arie-Nachimson, S. Z. Kovalsky, I. Kemelmacher-Shlizerman, A. Singer, and R. Basri. 2012. Global Motion Estimation from Point Matches. In *3DIMPVT '12*. 81–88.
4. Rahul Arora, Darolia Ishan, Vany P. Namboodiri, Karan Singh, and Adrien Bousseau. 2017. SketchSoup: Exploratory Ideation using Design Sketches. *Computer Graphics Forum* (2017).
5. Nicole M. Artner, Adrian Ion, and Walter G. Kropatsch. 2011. Multi-scale 2D tracking of articulated objects using hierarchical spring systems. *Pattern Recognition* 44, 9 (2011), 1969 – 1979.
6. Connelly Barnes, David E. Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. 2008. Video Puppetry: A Performative Interface for Cutout Animation. In *SIGGRAPH Asia '08*. ACM, Article 124, 9 pages.
7. Serge Belongie, Jitendra Malik, and Jan Puzicha. 2000. Shape context: A new descriptor for shape matching and object recognition. In *Nips*, Vol. 2.
8. Nir Ben-Zvi, Jose Bento, Moshe Mahler, Jessica Hodgins, and Ariel Shamir. 2015. Line-Drawing Video Stylization. In *Computer Graphics Forum*, Vol. 35. 18–32.
9. Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. 2013. Stylizing Animation by Example. *ACM Trans. Graph.* 32, 4, Article 119 (2013), 12 pages.
10. Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. 2002. Turning to the Masters: Motion Capturing Cartoons. In *SIGGRAPH '02*. 399–407.
11. A. Buchanan and A. Fitzgibbon. 2006. Interactive Feature Tracking using K-D Trees and Dynamic Programming. In *CVPR'06*, Vol. 1. 626–633.
12. Z. Cai, L. Wen, Z. Lei, N. Vasconcelos, and S. Z. Li. 2014. Robust Deformable and Occluded Object Tracking With Dynamic Graph. *IEEE Transactions on Image Processing* 23, 12 (2014), 5497–5509.
13. Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 2013. 3D Shape Regression for Real-time Facial Animation. *ACM Trans. Graph.* 32, 4, Article 41 (2013), 10 pages.
14. L. Cehovin, M. Kristan, and A. Leonardis. 2013. Robust Visual Tracking Using an Adaptive Coupled-Layer Visual Model. *TPAMI* 35, 4 (2013), 941–953.
15. Jiawen Chen, Shahram Izadi, and Andrew Fitzgibbon. 2012. KinÊTre: Animating the World with the Human Body. In *UIST '12*. ACM, 435–444.
16. Theodoros Chondrogiannis, Panagiotis Bouros, Johann Gamper, and Ulf Leser. 2015. Alternative routing: k-shortest paths with limited overlap. In *SIGSPATIAL GIS '15*. ACM, 68.
17. James Davis, Maneesh Agrawala, Erika Chuang, Zoran Popović, and David Salesin. 2003. A Sketching Interface for Articulated Figure Animation. In *SCA '03*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 320–328.
18. Julien Diener, Lionel Reveret, and Eugene Fiume. 2006. Hierarchical retargetting of 2D motion fields to the animation of 3D plant models. In *SCA'06*.
19. Yotam Gingold, Etienne Vouga, Eitan Grinspun, and Haym Hirsh. 2012. Diamonds From the Rough: Improving Drawing, Painting, and Singing via Crowdsourcing. In *HCOMP*.
20. Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. 2015. Space-time Sketching of Character Animation. *ACM Trans. Graph.* 34, 4, Article 118 (2015), 10 pages.
21. S. Hare, A. Saffari, and P. H. S. Torr. 2011. Struck: Structured output tracking with kernels. In *ICCV '11*. 263–270.
22. Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 2012. 3D Puppetry: A Kinect-based Interface for 3D Animation. In *UIST '12*. ACM, 423–434.
23. Alexander Hornung, Ellen Dekkers, and Leif Kobbelt. 2007. Character Animation from 2D Pictures and 3D Motion Data. *ACM Trans. Graph.* 26, 1, Article 1 (Jan. 2007).
24. Takeo Igarashi, Tomer Moscovich, and John F. Hughes. 2005. As-rigid-as-possible Shape Manipulation. In *ACM Trans. Graph.*, Vol. 24. 1134–1141.
25. Alec Jacobson, Ilya Baran, Jovan Popović, and Olga Sorkine. 2011. Bounded Biharmonic Weights for Real-time Deformation. In *SIGGRAPH '11*. ACM, Article 78, 8 pages.
26. Michael Kass, Andrew Witkin, and Demetri Terzopoulos. 1988. Snakes: Active contour models. *IJCV* 1, 4 (1988), 321–331.
27. Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: Bringing Life to Illustrations with Kinetic Textures. In *CHI '14*. ACM, 351–360.
28. Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. Motion Amplifiers: Sketching Dynamic Illustrations Using the Principles of 2D Animation. In *CHI '16*. ACM, 4599–4609.

29. Midori Kitagawa and Brian Windsor. 2008. *MoCap for Artists: Workflow and Techniques for Motion Capture*. Focal Press.

30. Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. 2013. State of the Art: A Taxonomy of Artistic Stylization Techniques for Images and Video. *TVCG* 19, 5 (2013), 866–885.

31. Yong Jae Lee, C. Lawrence Zitnick, and Michael F. Cohen. 2011. ShadowDraw: Real-time User Guidance for Freehand Drawing. In *ACM Trans. Graph.*, Vol. 30. Article 27, 10 pages.

32. Guanbin Li, Sai Bi, Jue Wang, Yingqing Xu, and Yizhou Yu. 2017. ColorSketch: A Drawing Assistant for Generating Color Sketches from Photos. *IEEE Computer Graphics and Applications* 38, 3 (2017), 70–81.

33. Brais Martinez and Xavier Binefa. 2008. Piecewise affine kernel tracking for non-planar targets. *Pattern Recognition* 41, 12 (2008), 3682 – 3691.

34. Min Je Park, Min Gyu Choi, Yoshihisa Shinagawa, and Sung Yong Shin. 2006. Video-guided Motion Synthesis Using Example Motions. *ACM Trans. Graph.* 25, 4 (2006), 1327–1359.

35. Bernhard Reinert, Tobias Ritschel, and Hans-Peter Seidel. 2016a. Animated 3D Creatures from Single-view Video by Skeletal Sketching. In *GI '16*. 133–141.

36. Bernhard Reinert, Tobias Ritschel, and Hans-Peter Seidel. 2016b. Animated 3D Creatures from Single-view Video by Skeletal Sketching.. In *Graphics Interface*. 133–141.

37. Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 309–314.

38. Stephanie Santosa, Fanny Chevalier, Ravin Balakrishnan, and Karan Singh. 2013. Direct Space-time Trajectory Control for Visual Media Editing. In *CHI '13*. ACM, 1149–1158.

39. Steven M. Seitz and Charles R. Dyer. 1996. View Morphing. In *SIGGRAPH '96*. ACM, 21–30.

40. Jianbo Shi and Carlo Tomasi. 1994. Good Features to Track. In *CVPR'94*. 593 – 600.

41. Eisung Sohn and Yoon-Chul Choy. 2012. Sketch-n-Stretch: Sketching Animations Using Cutouts. *IEEE Comput. Graph. Appl.* 32, 3 (2012), 59–69.

42. Qingkun Su, Wing Ho Andy Li, Jue Wang, and Hongbo Fu. 2014. EZ-Sketching: Three-Level Optimization for Error-Tolerant Image Tracing. *ACM Trans. Graph.* 33, 4 (2014), 54:1–54:9.

43. J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. 2016. Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In *CVPR'16*.

44. Matthew Thorne, David Burke, and Michiel van de Panne. 2004. Motion Doodles: An Interface for Sketching Character Motion. In *SIGGRAPH '04*. ACM, 424–431.

45. Jue Wang, Steven M. Drucker, Maneesh Agrawala, and Michael F. Cohen. 2006. The Cartoon Animation Filter. In *SIGGRAPH '06*. ACM, 1169–1173.

46. Jue Wang, Yingqing Xu, Heung-Yeung Shum, and Michael F. Cohen. 2004. Video Tooning. In *ACM Trans. Graph.*, Vol. 23. 574–583.

47. Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional Pose Machines. In *CVPR'16*.

48. Nora S. Willett, Wilmot Li, Jovan Popovic, and Adam Finkelstein. 2017. Triggering Artwork Swaps for Live Animation. In *UIST '17*. ACM, 85–95.

49. Jun Xing, Hsiang-Ting Chen, and Li-Yi Wei. 2014. Autocomplete Painting Repetitions. *ACM Trans. Graph.* 33, 6, Article 172 (2014), 11 pages.

50. Jun Xing, Rubaiat Habib Kazi, Tovi Grossman, Li-Yi Wei, Jos Stam, and George Fitzmaurice. 2016. Energy-Brushes: Interactive Tools for Illustrating Stylized Elemental Dynamics. In *UIST '16*. 755–766.

51. Jun Xing, Li-Yi Wei, Takaaki Shiratori, and Koji Yatani. 2015. Autocomplete Hand-drawn Animations. *ACM Trans. Graph.* 34, 6, Article 169 (2015), 11 pages.

52. Katsu Yamane, Yuka Ariki, and Jessica Hodgins. 2010. Animating non-humanoid characters with human motion data. In *SCA '10*. 169–178.

53. Alper Yilmaz, Omar Javed, and Mubarak Shah. 2006. Object tracking: A survey. *Acm computing surveys (CSUR)* 38, 4 (2006), 13.

54. Tong Zhang and C. Tomasi. 1999. Fast, robust, and consistent camera motion estimation. In *CVPR '99*, Vol. 1. 170 Vol. 1.

55. Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. 2005. Large Mesh Deformation Using the Volumetric Graph Laplacian. *ACM Trans. Graph.* 24, 3 (2005), 496–503.