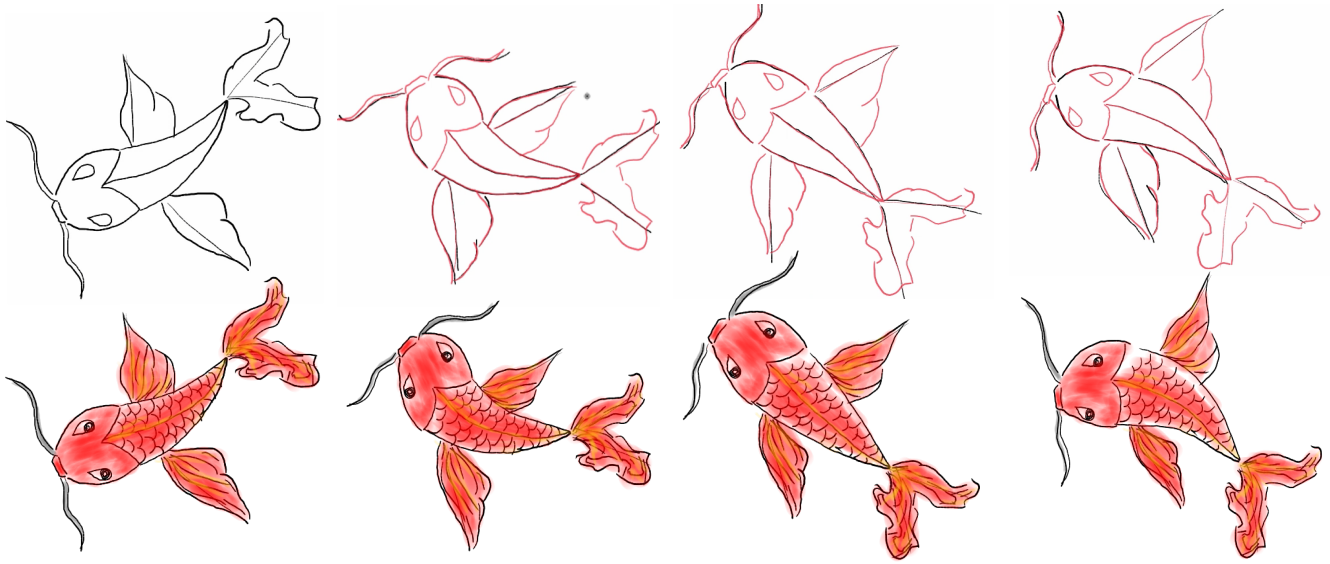
Autocomplete Hand-drawn Animations

Jun Xing[†§] Li-Yi Wei[†] Takaaki Shiratori[§] Koji Yatani[‡]

University of Hong Kong[†] Microsoft Research[§] University of Tokyo[‡]



(a) *frame 0* **(b)** *frame 1* **(c)** *frame 2* **(d)** *frame 3*

Figure 1: *Autocomplete hand-drawn animation example. Top: after outlining a fish in the first frame (a), the user starts to draw the subsequent frames. Our system can predict what the user might want to draw next and how to improve what has been drawn already, as indicated by the red strokes in (b), (c), and (d). Bottom: after finishing the outlines in all frames, the user adds details to the first frame (a), while our system automatically propagates and adapts the changes to all other frames. Please refer to the accompanying video for live actions and animations.*

Abstract

Hand-drawn animation is a major art form and communication medium, but can be challenging to produce. We present a system to help people create frame-by-frame animations through manual sketches. We design our interface to be minimalistic: it contains only a canvas and a few controls. When users draw on the canvas, our system silently analyzes all past sketches and predicts what might be drawn in the future across spatial locations and temporal frames. The interface also offers suggestions to beautify existing drawings. Our system can reduce manual workload and improve output quality without compromising natural drawing flow and control: users can accept, ignore, or modify such predictions visualized on the canvas by simple gestures. Our key idea is to extend the local similarity method in [Xing et al. 2014], which handles only low-level spatial repetitions such as hatches within a single frame, to a global similarity that can capture high-level structures across multiple frames such as dynamic objects. We evaluate our system through a preliminary user study and confirm that it can enhance both users' objective performance and subjective satisfaction.

ACM Reference Format
Xing, J., Wei, L., Shiratori, T., Yatani, K. 2015. Autocomplete Hand-drawn Animations.
ACM Trans. Graph. 34, 6, Article 169 (November 2015), 11 pages. DOI = 10.1145/2816795.2818079
http://doi.acm.org/10.1145/2816795.2818079.

ACM 978-1-4503-3931-5/15/11 ... $15.00.
DOI: http://doi.acm.org/10.1145/2816795.2818079

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; H.5.1 [Information Interfaces and Presentation (e.g., HCI)]: Multimedia Information Systems—Animations;

Keywords: animation, drawing, auto-complete, analysis, synthesis, beautification, workflow, user interface, deformation

1 Introduction

Hand-drawn animation is a popular art form and communication medium. However, it is challenging to produce, even for experienced professionals. In addition to appropriate spatial arrangement in one frame, users also need to maintain a consistent temporal flow across multiple frames. Existing computer graphics methods can ameliorate this difficulty to some extent by reusing underlying art content, such as deforming shape templates [Igarashi et al. 2005] or cloning animated texture sprites [Kazi et al. 2014a]. However, manual drawing can provide unique freedom of expression and more natural touch for many artists. Thus, there is a pressing need of interactive tools that can support creation of manual animation more effectively while maintaining artists' natural drawing practices. This demand is not only for professionals due to the rise of mobile and social applications for authoring and sharing hand-drawn animations, which are used by many amateur artists.

We present an interactive drawing system that helps users produce animation more easily and in a better quality while preserving manual drawing practices. Users simply create a series of drawings in our system as they would do in their common tools. Our system records and analyzes their past drawings in the background, and provides suggestions that can save manual labor and improve draw-

ing quality. As illustrated in Figure 1, the system can detect potential repetitions such as static objects and dynamic motions, and predict what might need to be drawn across spatial locations and temporal frames. Users can accept, ignore, or modify such suggestions analogous to the mechanisms for auto-complete spatial repetitions for single static sketches in [Xing et al. 2014]. Later, the users may want to modify existing animations, such as changing the shapes or colors of certain objects. They can simply make desired visual changes in one frame, and our system will propagate these changes across similar spatial objects at all temporal frames to reduce tedious manual repetitions. Unlike many existing deformation methods, our system allows users to create animations with topological changes, such as breaking objects or growing plants.

The key idea of this work is to extend the analysis and synthesis algorithms of drawing repetitions within individual frames [Xing et al. 2014] to multiple frames of animations. The method in [Xing et al. 2014] is based on *local similarity*, which can predict only low-level spatial repetitions such as hatches, but not high-level structures such as complex objects in animations. To overcome this challenge, we extend their local similarity to a *global similarity* that can capture both global contexts (e.g., object contours) and local details (e.g., individual strokes). We then measure the global similarity between past and current drawings to predict what the users might want to draw in the future. The prediction is not 100% accurate, but is robust enough for common variations (including number, shape, and order of strokes) as long as the drawing is reasonably coherent across frames, such as when people follow the well-known blocking technique [Iarussi et al. 2013]. Our preliminary user study confirmed that participants were able to produce a wide variety of animations in our system with a reduced number of strokes and in good quality, and expressed positive experiences.

2 Previous Work

Digital painting support A variety of methods exist to support digital painting, often based on procedural simulation [Chu and Tai 2005; Lindemeier et al. 2015] and data analysis [Lee et al. 2011; Lu et al. 2013; Lukáč et al. 2013]. However, they mostly focus on producing single static images. Animation sequences involve more manual work to create. There are also challenging issues to automate animation creation process, such as maintaining temporal coherence [Bénard et al. 2013]. Our work follows a similar line of philosophy — facilitating digital drawing through interactive systems, but emphasizes animation.

Animation authoring by sketch Even though powerful tools exist for creating 3D animations, some of which also offer intuitive sketch-based interfaces [Davis et al. 2003; Thorne et al. 2004; Milliez et al. 2014; Guay et al. 2015], drawing 2D animations still remains as a common activity for both professionals and amateurs. Some existing techniques, such as deforming shapes [Igarashi et al. 2005; Sýkora et al. 2009] and interpolating instances [Baxter and Anjyo 2006; Whited et al. 2010], might not satisfy user's desires and needs because the former cannot handle topological changes, such as a breaking objects or growing plants, and the latter often require manual specification or refinement of point correspondences. Cloning texture sprites [Kazi et al. 2014b] and combinations thereof [Jones et al. 2015] can quickly add animated effects to existing drawings but not author more general animations.

Enhancement by workflow analysis Workflow recording and analysis have been extensively investigated recently to support a variety of content creation tasks, as surveyed in [Nancel and Cockburn 2014; Chen et al. 2014; Myers et al. 2015]. Examples include handwriting beautification [Zitnick 2013], automatic creation of photo editing tutorials [Grabler et al. 2009], and auto-complete static drawing [Xing et al. 2014]. However, dynamic animations have received much less attention though workflow analysis has a potential to improve their creation process. Our work addresses this issue by incorporating workflow analysis into 2D hand drawings, and predicts what users may want to draw in animation frames.

Figure 2: *The user interface of our system. Our interface consists of a main canvas (center), a widgets pane (left), and a time frame slider (bottom). Users draw on the main canvas with tools from the widget pane, and drag the time frame slider to select existing frames or add new frames. The main canvas displays the current frame and few preceding frames in lighter colors for guidance. All these are similar to standard animation sketching tools. However, our system predicts what the users might want to draw next based on previous frames. The prediction is presented underneath the canvas as a hint (red strokes), which users can accept via a simple gesture or brush-select specific regions (shown in blue).*

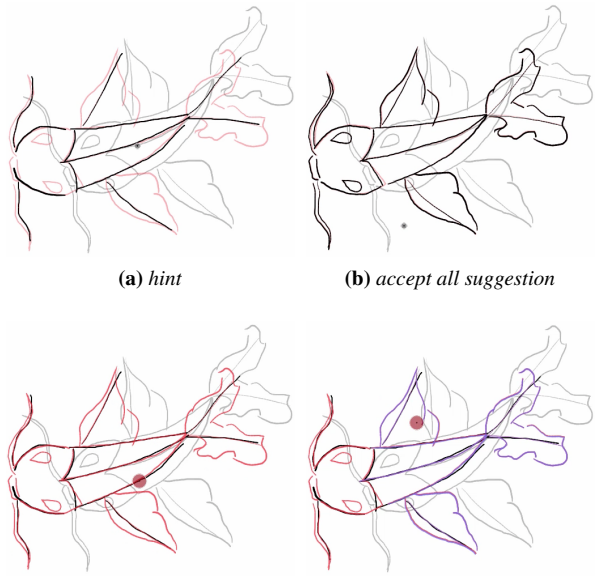
3 User Interface

As shown in Figure 2, our user interface is designed to have similar visual appearance to standard keyframe animation authoring tools. Users can sketch each frame in the main canvas, and play the frames as animation. Meanwhile our system automatically records and analyzes their sketches in the background. It then predicts what users might want to draw next, and visualizes the predictions directly on the canvas, which the users can accept, ignore, or modify, analogous to the spatial hints in [Lee et al. 2011; Xing et al. 2014]. These features are designed to reduce manual workload and improve output quality while maintaining the natural manual sketch practice.

Figure 3 shows an example of sketching a new frame. As the user sketches, our system provides hints based on the preceding frames and the current drawing (Figure 3a). The user can accept them via a simple gesture (Figure 3b), ignore it by simply continuing to draw, or press and hold on the canvas to enable the selection brush on parts of the hints (Figures 3c and 3d).

The interface displays hints whenever any spatial or temporal repetition is detected (Figure 4). These hints are predicted from motion and shape in previous frames. As the user adds more strokes, the system updates the hints accordingly. Our system supports both temporal repetition with large-scale structures (Figures 4a and 4b) and spatial repetition of detail-level strokes [Xing et al. 2014] (Figures 4c and 4d). It also provides an initial hint when the user starts a new empty frame for bootstrapping (Figure 4a).

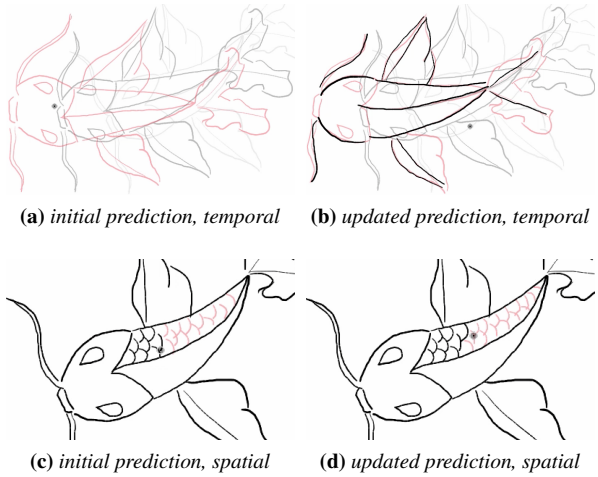
Beautification Using the same underlying stroke analysis, our system can improve what has already been drawn as well as predict what should be drawn next. Figure 5 illustrates our beautification capability. Different from the hand writing/drawing beautification in [Zitnick 2013] that computes an *average* object based on past similar objects, our system deforms the previous frame to ob-



(a) *hint* **(b)** *accept all suggestion*

(c) *enter selection* **(d)** *brush select*

Figure 3: *Hint. While the user sketches the fish in a new frame, our system shows both the previous frame(s) (light gray) and the hint for current frame (light red) in the canvas (a). The user can ignore the hint by continuing drawing, accept all hints via a simple gesture (b), or activate the brush-select function (c) to select parts of the hint (d) (brush shown as a circle, hint highlighted with red color, and selected strokes rendered in blue color).*



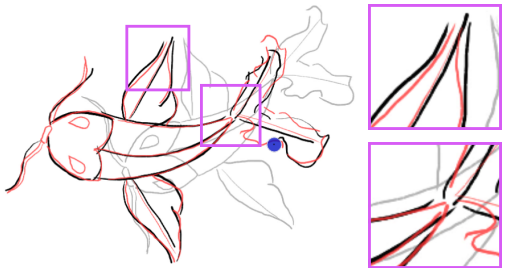
(a) *initial prediction, temporal* **(b)** *updated prediction, temporal*

(c) *initial prediction, spatial* **(d)** *updated prediction, spatial*

Figure 4: *Online prediction update. When the user creates a new empty frame (a), the system automatically provides an initial prediction based on the preceding frames, visualized in red color. The user can ignore the prediction and continue drawing (black strokes), and the prediction will be updated accordingly in real-time (b). Our system supports both temporal and spatial repetition, such as motion deformation in (a) and (b) and fish scales in (c) and (d).*

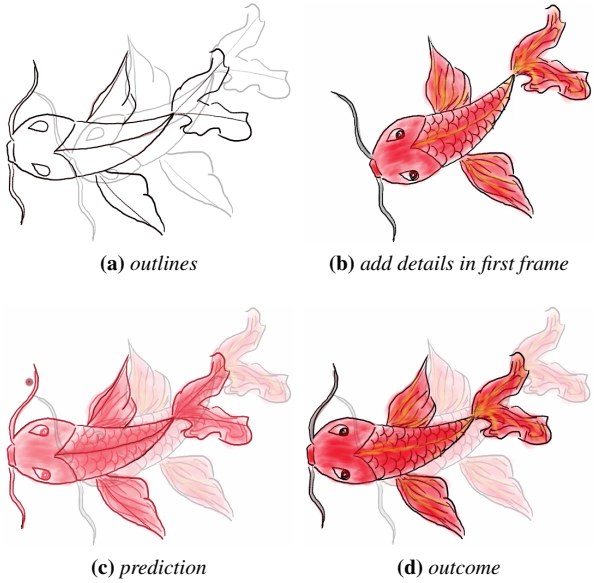
tain the suggestions, which are in general more temporally coherent than fresh drawings [Noris et al. 2011].

Edit propagation A common drawing practice is to create a rough sketch first and then add details. However, this can cause an issue in animation authoring because users have to make modifications across frames. Our prediction mechanism can handle such cross-frame editing. As an example in Figure 6, users first draw



(a) *beautification* **(b)** *zoom in*

Figure 5: *Beautification. As the user draws (shown in black), our system can suggest beautifying existing drawings (shown in red) based on both spatial and temporal information, e.g. what has been drawn on the same frame and previous frames. Users can accept, ignore, or brush-select the suggestions similar to Figure 3.*



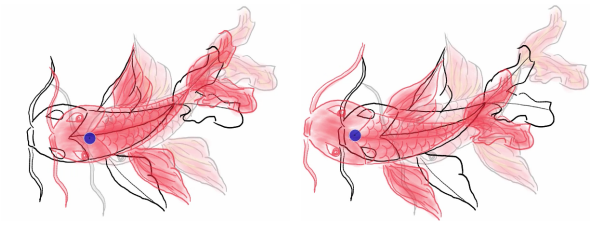
(a) *outlines* **(b)** *add details in first frame*

(c) *prediction* **(d)** *outcome*

Figure 6: *Edit propagation. After drawing outlines of all frames (a) as shown in Figures 3 and 4, the user starts to fill in interior details in the first frame (b). When the user switches to the next frames, the details are automatically propagated as shown in red in (c). Similar to Figure 3, the user can accept, ignore, or brush select the hints (d).*

the outlines of the fish in each frame and decide to add details later. They only need to do so in one frame (Figure 6b), and our system will automatically propagate and adapt the changes to all other frames (Figure 6c). Again, users are free to accept, ignore, or modify these suggestions.

Timeline preview The hint quality depends on not only drawings at individual frames but also the relative motions across nearby frames. We thus allow users to preview the hint by manually adjusting its timestamp around the current frame. As shown in Figure 7, users can press-and-hold and then drag the hint directly over the main canvas to adjust its time stamp along the motion trajectory. For more intuitive control, our system maps the timestamp of the hint to the spatial canvas position based on its motion calculated from Section 4.3, so that users can directly interact with the spatial hint (e.g. dragging in the same direction with the motion will increase the timestamp) instead of a traditional linear timeline.



(a) *backward* **(b)** *forward*

Figure 7: *Timeline preview. The user can directly drag the hints to adjust its time stamp, such as backward to set an early time (a) or forward for a later time (b), while seeing the prediction automatically adjusted.*

Mode selection As described above, our system includes two modes: painting and selection. Users can press-and-hold on the main canvas to activate the selection mode, and double-click on the main canvas to switch back to the painting mode. Furthermore, while in the selection mode, users can perform another press-and-hold to enable timeline preview, and release the contact to go back to the selection mode. Users can play the whole animation by clicking the play button located on the left-bottom corner of the interface. Our system provides a smooth animation by automatically adding in-betweens.

4 Method

We follow the methodology in [Xing et al. 2014] to analyze repetitions in the recorded drawing workflow, and use those repetitions to synthesize suggestions based on the current drawing. Xing et al. [2014] is based on a *local similarity* analysis [Ma et al. 2011], which is limited to small-scale spatial repetitions such as hatches. To handle animations with large/complex objects, we extend their local similarity to a *global similarity*, which can capture large scale structures (e.g. tree branches) and adapt to local variations (e.g. different branch shapes). We then incorporate this global similarity into the registration methods in [Sumner et al. 2007; Li et al. 2008] to deform drawings from the previous frame towards the current frame for predicting what the users might want to draw next. The prediction preserves the spatial shape and temporal coherence, and adapts to the current user drawings. Our global similarity establishes a form of fuzzy correspondence [Kim et al. 2012] and can handle ambiguous situations common in manual drawings, such as different number, shape, and order of strokes for the same object across frames for which traditional registration methods may have trouble with. To facilitate interactive performance, our global similarity analysis can be computed incrementally while incorporating incoming user edits.

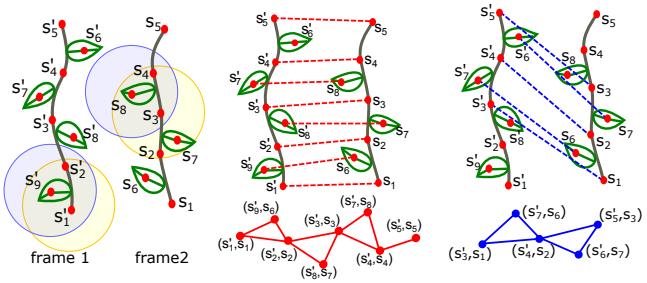
We describe the basic representation of drawing workflows in Section 4.1, analysis of global similarity in Section 4.2, and synthesis of predictions in Section 4.3. We illustrate our method with a concrete case in Figure 8. Since our method follows [Xing et al. 2014] for computing small-scale spatial repetitions, this paper mainly explains our algorithm on the analysis and synthesis of large-scale temporal repetitions.

4.1 Representation

Sample As illustrated in Figure 9, we represent animation sketches via samples. We sample each stroke uniformly, and represent each sample s as

$$\mathbf{u}(s) = (\mathbf{p}(s), \mathbf{m}(s), \mathbf{a}(s), \mathbf{t}(s)), \tag{1}$$

which includes spatial parameters $\mathbf{p}(s)$ that record the position at s; motion parameters $\mathbf{m}(s)$ that capture the local transformation



(a) *external UI* **(b)** *red graph* **(c)** *blue graph*

Figure 8: *Algorithm overview via a simple example. On the UI (a), the user first draws frame* 1 *(left) followed by frame* 2 *(right). The red dots indicate samples (Section 4.1), and the subscripts indicate their drawing order. Note that the two frames can have different numbers, shapes, and orders of strokes. Within our method, as the user draws a new sample* s_8 *in frame* 2*, our system records it in the workflow, and extracts its neighborhood (blue circle, Section 4.1). Our system then analyzes the likelihood of* s_8 *corresponding to each sample* s' *in frame* 1 *(Section 4.2) by considering neighborhood similarity [Ma et al. 2011; Xing et al. 2014] of not only* s_8 *itself (blue circle) but also its neighbors such as* s_3 *(yellow circle) to capture both local and global structures. For example, although* s'_9 *and* s_8 *have a high* local *similarity (blue circles), their neighborhood samples* s'_1 *and* s_3 *don't match well (yellow circles), indicating* s'_9 *matches to* s_8 *only locally but not globally. We mark those sample pairs with high* global *similarity as matching-pairs, and dynamically group each new matching-pair (e.g.* (s'_7, s_8)*) with existing matching-pairs in its neighborhood (e.g.* (s'_3, s_3)*) into graphs (bottom (b) and (c)). Each graph represents a group of consistent correspondences [Huang et al. 2008] (top (b) and (c)), and larger graphs (e.g. red) usually indicate better correspondences than smaller ones (e.g. blue). Based on the constructed graphs, we can further refine the global similarities, as they are usually ambiguous in the beginning of drawing. For example, when only* $s_{1,2,3}$ *are drawn,* s_3 *could match* $s'_{3,4,5}$*. As more samples are drawn, the largest graph (red) indicates* s'_3 *as the most likely match to* s_3*. By using the refined global similarity as a kind of fuzzy correspondence between the two drawing frames, we can deform frame* 1 *toward frame* 2*, which will be presented to the user as suggestions (Section 4.3). The suggestion is updated each time the user draws a new stroke, and such frequency can be manually adjusted.*

at s across frames including translation, rotation, and scaling; appearance parameters $\mathbf{a}(s)$ including color, pressure and size; and temporal parameters $\mathbf{t}(s)$ that include the global time stamp and a sample-id for the relative position within a stroke.

We measure the difference between two samples s_j and s_i as

$$\widehat{\mathbf{u}}(s_j, s_i) = \left(\widehat{\mathbf{p}}(s_j, s_i), \alpha\widehat{\mathbf{m}}(s_j, s_i), \beta\widehat{\mathbf{a}}(s_j, s_i), \gamma\widehat{\mathbf{t}}(s_j, s_i)\right), \tag{2}$$

where $\widehat{\mathbf{p}}$, $\widehat{\mathbf{m}}$, $\widehat{\mathbf{a}}$ and $\widehat{\mathbf{t}}$ represent the difference in position $\mathbf{p}$, transformation $\mathbf{m}$, appearance $\mathbf{a}$, and time-stamp $\mathbf{t}$, respectively, and are all computed with respect to s_i, i.e. $\widehat{\mathbf{p}}(s_j, s_i) = \mathbf{p}(s_j) - \mathbf{p}(s_i)$. To achieve rotational invariance, we compute $\widehat{\mathbf{p}}(s_j, s_i)$ in the local coordinates of s_i, with x-axis being the drawing direction (tangent) at s_i (Figure 9). We compute $\widehat{\mathbf{m}}$ in the global coordinate frame. α, β and γ are weighting parameters discussed in Section 4.4.

Neighborhood As illustrated in Figure 9, we define the neighborhood $\mathbf{n}(s)$ of s as the set of samples that are located within its spatial vicinity (in the same frame) and drawn before s. This temporal causality allows a neighborhood to remain fixed once computed, facilitating fast incremental computation.

We measure the *local similarity* between two local neighborhoods

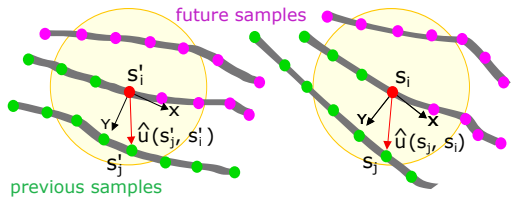


Figure 9: *Illustration of samples and neighborhoods. Each stroke (gray color) is represented by samples (small colored circles). The neighborhood $\mathbf{n}(s_i)$ of each sample s_i (red color) only contains samples (1) within its spatial vicinity (yellow circle) and (2) drawn before it (green color). The similarity between two sample neighborhoods $\mathbf{n}(s'_i)$ and $\mathbf{n}(s_i)$ is calculated by first matching neighborhood samples (e.g. s_j and s'_j) followed by summing up their pairwise similarities (e.g. $\widehat{\mathbf{u}}(s'_j, s'_i)$ and $\widehat{\mathbf{u}}(s_j, s_i)$). In particular, the spatial similarity ($\widehat{\mathbf{p}}(s'_j, s'_i)$ and $\widehat{\mathbf{p}}(s_j, s_i)$) are matched in the local frames (black axes) of s'_i and s_i.*

$\mathbf{n}(s'_i)$ and $\mathbf{n}(s_i)$ by summing up the pairwise similarity of their neighborhood samples:

$$\pi(s'_i, s_i) = \sum_{s'_j \in \mathbf{n}(s'_i), s_j \in \mathbf{n}(s_i)} \sigma(s'_j, s_j), \tag{3}$$

where s_j runs through all samples in $\mathbf{n}(s_i)$, s'_j is the "matching" sample of s_j selected via the greedy matching method described in Section 4.4, and $\sigma(s_j, s'_j)$ is the *pairwise similarity* between two neighborhood samples:

$$\sigma(s'_j, s_j) = \exp\left(-\left|\widehat{\mathbf{u}}(s'_j, s'_i) - \widehat{\mathbf{u}}(s_j, s_i)\right|^2\right), \tag{4}$$

with $\widehat{\mathbf{u}}(s_j, s_i)$ defined in Equation (2).

4.2 Global Similarity Analysis

Equation (3) only measures the local similarity between samples, which may cause mismatches due to ambiguity, as illustrated in Figure 8. We address this by extending this local similarity to a *global similarity*, which can be applied to samples in the same frame to identify spatial repetitions like [Xing et al. 2014] as well as samples across different frames for sketching animations.

As illustrated in Figure 8, the global similarity Π of two samples s'_i and s_i is influenced by two factors, the global similarity of their neighborhood samples (recursively) and their local neighborhood similarity σ defined in Equation (4). We formulate Π as

$$\Pi(s'_i, s_i) = \sum_{s'_j \in \mathbf{n}(s'_i), s_j \in \mathbf{n}(s_i)} \Pi(s'_j, s_j)\sigma(s'_j, s_j). \tag{5}$$

Equation (5) can be seen as a Π-weighted extension of Equation (3).

Note that Equation (5) is a recursive formulation. Since the neighborhood only contains nearby samples drawn before (Figure 9), the process goes only backward in time and thus avoids infinite cycles. Each recursive step is calculated in a different local coordinate frame (illustrated in Figure 9) to adapt to the local structures (e.g. red arrows following branches in Figure 10), and the whole recursive process can diffuse to a larger area by going through multiple local frames (e.g. blue circles in Figure 10). Since our system calculates and stores all previous global similarities, the recursive formulation can be easily calculated incrementally as the user draws. In contrast, although broader contexts can also be captured by local similarity with very large neighborhoods [Belongie et al. 2002] (e.g. enlarging the yellow circles in Figure 10), it is harder to accelerate and less adaptive to local structural changes than global similarity, e.g. the deformed branches in Figure 10.

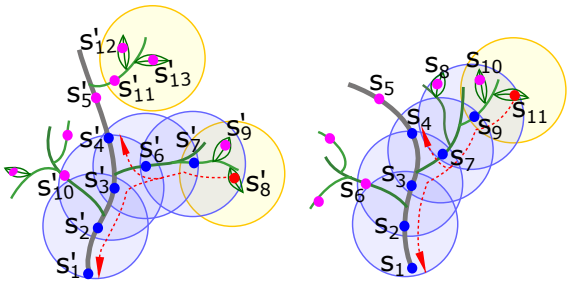


Figure 10: *Global similarity example. When calculating the global similarity between s'_8 and s_{11} (red dots), we start with their local neighborhoods (yellow circles) and gradually diffuse (indicated by the red arrows) towards the nearby neighborhoods (blue circles), capturing global structures in a larger area (blue samples). The process only goes backward in time, e.g. no diffusion to magenta samples. Due to the different drawing orders between $s'_{8,9}$ and $s_{10,11}$, s_{11} matches less well with s'_8 than s'_{13} locally. However, the diffused structures (blue circles) indicate a better global match between s_{11} and s'_8 than s'_{13}.*

Since our global similarity is temporally causal (e.g. neighborhoods and diffusion order), it is influenced by the drawing orders. In particular, the more coherent the drawing orders among frames, the more effective the analysis is. For example in Figure 9, if the user draws top-down in the left frame and bottom-up in the right frame, the global context between the two central red samples would be very different. Fortunately, since people usually draw coherently [Iarussi et al. 2013], our method works well in practice.

Initialization Since the global similarity is unknown for the first frame or at the beginning of drawing each subsequent frame, we initialize $\Pi(s', s)$ as a uniform/constant value.

Normalization For each new sample s produced in the current frame, we calculate its global similarity $\Pi(s', s)$ with each sample s' in previous frame, and normalize the sum $\sum_{s'} \Pi(s', s)$ to 1. Such normalization is to control each $\Pi(s', s)$ to be within $(0, 1)$ and not influenced by the recursion depth in Equation (5).

Refinement As illustrated in Figure 8, the global similarity can be ambiguous at the initial stage. We thus use subsequent drawings to gradually refine previous global similarities based on the consistency between matching samples [Huang et al. 2008]. In particular, we define (s', s) as a *matching pair* if $\Pi(s', s)$ is over a threshold value. We then construct graphs of matching pairs by establishing an edge between two matching pairs (s'_i, s_i) and (s'_j, s_j) if $s'_j \in \mathbf{n}(s'_i)$ and $s_j \in \mathbf{n}(s_i)$. As the user draws, we incrementally construct the graphs based on the above criterion for all matching pairs. For each matching-pair graph, we compute the whole consistency by summing $\Pi(s', s)$ over the graph. The larger the whole consistency, the more likely the better sample correspondences, and therefore we use it as a weight to modify each $\Pi(s', s)$:

$$\widetilde{\Pi}(s', s) = \Pi(s', s) \sum_{(\varsigma', \varsigma) \in \Phi} \Pi(\varsigma', \varsigma), \tag{6}$$

where Φ is the graph where matching-pair (s', s) belongs to, and $\widetilde{\Pi}(s', s)$ is the refined global similarity.

4.3 Prediction Synthesis

We predict the current frame by deforming drawings from the previous frames. For quality and consistency, the deformation should (1) maintain the structures and details of previous frames, (2) fit the drawing in the current frame, (3) observe the motions among

the frames for temporal coherence, and (4) be general enough for various drawing styles and contents and yet fast enough for interaction. Based on these considerations, we extend the embedded deformation model and optimization method in [Sumner et al. 2007; Li et al. 2008]. A key for such deformation is establishing correspondences between drawings across frames. Existing matching methods [Belongie et al. 2002; Li et al. 2008] are designed for sufficiently complete and coherent objects. However, manual drawing is largely incomplete during dynamic updates, and users may draw the same object across frames incoherently, such as with different numbers, shapes, and orders of strokes. To address these challenges, we use the global similarity in Section 4.2 to dynamically measure the confidence of correspondences, similar to the fuzzy correspondence idea in [Kim et al. 2012].

In describing our method below, we denote s, s', and s'' as samples in the current frame, the previous frame, and the frame before the previous frame, respectively, as illustrated in Figure 11.

Deformation model We assign an affine transformation $\mathbf{m}(s')$ for each s' in order to represent the local deformation on its nearby space. $\mathbf{m}(s')$ is specified by a 2×2 matrix $\mathbf{A}'$ and a 2×1 translation vector δ', both centered at s'. Under the influence of sample s'_i, each nearby sample s'_j is transformed to a new position $\widetilde{\mathbf{p}'_{ji}}$ according to:

$$\widetilde{\mathbf{p}'_{ji}} = \mathbf{A}'_i \left(\mathbf{p}(s'_j) - \mathbf{p}(s'_i)\right) + \mathbf{p}(s'_i) + \delta'_i. \tag{7}$$

In particular, when s'_i equals s'_j, $\widetilde{\mathbf{p}'_{jj}}$ can be simplified to $\mathbf{p}'_j + \delta'_j$.

Constraints Similar to [Sumner et al. 2007; Li et al. 2008], our deformation model includes the rigidity term E_r and smoothness term E_s to maintain the structure and details of the drawing in previous frames, and the fitness term E_f to pull the deformation towards current drawing. Specifically:

E_r penalizes the deviation of each $\mathbf{A}'$ from a pure rotation, and is defined as:

$$E_r = \sum_i \gamma(\mathbf{A}'_i), \tag{8}$$

where $\gamma(\mathbf{A}') = (a_1^T a_2)^2 + (1 - a_1^T a_1)^2 + (1 - a_2^T a_2)^2$, and a_1 and a_2 are the column vectors of $\mathbf{A}'$

E_s penalizes transformations that are inconsistent with one another:

$$E_s = \sum_{s'_i} \sum_{s'_j \in \mathbf{n}(s'_i)} \kappa_{ij} \left| \widetilde{\mathbf{p}'_{ji}} - \widetilde{\mathbf{p}'_{jj}} \right|^2, \tag{9}$$

where s'_j is a neighborhood sample of s'_i, $\widetilde{\mathbf{p}'_{ji}}$ and $\widetilde{\mathbf{p}'_{jj}}$ are the deformed position of s'_j under influence of s'_i and s'_j, respectively (Equation (7)). κ_{ij} is a weight factor measuring the likelihood that the transformation influence of s'_i is consistent with s'_j. As illustrated in Figure 11, the transformation consistency could be measured by the spatial consistency between s'_i and s'_j across frames. We use the local analysis in [Xing et al. 2014] to measure this spatial consistency:

$$\kappa_{ij} = \exp\left(-\frac{\theta\left(\widehat{\mathbf{p}}(s'_i, s'_j)\right)}{\theta_1}\right), \tag{10}$$

where the relative position $\widehat{\mathbf{p}}(s'_i, s'_j)$ is as in Equation (2), and $\theta(\widehat{\mathbf{p}}(s'_i, s'_j))$ is its variation across frames (e.g. $\widehat{\mathbf{p}}(s''_2, s''_3)$ and $\widehat{\mathbf{p}}(s'_2, s'_3)$ in Figure 11). We calculate $\theta(\widehat{\mathbf{p}}(s'_i, s'_j))$ by averaging the standard deviations of individual vector/matrix components:

$$\theta(.) = \overline{\mathrm{std}(.)}. \tag{11}$$

We discuss the parameter θ_1 in Section 4.4.

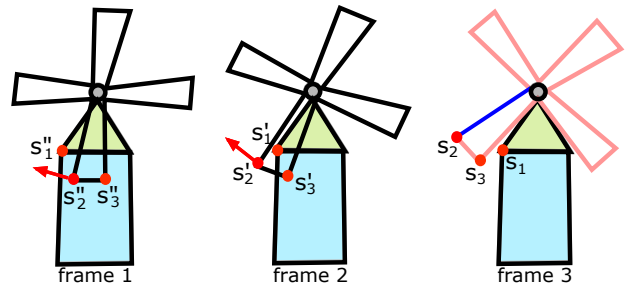


Figure 11: *Motion consistency and prediction. The user has drawn the first two frames and the black strokes in frame 3. Our system provides hint (in red strokes) while the user is drawing the blue stroke in frame 3. In particular, s_1 is a sample already drawn by the user, s_2 is currently being drawn, and s_3 is predicted by our system. Instead of regularizing the motion of each sample in frame 2 (e.g. $s'_{1,2,3}$) to be consistent with one another (in Equation (9)), those samples with similar motions in frame 1 (e.g. $s''_{2,3}$) should maintain similar motions in frame 2 (e.g. $s'_{2,3}$ but not with s'_1). The stability of relative positions between samples across frames (e.g. $\widehat{\mathbf{p}}(s''_2, s''_3)$ and $\widehat{\mathbf{p}}(s'_2, s'_3)$) can be used to measure the similarity of their motions (Equation (10)). The predicted hint also observes the motion in previous frames (Equation (13)), e.g. s_3 is shifted from s'_3 with a similar relative motion from s''_3, as indicated by the red arrows, in addition to the relative position to s_2.*

E_f serves as a position constraint so that the deformed shape is close to the current drawing:

$$E_f = \sum_{s'_i} \widetilde{\Pi}(s'_i, s_i) \left|\mathbf{p}(s'_i) + \delta'_i - \mathbf{p}(s_i)\right|^2, \tag{12}$$

where s_i is the corresponding sample of s'_i in the current frame, and $\widetilde{\Pi}(s'_i, s_i)$ is the refined global similarity (Equation (6)) between s'_i and s_i, as a confidence of their correspondence.

In order for the deformation to observe the motion among frames, we also add an additional motion constraint term E_m based on the assumption that neighboring frames should have similar motions (Figure 11):

$$E_m = \sum_{s'_i} \kappa_{\delta'_i} \left|\delta'_i - \delta''_i\right|^2 + \kappa_{\mathbf{A}'_i} \|\mathbf{A}'_i - \mathbf{A}''_i\|_F^2, \tag{13}$$

where s''_i is the corresponding sample of s'_i in its previous frame and $\|.\|_F$ is the Frobenius norm. We use the transformation of s''_i (δ''_i and $\mathbf{A}''_i$) as the prediction of δ'_i and $\mathbf{A}'_i$, and use $\kappa_{\delta'_i}$ and $\kappa_{\mathbf{A}'_i}$ to measure the quality of such prediction:

$$\kappa_{\delta'_i} = \exp\left(-\frac{\theta\left(\delta'_i\right)}{\theta_2}\right), \ \kappa_{\mathbf{A}'_i} = \varphi_1 \exp\left(-\frac{\theta\left(\mathbf{A}'_i\right)}{\theta_3}\right), \tag{14}$$

where θ is defined in Equation (11). θ_2, θ_3 and φ_1 are parameters discussed in Section 4.4. Since there is no s''_i defined for the first two frames, we consider E_m starting from the third frame.

Correspondence Similar to [Li et al. 2008; Huang et al. 2008], we need to update the correspondences during the optimization pro-

cess. Instead of ICP as in these methods, for more accurate correspondence and faster search speed we dynamically update the corresponding sample s_i of each s'_i in Equation (12) accordingly to:

$$s_i = \arg\max_s \left(\widetilde{\Pi}(s'_i, s) \left(\varphi_2 + \exp\left(-\frac{|\widehat{\mathbf{p}}(s'_i, s)|}{\theta_4} \right) \right) \right), \quad (15)$$

where s runs through all matching samples of s'_i as illustrated in Figure 8, and $|\widehat{\mathbf{p}}(s'_i, s)|$ is the spatial distance between samples s'_i and s. We choose this formulation so that the correspondence update is dominated by the global similarity term $\Pi(s', s)$ when s' and s are far away, and yet allows local adjustment when they are close. The values of φ_2 and θ_4 are described in Section 4.4.

Optimization The optimization function to be minimized can be represented as follows by summing the above constraint terms:

$$E\left(\{\mathbf{m}(s')\}\right) = \kappa_r E_r + \kappa_s E_s + \kappa_f E_f + \kappa_m E_m. \quad (16)$$

The unknowns comprise the motion parameters $\{\mathbf{m}(s')\}$ and the correspondences in Equation (12) for samples between the previous and current frames. We follow a basic framework of non-rigid point cloud registration [Li et al. 2008] by alternating, until convergence, the following steps: (1) updating correspondence (Equation (15)) with $\{\mathbf{m}(s')\}$ fixed, and (2) refining motion parameters $\{\mathbf{m}(s')\}$ via the Levenberg-Marquardt method with the correspondences fixed. Similar to [Li et al. 2008], we adopt a heuristic that automatically updates the optimization weights to initially favor a rigid alignment and subsequently lowers the stiffness to allow increasing deformation as the optimization progresses. This is described in more details in Section 4.4.

Our system updates the deformation every time the user draws a new stroke. When the user creates a new empty frame, Equation (13) controls the deformation as there is no stroke drawn in the current frame. As the user draws more strokes, Equation (12) has more influence on the deformation. This means that deformation will adapt to the current drawing.

Prediction After getting the motion parameters from the optimization process in Equation (16), we can calculate the position of each sample based on Equation (7). These samples are then interpolated to reconstruct the strokes for prediction.

4.4 Implementation

Sample parameters We sample all strokes with an average of 15-pixel spacing to balance between speed and accuracy. This sampling is uniform at low curvature regions but adaptive at high curvature regions, where the spacing between two neighboring samples vary between 12 to 18 pixels.

Each term in Equation (1) is represented as a vector without normalization. $\mathbf{p}(s)$ is the pixel-wise position within the canvas and $\mathbf{a}(s)$ is an 8-bit RGBA color. The drawing direction at each sample is not stored, as it can be quickly derived from nearby sample positions. In Equation (4), we set $\alpha = 1$ and $\beta = 1$ for the calculation of $\widehat{\mathbf{u}}(s', s)$. To make the neighborhood matching observe the stroke topology, we set $\gamma = 100$ when sample s' and s belong to the same stroke, and $\gamma = 0$ otherwise.

Neighborhood matching For each sample s, we first extract all previous samples (in the same frame) within a distance 10% of the canvas size, then select the nearest 7 samples as its neighborhood $\mathbf{n}(s)$. When matching $\mathbf{n}(s)$ with $\mathbf{n}(s')$ in Equation (5), we determine the pairings in Equation (3) by first identifying the pair (s'_j, s_j) with largest $\sigma(s'_j, s_j)$, excluding them from further consideration, and repeating the process to find the next pair until s_j runs out of samples [Ma et al. 2011]. In particular, we use a partial neighborhood of $\mathbf{n}(s)$ (containing the 5 nearest samples) to match with the full neighborhood $\mathbf{n}(s')$ to prevent $\mathbf{n}(s')$ from running out of samples before $\mathbf{n}(s)$.

Matching sample Similar to [Xing et al. 2014], we apply an online method to measure the threshold for selecting global matching samples. Specifically, we find the largest global similarity Π_{max} as a reference, and define as matching samples those with global similarity larger than $0.3 \times \Pi_{max}$.

Constraint parameters We set $\theta_1 = 5$, $\theta_2 = 5$, and $\theta_3 = 0.5$ for Equations (10) and (14), and $\varphi_1 = 50$ in Equation (13).

Correspondence parameters In Equation (15), we set $\theta_4 = 20$ and $\varphi_2 = 0.1$.

Optimization parameters For Equation (16), we initialize $\kappa_r = 1000$, $\kappa_s = 15$, $\kappa_f = 10$, and $\kappa_m = 1$. κ_s is halved whenever $|E_k - E_{k-1}| < 10^{-1}(1 + E_k)$, where E_k is the cost at the k-th iteration, until $\kappa_s < 1$. Other weightings are held constant during the optimization. Since Equation (13) starts from the third frame, we set $\kappa_m = 0$ for the second frame.

5 User Evaluation

We conducted a preliminary user study to evaluate the usability of our system by comparing (1) fully manual drawing as in traditional tools and (2) interactive drawing enabled by our system.

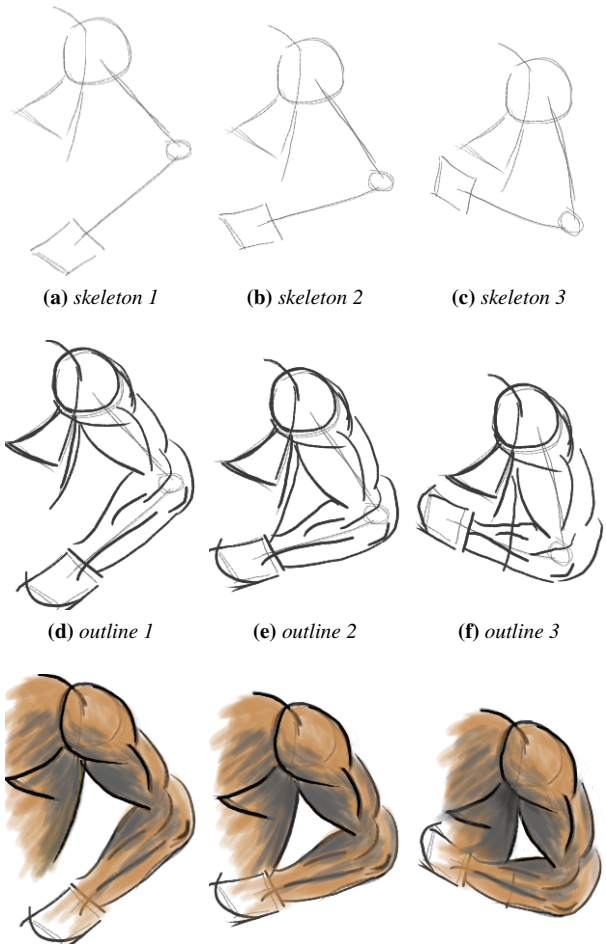
5.1 Procedure

We recruited 2 experienced animation sketching artists and 7 novice users with different levels of sketching experiences as participants. All tasks were conducted on a 13-in laptop with a Wacom tablet. The study consisted of four sessions: warm-up (20 min), target sketch (50 min), open sketch (20 min), and final interview (10 min).

Warm-up session The warm-up session was designed to familiarize the participants with the interface and various functions of our system. The tasks consisted of sketching and animating simple objects, such as a bouncing ball, a swinging flower, or a walking stick-man. One of the authors gave the participants guidance throughout the entire process.

Target sketch session The goal is to measure and compare the objective performance and subjective experience of traditional animation authoring versus our system. We asked our collaborating artists to create 3 frames of an animated object with initial skeletons and final outputs (Figure 12). We then asked the participants to draw the outlines of all frames first followed by adding the details and colors, using the initial artist skeletons for start and the complete artist drawings for reference. Each participant performed this task both with and without our autocomplete functions, and the orders of these two conditions were counter-balanced among all participants.

Open sketch session The goal of this session was to identify merits of our system in various types of drawings and uncover its potential usability issues. Participants were free to perform open-ended drawings using our system with the only requirement of creating at least three frames. One of the authors accompanied the participants through the session and encouraged them to try out different features provided by the system.

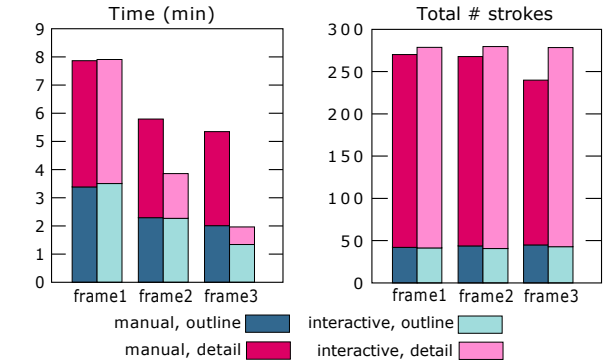


(a) *skeleton 1* (b) *skeleton 2* (c) *skeleton 3*

(d) *outline 1* (e) *outline 2* (f) *outline 3*

(g) *finish 1* (h) *finish 2* (i) *finish 3*

Figure 12: *Target sketching task. Our participating artists created the initial skeletons (top row) and final drawings (not shown but similar to the bottom row). We then ask each participant to sketch the outlines first (middle row), followed by adding the details and colors (bottom row).*



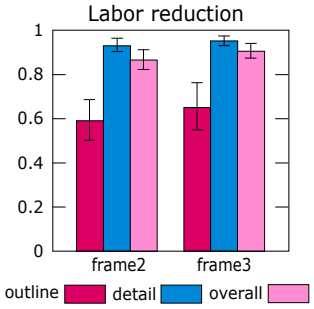


Figure 13: *Performance. We measure the completion time and total # of strokes (including both manual-user and automatic-system strokes) for the tasks in Figure 12 to compare the traditional manual and our interactive drawing modes. We also show labor reduction, the ratio of automatic-system to total # of strokes, as mean ± standard deviation.*

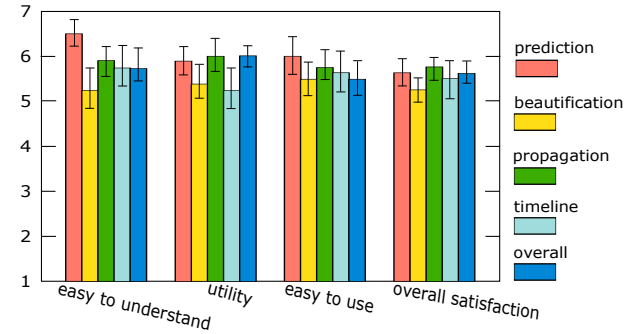


Figure 14: *User feedback. All quantities are expressed as mean ± standard deviation in a 7-point Likert scale.*

5.2 Outcome

Figure 13 provides quantitative measures of completion time and stroke counts for the target sketch task in Figure 12. The ratio of strokes-per-minute by our system versus manual drawing was 1.57 and 3.16 for the second and third frames. The ratio was greater (4.32) when adding details than when sketching outlines (1.2).

Figure 13 also shows *labor reduction* – the ratio of autocomplete strokes accepted by the user to the total number of strokes. The ratio is 0.62 for outline drawing and 0.97 for details drawing. The labor reduction measure indicates that our system can help users avoid a substantial amount of repetition. When asked about comparing traditional manual with our approach, most participants (8 out of 9) stated that they easily lost patience to manually add details for each frame, and our system was able to help them reduce a large amount of work and maintain better consistency among frames.

Figure 14 summarizes subjective feedback from our participants about the functionalities in our system. Overall, participants were satisfied with our system. Seven of the participants commented that they were able to easily switch between painting, selection, and timeline preview modes, while the other two participants told us they sometimes entered the selection mode by accident. The professional artists also suggested the addition of motion path prediction as it is difficult to design a good motion path without enough experience, especially for complicated animations.

6 Results and Comparison

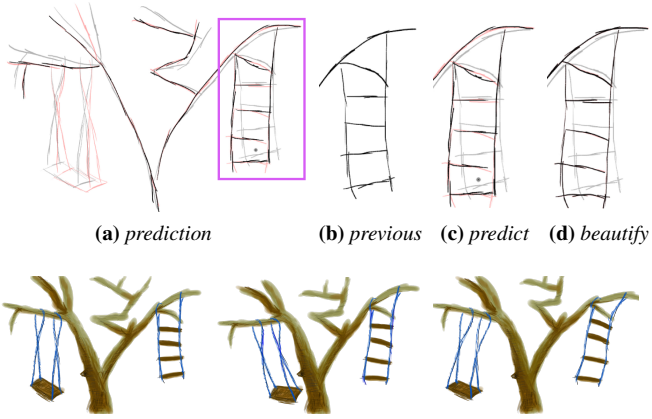
We demonstrate how our system can help author hand-drawn animations with higher quality and less effort, by presenting sample open-sketch results and comparing with 2D shape deformation.

To direct viewers' attention and save creators' workload, it is common to animate only parts of a scene while keeping the rest stationary. Figure 15 is one such partial-animation example. Here, an artist drew the first frame of a cartoon character and wanted to animate only its ears. Since there was only one existing frame, our system predicted the second frame to be identical to the first one at the beginning. The artist accepted this stationary hint for the majority of the character (Figure 15a) and manually updated only the parts that needed to be deformed or animated (Figure 15b). Our system could automatically suggest potential beautification based on past drawings as shown in Figure 15c. After finishing the outlines of the animation, the artist started to add details and fill in colors (Figure 15d). This only needed to be done in one frame and our system automatically propagated and adapted the edits to all other frames (Figure 15e). As can be seen from Figures 15c and 15f, our system was able to help reduce manual workload while enhancing

(a) *partial accept initial hint* **(b)** *partial animation* **(c)** *beautification*

(d) *fill and detail* **(e)** *propagation and refinement* **(f)** *finish*

Figure 15: *Partial-animation example. The user outlines the first frame of a cartoon character (a) and would like to animate its ears while keeping the rest stationary. This can be achieved by accepting the stationary hint (purple in (a)) while redrawing the ears (red in (a)) in slightly changed shapes (b). Our system can automatically beautify the drawing to maintain temporal-spatial consistency (c). After finishing the outlines of the whole animation, the user starts to fill in colors and details to the first frame (d), and our system will automatically propagate and adapt the changes to the other frames (e). The user can accept, ignore, or modify the suggestions (e). The final result is in (f).*



(a) *prediction* **(b)** *previous* **(c)** *predict* **(d)** *beautify*

(e) *1299/1299 op, 45 min* **(f)** *165/1371 op, 7 min* **(g)** *198/1429 op, 6 min*

Figure 16: *Rough sketch example. Despite of the differences (e.g. number and length of strokes) in drawing the same object across frames, our system can detect the repetitions and establish correspondences to suggest prediction (e.g. the swing in (a)) and beautification (e.g. the ladder in (d)) with temporal and spatial consistency. (e) to (g) are finished frames with statistics described in Figure 17.*

animation quality.

Unlike the smooth, consistent, and integral curves in Figure 15, Figure 16 demonstrates a different drawing style with fast, rough, random, and short strokes. In particular, a participant drew the same object with different number, length, or order of strokes across different frames, such as the ladder in Figures 16b and 16c. Even with such inconsistency, our system was still able to detect the repetitions and establish correspondences to suggest prediction (e.g. the swing in Figure 16a) and beautification (e.g. Figure 16d).

As demonstrated in Figure 17, our system can support authoring animations with different complexities and characteristics. As indicated by the statistics, with the assistance enabled by our system, it generally took much less time and effort to draw subsequent frames of an animation compared to the first frame. This can help both novices and professionals to create a variety of sketch-based animations. As shown in the statistics, each frame often has a different number of user- and total-strokes. This reflects the open-ended nature of our system which facilitates topological changes (e.g. a broken heart or a growing plant) in addition to geometrical changes offered by prior deformation-based methods.

Shape deformation has been employed to create 2D animations by first specifying certain forms of control, such as handles [Igarashi et al. 2005], cages [Weber et al. 2009], or curves [Guay et al. 2015], followed by manipulating these controls to achieve the desired shape deformation. Specifying and manipulating such controls can become quite tedious for complex deformations or shapes, and is not very suitable for topological changes such as adding, removing, or breaking objects. Our method, in contrast, treats user sketches as an *implicit* form of control, allowing natural interaction and complete freedom in expression without any restrictions in the form or flow of sketch. Figure 18 visually compares explicit deformation and our sketching method. Several results shown in the paper are difficult (if not impossible) to author via deformation, such as the animated shading effect in Figure 12, and the broken-heart, growing-plant, and walking stick figure in Figure 17.

7 Limitations and Future Work

The core of our system is the analysis and synthesis of drawing repetitions. Since our current method is built upon the assumption that users draw in a coherent fashion across frames (discussed in Section 4.2), it is not guaranteed to offer predictions or sug-

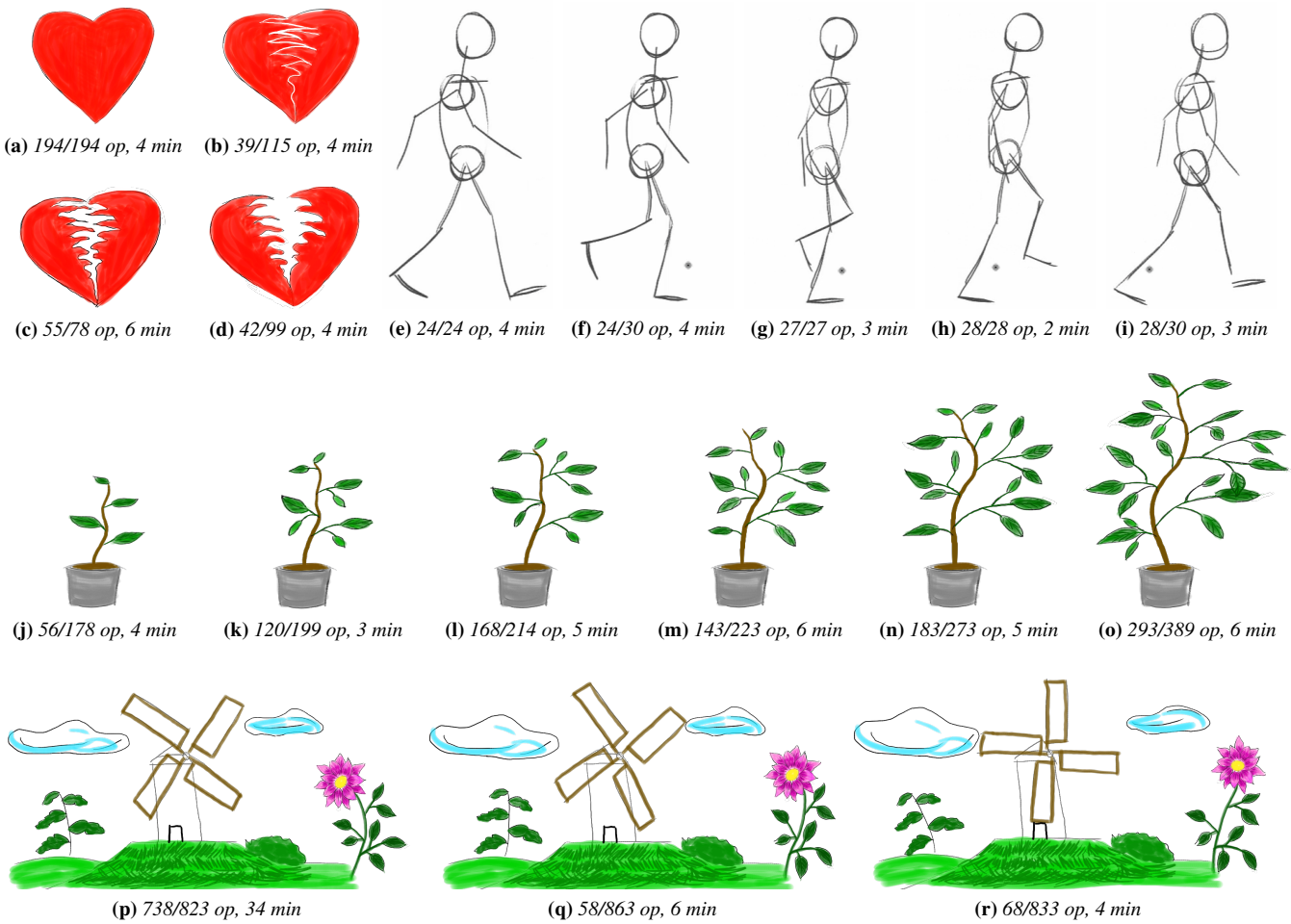


Figure 17: *More results. Shown here are example animation frames produced by participants with our system. Each frame is denoted with the following statistics: number of manual drawn strokes, total number of strokes (manual + automatically generated by the system), and drawing time in minutes.*

gestions users would desire to apply. However, the system is no worse than animation authoring without any assistance as users can choose to ignore or modify the suggestions. Our current method is hand-crafted to offer interactive performance, but its quality could be further enhanced by machine learning techniques which have shown promises in text and hand-writing synthesis [Graves 2013].

Our current implementation and user study are based on a pen-input system (Wacom tablet). We plan to deploy a touch-screen version of our system to further evaluate and refine its usability for mobile devices. This also allows the possibility of crowd-sourcing [Limpaecher et al. 2013] to help learn and create animations.

Our current user interface provides only the temporal extension of the hint mode in [Xing et al. 2014]. Temporal extension of the clone mode in [Xing et al. 2014] could be achieved via algorithm components in Section 4. Such spatial-temporal clone can be useful for scenarios such as creating a large herd of creatures from a single animated one. We leave this as a potential future work.

Acknowledgements

We would like to thank Carrie de Souza and Jim Kilmer for help resolving a file upload issue, our user study participants for their time and feedbacks, and anonymous reviewers for their valuable comments. This work has been partially supported by a HKU postgraduate fellowship (UPF), general research fund *Workflow Textures* (project number 17202415), and the Microsoft Research Asia CORE 11 program.

References

BAXTER, W. V., AND ANJYO, K. 2006. Latent doodle space. *Comput. Graph. Forum 25*, 3, 477–485.

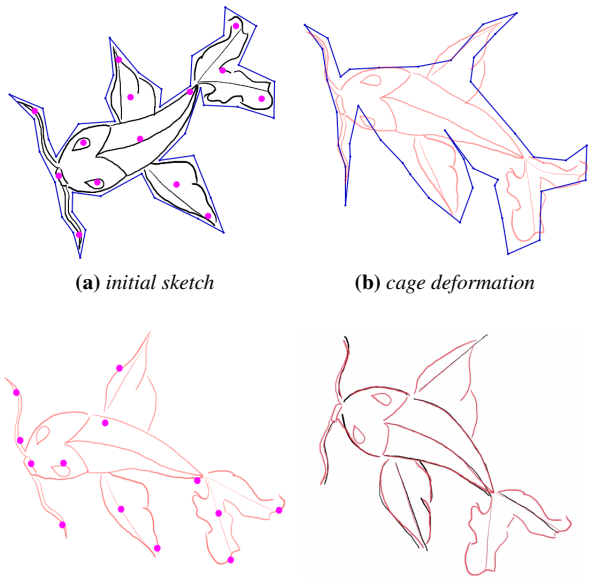
BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell. 24*, 4 (Apr.), 509–522.

BÉNARD, P., COLE, F., KASS, M., MORDATCH, I., HEGARTY, J., SENN, M. S., FLEISCHER, K., PESARE, D., AND BREEDEN, K. 2013. Stylizing animation by example. *ACM Trans. Graph. 32*, 4 (July), 119:1–119:12.

CHEN, H.-T., GROSSMAN, T., WEI, L.-Y., SCHMIDT, R. M., HARTMANN, B., FITZMAURICE, G., AND AGRAWALA, M. 2014. History assisted view authoring for 3d models. In *CHI '14*, 2027–2036.

CHU, N. S.-H., AND TAI, C.-L. 2005. Moxi: Real-time ink dispersion in absorbent paper. *ACM Trans. Graph. 24*, 3 (July), 504–511.

DAVIS, J., AGRAWALA, M., CHUANG, E., POPOVIĆ, Z., AND SALESIN, D. 2003. A sketching interface for articulated figure animation. In *SCA '03*, 320–328.



(a) *initial sketch* **(b)** *cage deformation*

(c) *handle deformation* **(d)** *our method*

Figure 18: *Explicit deformation versus implicit sketching. In traditional deformation-based methods, users need to explicitly specify certain forms of control, such as a cage (blue) or handles (magenta) in (a), and manipulate these controls, such as the cage [Weber et al. 2009] (b) or handles [Igarashi et al. 2005] (c), to deform the underlying shape (red). In contrast, our method can achieve such deformation implicitly via direct sketching (d).*

GRABLER, F., AGRAWALA, M., LI, W., DONTCHEVA, M., AND IGARASHI, T. 2009. Generating photo manipulation tutorials by demonstration. *ACM Trans. Graph. 28*, 3 (July), 66:1–66:9.

GRAVES, A. 2013. Generating sequences with recurrent neural networks. *CoRR abs/1308.0850*.

GUAY, M., RONFARD, R., GLEICHER, M., AND CANI, M.-P. 2015. Space-time sketching of character animation. *ACM Trans. Graph. 34*, 4 (July), 118:1–118:10.

HUANG, Q.-X., ADAMS, B., WICKE, M., AND GUIBAS, L. J. 2008. Non-rigid registration under isometric deformations. In *SGP '08*, 1449–1457.

IARUSSI, E., BOUSSEAU, A., AND TSANDILAS, T. 2013. The drawing assistant: Automated drawing guidance and feedback from photographs. In *UIST '13*, 183–192.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph. 24*, 3 (July), 1134–1141.

JONES, B., POPOVIC, J., MCCANN, J., LI, W., AND BARGTEIL, A. W. 2015. Dynamic sprites: artistic authoring of interactive animations. *Journal of Visualization and Computer Animation 26*, 2, 97–108.

KAZI, R. H., CHEVALIER, F., GROSSMAN, T., AND FITZMAURICE, G. 2014. Kitty: Sketching dynamic and interactive illustrations. In *UIST '14*, 395–405.

KAZI, R. H., CHEVALIER, F., GROSSMAN, T., ZHAO, S., AND FITZMAURICE, G. 2014. Draco: Bringing life to illustrations with kinetic textures. In *CHI '14*, 351–360.

KIM, V. G., LI, W., MITRA, N. J., DIVERDI, S., AND FUNKHOUSER, T. 2012. Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph. 31*, 4 (July), 54:1–54:11.

LEE, Y. J., ZITNICK, C. L., AND COHEN, M. F. 2011. ShadowDraw: Real-time user guidance for freehand drawing. *ACM Trans. Graph. 30*, 4 (July), 27:1–27:10.

LI, H., SUMNER, R. W., AND PAULY, M. 2008. Global correspondence optimization for non-rigid registration of depth scans. In *SGP '08*, 1421–1430.

LIMPAECHER, A., FELTMAN, N., TREUILLE, A., AND COHEN, M. 2013. Real-time drawing assistance through crowdsourcing. *ACM Trans. Graph. 32*, 4 (July), 54:1–54:8.

LINDEMEIER, T., METZNER, J., POLLAK, L., AND DEUSSEN, O. 2015. Hardware-based non-photorealistic rendering using a painting robot. *Computer Graphics Forum 34*, 2, 311–323.

LU, J., BARNES, C., DIVERDI, S., AND FINKELSTEIN, A. 2013. Realbrush: Painting with examples of physical media. *ACM Trans. Graph. 32*, 4 (July), 117:1–117:12.

LUKÁČ, M., FIŠER, J., BAZIN, J.-C., JAMRIŠKA, O., SORKINE-HORNUNG, A., AND SÝKORA, D. 2013. Painting by feature: Texture boundaries for example-based image creation. *ACM Trans. Graph. 32*, 4 (July), 116:1–116:8.

MA, C., WEI, L.-Y., AND TONG, X. 2011. Discrete element textures. *ACM Trans. Graph. 30*, 4 (July), 62:1–62:10.

MILLIEZ, A., NORIS, G., BARAN, I., COROS, S., CANI, M.-P., NITTI, M., MARRA, A., GROSS, M., AND SUMNER, R. W. 2014. Hierarchical motion brushes for animation instancing. In *NPAR '14*, 71–79.

MYERS, B. A., LAI, A., LE, T. M., YOON, Y., FAULRING, A., AND BRANDT, J. 2015. Selective undo support for painting applications. In *CHI '15*, 4227–4236.

NANCEL, M., AND COCKBURN, A. 2014. Causality: A conceptual model of interaction history. In *CHI '14*, 1777–1786.

NORIS, G., SÝKORA, D., COROS, S., WHITED, B., SIMMONS, M., HORNUNG, A., GROSS, M., AND SUMNER, R. W. 2011. Temporal noise control for sketchy animation. In *NPAR '11*, 93–98.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3 (July).

SÝKORA, D., DINGLIANA, J., AND COLLINS, S. 2009. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *NPAR '09*, 25–33.

THORNE, M., BURKE, D., AND VAN DE PANNE, M. 2004. Motion doodles: An interface for sketching character motion. *ACM Trans. Graph. 23*, 3 (Aug.), 424–431.

WEBER, O., BEN-CHEN, M., AND GOTSMAN, C. 2009. Complex barycentric coordinates with applications to planar shape deformation. *Computer Graphics Forum (Proceedings of Eurographics) 28*, 2.

WHITED, B., NORIS, G., SIMMONS, M., SUMNER, R., GROSS, M., AND ROSSIGNAC, J. 2010. BetweenIT: An interactive tool for tight inbetweening. *Comput. Graphics Forum (Proc. Eurographics) 29*, 2, 605–614.

XING, J., CHEN, H.-T., AND WEI, L.-Y. 2014. Autocomplete painting repetitions. *ACM Trans. Graph. 33*, 6 (Nov.), 172:1–172:11.

ZITNICK, C. L. 2013. Handwriting beautification using token means. *ACM Trans. Graph. 32*, 4 (July), 53:1–53:8.