Temporal Noise Control for Sketchy Animation

G. Noris[1,2] D. Sýkora[3] S. Coros[2] B. Whited[4] M. Simmons[4] A. Hornung[2] M. Gross[1,2] R. W. Sumner[2]
[1]ETH Zurich, CGL [2]Disney Research Zurich [3]CTU in Prague, FEE [4]Walt Disney Animation Studios

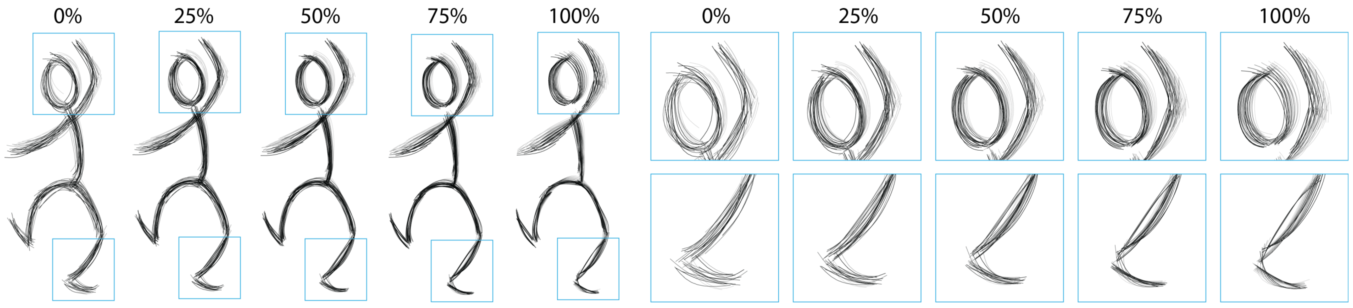

Figure 1: *Balancing scene. We compare different noise reduction values from* 0% *(input animation) to* 100% *(noise-free). Close-ups are marked by blue squares.*

Abstract

We propose a technique to control the temporal noise present in sketchy animations. Given an input animation drawn digitally, our approach works by combining motion extraction and inbetweening techniques to generate a reduced-noise sketchy animation registered to the input animation. The amount of noise is then controlled by a continuous parameter value. Our method can be applied to effectively reduce the temporal noise present in sequences of sketches to a desired rate, while preserving the geometric richness of the sketchy style in each frame. This provides the manipulation of temporal noise as an additional artistic parameter, e.g. to emphasize character emotions and scene atmosphere, and enables the display of sketchy content to broader audiences by producing animations with comfortable noise levels. We demonstrate the effectiveness of our approach on a series of rough hand-drawn animations.

1 Introduction

Compared to traditional cleaned-up drawings, sketches present a kind of visual richness, where both silhouette and interior lines are composed of many rough strokes. This style allows another dimension of expressiveness - emotion, action, and other features can be conveyed through the "sketchy" drawings.

The richness provided by the sketchy style can be considered to be a form of geometric noise. Despite its positive benefits in still images, geometric noise becomes temporal noise in sequences of sketches and is generally unpleasant to view. The industry solution to this problem is to remove the geometric noise. In production environments, early versions of animation (both 2D and 3D) are often composed of sequences of rough sketches. Later in the pipeline, these are systematically replaced either with clean-line drawings or with renderings of 3D scenes, which typically present cleaner visuals. Animations completely made of sketches are less common and generally confined to short sequences or small productions[1].

Our goal is to preserve the expressiveness inherent in sketchy drawings while removing unpleasant temporal issues. We propose to reduce *temporal* noise while keeping *geometric* noise by supporting interpolation at a finer level, down to the individual sketchy strokes. Instead of appearing and disappearing, the strokes transition smoothly from frame to frame. Enforcing these constraints manually in typical production environments would be impractical: establishing a fine-scale correspondence of strokes within the sequence of sketches and generating the proper animation path is too labor-intensive.

A key insight of our approach is that we can first construct a noise-free animation using only a representative subset of the input frames such that effectively all temporal noise is removed. Then, the desired amount of noise can be continuously varied on top of this noise-free animation. Another key idea is the use of motion extraction to allow local stroke searches within the global motion - enabling an automated solution to the fine-scale stroke correspondence problem.

2 Related Work

Research efforts related to sketchy or hand-drawn styles of illustration can be divided into two main topics: simplification and generation (static images as well as temporally coherent animations).

In the area of automated simpification/beautification, several techniques have been developed that reduce the number of lines in a drawing using a density measure to prune possibly redundant lines while still conveying the notion of the original shape [Wilson and Ma 2004; Grabli et al. 2004]. Instead of solely performing stroke reduction, Barla et al. [2005] propose a perceptually motivated technique for synthesizing representative lines. Input strokes are first grouped using a greedy clustering algorithm that pairs strokes according to screen-space proximity. A geometric reconstruction step then follows to produce a single line for each group. The described approach is for static drawings, though they propose incorporating a temporal aspect of perceptual grouping- "common fate", expressed by grouping line segments with similar velocity to produce coherent animation. Shesh et al. [2008] later extended this approach to handle time coherence by building a simplification hierarchy of strokes and using opacity blending to interpolate between a pair of strokes and its simplified version to create smooth transitions.

Generation of static sketchy or pen-and-ink style illustrations from input 2D images/photographs and from 3D models is a popular topic in the field of non-photorealistic animation and rendering (NPAR) (e.g. [Winkenbach and Salesin 1994; Salisbury et al. 1997; Coconu et al. 2006]). Far less attention has been focused

[1]Notable examples include Frédéric Back's short "L'homme qui plantait des arbres" and a brief series of sketches in the "Colors of the Wind" sequence in Disney's animated feature *Pocahontas*.

on developing temporally coherent results suitable for animation. Existing techniques [Curtis 1998; Bourdev 1998; Kalnins et al. 2002; Kalnins et al. 2003] focus on rendering stylized silhouettes of animated 3D models. After silhouettes are extracted from the 3D model the main challenge is to generate coherent "sketchiness" along the silhouette strokes over time, e.g. through assigning temporally coherent parameterizations to strokes in the image plane [Kalnins et al. 2003; Bourdev 1998] or alternatively achieving coherence via a particle system seeded along the silhouette[Curtis 1998] or through stroke texture representations [Bénard et al. 2010] designed for temporal coherence. The key issue with all of these approaches is that they require an underlying 3D model or a clean 2D image with known stroke correspondences. In our case the challenge is that we have a set of unordered strokes in each frame which are not necessarily moving coherently and we need to determine how to best match and interpolate them over time.

3 Method

The method we introduce offers artistic control over the level of temporal noise in hand-drawn animations. Mismatches in the individual strokes used to define the same silhouette in consecutive frames can be a major source of temporal noise, particularly in rough sketches. Generally speaking, we seek to maintain the global motion of an input animation, as well as the overall drawing style, while manipulating the temporal noise level. Smooth output animations are typically preferred, but we note that high-frequency noise can be an effective artistic tool. The animation of a character who is scared, for instance, could benefit from a certain amount of temporal noise to emphasize emotion.

Before describing our method, we introduce the notation used throughout the paper. Our algorithm operates on sequences of frames, where each frame $\mathcal{F}$ contains a set of strokes that appear in the animation at the same moment in time. Each stroke s is a piece-wise linear curve defined by a sequence of vertices. The i-th stroke in a frame $\mathcal{F}$ is given by $\mathcal{F}(i)$.

A motion field is a function $\mathcal{M} : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ that tracks the movement of every point on the 2D canvas. In particular, $\mathcal{M}_{\mathcal{F}_1}^{\mathcal{F}_2}$ is the motion field that describes the relative motion between frames $\mathcal{F}_1$ and $\mathcal{F}_2$. We define $\mathcal{D}(\mathcal{M}, \mathcal{F})$ to be the deformed frame that is obtained by taking the vertices of every stroke in $\mathcal{F}$ and displacing them according to $\mathcal{M}$.

3.1 Overview

Our approach takes as input a sequence of frames from a hand-drawn animation. The core algorithm works in two passes (see Figure 2). First, the input sequence is processed to create a *noise-free* animation (see Figure 2a). This is done by sampling the original input animation to choose representative frames. These frames are smoothly interpolated to create a new animation, and, for the time-being, all other frames not in the representative subset are ignored. The output of this stage is a set of smooth, automatically-created inbetween frames.

In the limit if we choose only the first and very last frame of the animation as the representative set, replacing frames from the original sequence with these newly synthesized inbetweens would effectively produce a noise-free animation. However, much of the finer scale motion and sketchy details would also be removed. On the other hand, if the representative set includes all of the original frames, then no interpolation is performed and we have the original, noisy content.

Our goal is to allow an artist to precisely control the level of temporal noise. This is enabled by the second pass of our algorithm, which smoothly interpolates the original noisy animation with the smooth inbetweens created during the first pass (see Figure 2b).

Algorithm 1 CreateInbetweens

```
 1: input $\mathcal{F}_1, \mathcal{F}_2$: animation frames
 2: input $t$: interpolation parameter, $0 \leq t \leq 1$
 3: output $\mathcal{F}_t$: an inbetween
 4: $\hat{\mathcal{F}}_1 \leftarrow \mathcal{D}(\mathcal{M}_{\mathcal{F}_1}^{\mathcal{F}_2}, \mathcal{F}_1)$
 5: $\mathcal{S} \leftarrow computeStrokeCorrespondencePairs(\hat{\mathcal{F}}_1, \mathcal{F}_2)$
 6: $\mathcal{F}_t \leftarrow \{\}$
 7: for all $(i, j) \in S$ do
 8:     $s \leftarrow interpolateStrokes(\mathcal{F}_1(i), \mathcal{F}_2(j), t)$
 9:     $\mathcal{F}_t \leftarrow \mathcal{F}_t \cup \{s\}$
10: end for
```

Both passes must solve the same problem of creating smooth inbetween frames. In other words, given two frames of animation, we must establish a fine-scale stroke correspondence and interpolate smoothly between each stroke pair.

3.2 Representative Frame Sampling

The first phase of our algorithm generates a noise-free sequence which ideally should resemble the original animation as much as possible. The sampling scheme (i.e. choice of representative frames) is crucial to the quality of the resulting animation. Our method is comprised of two main components: the sampling strategy and timing control.

We propose two selection strategies for choosing representative frames. *Uniform sampling* selects representative frames distributed at equal intervals specified by a window size w. *Keyframes* uses important or extreme frames that are manually selected from the original animation input.

Once the representative frames are selected, our system assumes a uniform division of time units inside each interval, resulting in an approximation of the input timing. Our experience is that, by respecting the input animation keyframes, this approximation is acceptable. However, for particular scenes it might be desirable to have a finer control. This can be achieved by altering the timing values used to generate the inbetweens in [Whited et al. 2010].

It is assumed that the keyframe specification and timing information when needed is explicitly provided as input, along with the original animation. This is suitable for a classic animation environment, where both keyframes and timing information are captured by timing charts.

3.3 Creating Smooth Inbetween Frames

Algorithm 1 describes the steps to create smooth inbetween frames. Both passes of our method use this algorithm. The input consists of a pair of representative animation frames, $\mathcal{F}_1$ and $\mathcal{F}_2$, and an interpolation parameter $t, 0 \leq t \leq 1$. Our goal is to create a new frame $\mathcal{F}_t$ using solely the strokes from frames $\mathcal{F}_1$ and $\mathcal{F}_2$. This process ensures continuity in the strokes that are output as t varies between 0 and 1, and thus results in smooth animations.

The first step towards creating smooth inbetween frames consists of identifying the pairs of strokes from $\mathcal{F}_1$ and $\mathcal{F}_2$ that represent the same features in the drawing at different moments in time. Every stroke from one input frame is matched to the most likely candidate stroke from the other frame, as expressed by a stroke correspondence measure. We refer to this process as finding the stroke-to-stroke correspondences (see Section 3.3.2).

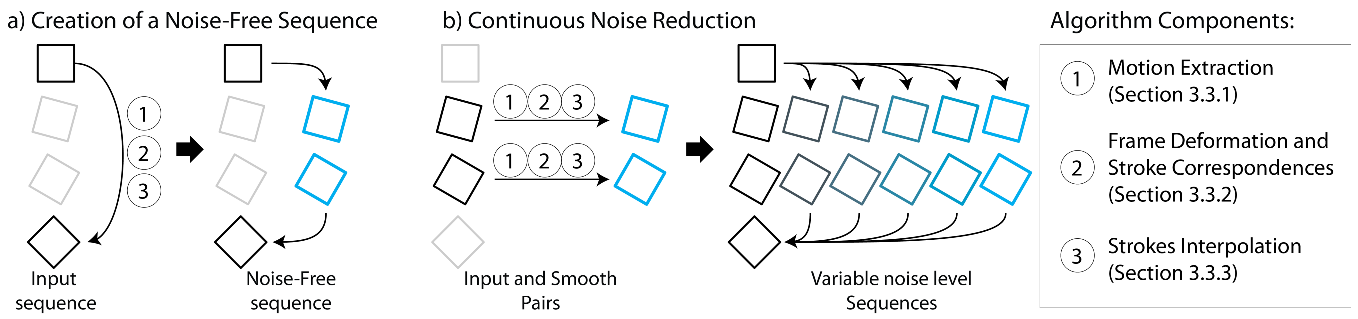
a) Creation of a Noise-Free Sequence
b) Continuous Noise Reduction
Algorithm Components:
1 Motion Extraction (Section 3.3.1)
2 Frame Deformation and Stroke Correspondences (Section 3.3.2)
3 Strokes Interpolation (Section 3.3.3)
Input sequence
Noise-Free sequence
Input and Smooth Pairs
Variable noise level Sequences
1
2
3

Figure 2: *Method Overview. Our method works in two phases. (a) The input sequence is processed to create a Noise-Free sequence. To do so, three steps are performed: (1) Motion extraction (2) Frames Deformation and Stroke-to-Stroke correspondences, and (3) Interpolation. (b) For each pair of input vs. noise-free frames, an arbitrary number of sequences with increasing noise reduction can be generated, again using the same three steps.*

Strokes that are used to define the same element in a drawing (e.g. a portion of the silhouette) can be far apart spatially from frame to frame. Similarly, strokes that represent different elements of the drawing can become spatially close. Computing appropriate stroke-to-stroke correspondences in this setting is therefore a very difficult task, since proximity is not a reliable measure of similarity. To mitigate this problem we compute the stroke correspondence measure after deforming the strokes of $\mathcal{F}_1$ according to the motion field $\mathcal{M}_{\mathcal{F}_1}^{\mathcal{F}_2}$ (see Section 3.3.1). More precisely, we compute the stroke-to-stroke correspondences between the frames $\hat{\mathcal{F}}_1 = \mathcal{D}(\mathcal{M}_{\mathcal{F}_1}^{\mathcal{F}_2}, \mathcal{F}_1)$ and $\mathcal{F}_2$. In general, the spatial distances between the strokes in the deformed version of $\mathcal{F}_1$ and $\mathcal{F}_2$ are significantly smaller, so computing stroke-to-stroke correspondences in this setting is less prone to mis-matches.

The stroke-to-stroke correspondence step results in a set of pairs (i, j), where each pair indicates a correspondence between stroke i of $\mathcal{F}_1$ and stroke j of $\mathcal{F}_2$. All pairs of corresponding strokes are then interpolated to create the inbetween frames $\mathcal{F}_t$ (see Section 3.3.3).

The remainder of this section outlines the method used to create the motion fields, the process of deforming the input frame, the criteria used to compute the stroke-to-stroke correspondences, and the stroke interpolation method.

3.3.1 Motion Extraction

An important part of our processing pipeline consists of computing the relative motion, or motion field, between two frames $\mathcal{F}_1$ and $\mathcal{F}_2$. We use a slightly modified version of the As-Rigid-As-Possible (ARAP) method described in [Sýkora et al. 2009]. The original approach assumes the mask of the registered image is known beforehand. This allows the control lattice, which represents the motion field $\mathcal{M}_{\mathcal{F}_1}^{\mathcal{F}_2}$, to be adapted to the topology variations of the underlying shape.

In our problem domain, we are dealing with a more complicated scenario, as the input consists of an unordered set of strokes without any connectivity information. However, it is still important to take into account the topology of the input sketch, as opposed to using a uniform grid. To overcome this difficulty we compute a rasterized distance field, as illustrated in Fig. 3, which we use as a mask. The computed distance field closes small gaps between the input strokes. In addition, the distance field provides a better cue for image registration (similar to *chamfer matching* [Borgefors 1988]).

In addition to the distance field, we use a hierarchical coarse-to-fine refinement. We build a multi-resolution pyramid by recursively reducing the image scale by a factor of two. Then we proceed from the coarse to fine level and run the registration algorithm with a control lattice of constant quad size. When moving up each level, we render a pixel accurate motion field to initialize the position of the control points on the finer lattice. This helps us to speed up the convergence of the motion field extraction algorithm and increases robustness under large motions. The hierarchical refinement also allows us to adapt to fine details and provide tight image registration.

As noted in [Sýkora et al. 2009] the image registration algorithm can potentially get trapped in a local optimum. These cases are typically rare but when they occur we let the user drag-and-drop selected control points in order to guide the algorithm towards a better solution. This operation can be implemented simply by fixing the position of the selected control point and changing its weight to some very large value as in [Wang et al. 2008].

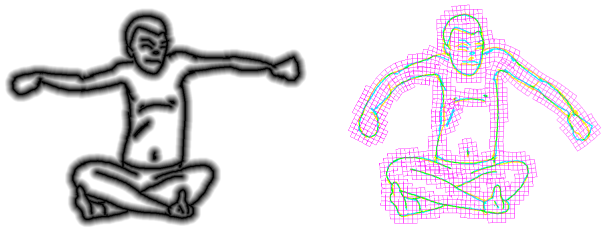

Figure 3: *Building the control lattice for the ARAP image deformation: distance field computed from rasterized strokes (left), ARAP registration using the control lattice based on the distance field (right).*

3.3.2 Frame Deformation and Stroke Matching

The motion field $\mathcal{M}_{\mathcal{F}_1}^{\mathcal{F}_2}$ represents the relative motion between frames $\mathcal{F}_1$ and $\mathcal{F}_2$. This provides a way of estimating the location of each stroke from $\mathcal{F}_1$, if it had been drawn at the time represented by frame $\mathcal{F}_2$. The computation of the deformed frame is straightforward. The vertices v (which are simply 2-dimensional points on the digital canvas) defining the strokes in $\mathcal{F}_i$ are displaced according to the motion field: $v \leftarrow v + \mathcal{M}_{\mathcal{F}_1}^{\mathcal{F}_2}(v)$.

The stroke correspondence step is used to compute a measure of how well two strokes from different frames will interpolate. Intuitively, the better aligned and spatially close the two strokes are, the better their correspondence measure should be. For our work, we define the correspondence measure between two strokes s_1 and s_2 as $h(s_1, s_2) * h(s_2, s_1)$, where $h(A, B)$ is a component of the Hausdorff distance. More precisely, $h(A, B) = max_{a \in A}(min_{b \in B}(d(a, b)))$ and $d(a, b)$ is the Euclidean distance

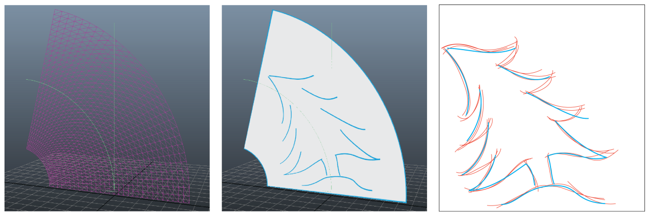

Figure 5: *Generation of the motion ground truth. A planar grid is generated and deformed with Autodesk Maya (left). A reference image is applied as texture on the mesh (center), generating a reference animation over which the final tree animation is sketched (right). The animation motion is therefore captured by the mesh deformation.*

between points a and b, i.e. the vertices of strokes A and B. We note that this is one of many choices of similarity measures [Seah and Feng 2000; Veltkamp 2001; Lie et al. 2010]. However, in our experiments, we found the Hausdorff distance to work well and be more robust than other measures due to the lack of structure in sketchy drawings.

3.3.3 Stroke Interpolation

The stroke correspondence algorithm is used to find all pairs of strokes that need to be interpolated to create the inbetween frames. We use the three-step deformation method introduced in Whited et al. [2010] to create smooth blends for each pair of strokes. The strength of this approach over other techniques [Fu et al. 2005; Baxter and Anjyo 2006; Baxter et al. 2009] is that in addition to interpolating the stroke curvatures, it also computes the global motion between pairs of strokes along logarithmic spiral trajectories, thus capturing the affinity of the global motion (rotation, translation and uniform scaling).

4 Results

We tested the effectiveness of our algorithm on hand-drawn animation sequences containing between 30 and 90 frames. In order to evaluate our motion extraction algorithm, the animation of the *Tree* shown in Figure 4 was generated procedurally (see Figure 5). All other animations were created manually with a digital drawing tool. The *Square* animation, shown in Figure 6, presents a simple case with an almost rigid motion. The use of temporal noise as an artistic tool is investigated using the *Face* animation, which is shown in Figure 11. Lastly, the *Balancing* animation, Figure 1, illustrates another challenging example handled by our framework.

For the examples in this Section, to emphasize the temporal progression, consecutive animation frames are displayed with decreasing opacity. For the *Square*, *Balancing*, and *Face* scenes, we used a uniform representative frame sampling, with a window size of 7 frames. The *Tree* scene uses a set of 8 selected keyframes and the resulting window sizes are between 4 and 7 frames. Temporal reduction values are displayed as percentages, where 0% is the input animation and 100% is the Noise-Free animation.

Our algorithm was developed in C++ and runs as a single thread. On a standard workstation, the execution of Algorithm 1 takes up to 10 seconds per pair of frames, with the motion extraction and the search for correspondences being the most time-consuming tasks. Overall, computing a full scene takes a few minutes with the unoptimized prototype.

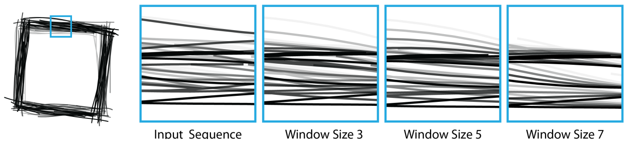

Figure 6: *Square Scene. This image shows the effect of increasing window sizes. Each close-up shows 5 animation frames overlayed with decreasing opacity. When the noise is high, the lines look evenly distributed and unstructured (Input Sequence). With low temporal noise, lines appear to follow a structure and tend to be clustered (5 and 7).*

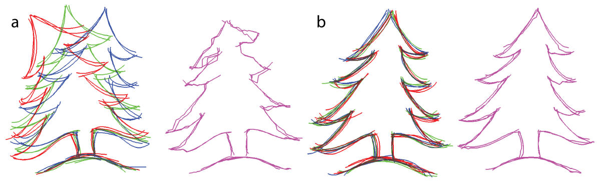

Figure 8: *Neighborhood Averaging. A window of three frames (red, green, blue) is used to compute a per point neighborhood average (purple). When the motion is large (a), the resulting frame is strongly degenerated. Even for slow animations (b), this approach leads to undesirable kinks and deformations.*

4.1 Ground Truth Comparison

To evaluate our motion extraction method we created a "ground truth" animation of the Tree in Maya. Starting with a planar textured mesh of the undeformed tree (see Figure 5), the mesh was deformed by using the bend deformer tool and keyframing the curvature. The deformed tree texture was then used as a reference over which an artist sketched each frame of the animation.

Figure 7 compares the extracted motion field with the ground truth generated in Maya. In general, the extracted motion captures both the global animation motion and local deformations due to the geometric noise. As a result, it provides a very precise stroke alignment, which greatly simplifies the task of finding stroke to stroke correspondences.

4.2 Neighborhood Averaging Comparison

We compare our results with those obtained through neighborhood interpolation, which proceeds as follows. For each sample point p_1 of a stroke j in a frame i, we collect the two nearest neighbor points p_2 and p_3 in frames $i-1$ and $i+1$. p_1 is then updated to $p_1 \leftarrow (p_1 + p_2 + p_3)/3$. An example result is shown in Figure 8. Since each sample point p_k is free to move independently, divergent attractions result in breaks of the line continuity. This usually happens when sample points of one stroke are influenced by different sets of strokes.

This result shows that a simple averaging approach is not desirable. In general, we observe that when the motion is large, this approach produces obvious artifacts, losing important features of the frame. This motivates the use of motion extraction to alleviate the effects of large animation motions. Even when the motion is small, kinks and undesirable deformations are present. This motivates the consideration of strokes as atomic entities, therefore shifting the correspondence from the individual points samples to the stroke level. Our approach benefits from both considerations.

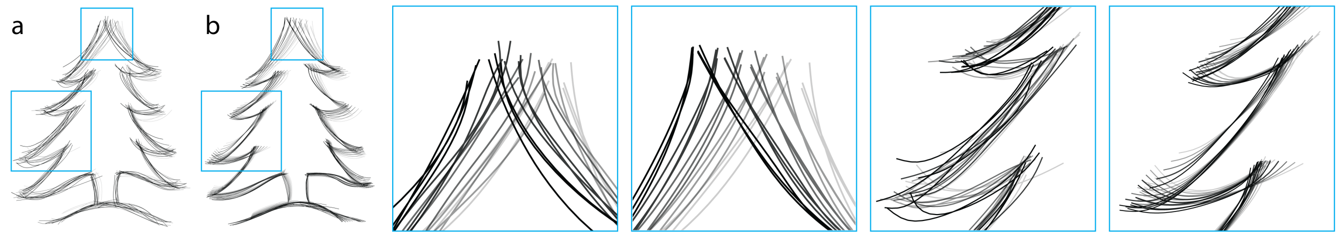

Figure 4: *Tree scene. We compare the input animation (a) with the noise free result (b). Close-ups are marked by blue squares.*

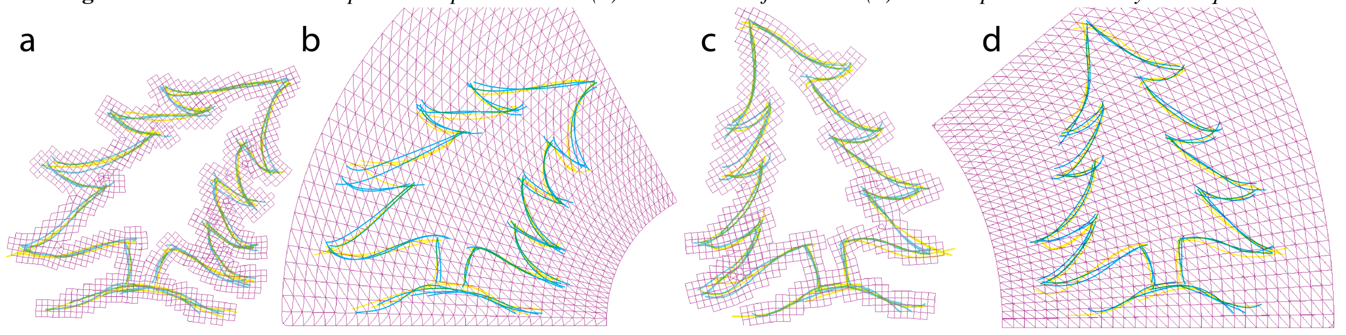

Figure 7: *Motion Extraction Comparison. The extracted motion (a and c) is compared with the ground truth motion (b and d). The blue and yellow frames represent $\mathcal{D}_i^j$ and $\mathcal{F}_j$, with $i = 22, j = 26$ (a,b) and $i = 29, j = 33$ (c,d). The extracted motion is precise and simplifies the search of stroke to stroke correspondences.*

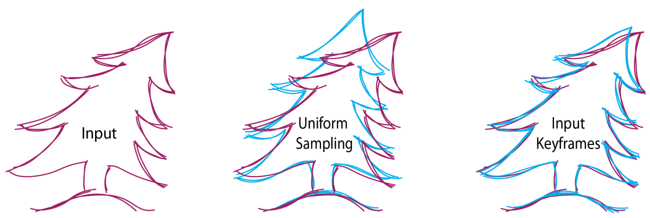

Figure 9: *Sampling Strategy. This image shows the maximum motion error (visualized as misalignment) obtained in the Tree animation using two different sampling strategies: Uniform Sampling, with a window size of 7 frames, and Keyframes, with 6 keyframes marked at the important animation times.*

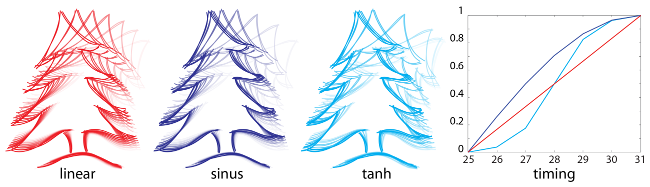

Figure 10: *Timing Control. This image shows the effect of different timing functions applied to the same set of frames.*

4.3 Sampling and Timing Control

As discussed in Section 3.2, we implemented two selection strategies for choosing representative frames, *uniform sampling*, and manually designated *keyframes*. Figure 9 compares results of using the different selection strategies. Notice that using the input animation keyframes, as opposed to uniformly sampling the frames, greatly reduces the motion error.

Figure 10 shows the effect of different timing charts applied to the same set of frames where the timing values used to generate the inbetweens in [Whited et al. 2010] are altered. Notice how the spaces are affected by the choice of the timing functions - most notably at the tip of the tree.

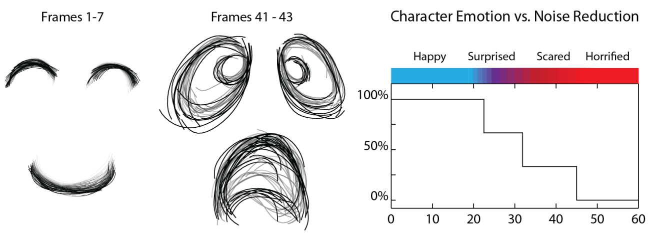

Figure 11: *Face. This scene presents a character with an emotional evolution from happy to horrified. We adapt the noise reduction to these emotions.*

4.4 Temporal Noise as an Artistic Tool

The second phase of our method allows the generation of sequences with varying temporal noise. By manipulating the noise reduction level in different parts of the animation, an artist has the ability to use the noise as an additional storytelling element. Noisy animations can be used to portray certain feelings, such as anger or fear. A proof of concept is shown in Figure 11.

5 Conclusion

We have presented a novel technique for noise reduction/manipulation in sketchy animations and demonstrated its use on a series of hand-drawn inputs. Our approach not only makes the production of larger scale sketchy animations feasible through automated noise reduction, but also widens the scope of artistic control to include noise as a first-class creative device.

5.1 Limitations and Extensions

One limitation of this work is due to the intrinsic difficulty of handling occlusions, topology changes, and disconnected components moving independently in a 2D environment. For the motion extraction step, distinct objects with different motions can create divergent motion fields and occlusions can force unnatural compressions; both cases are difficult to capture with a ARAP model. In order to handle these scenarios in complex scenes, the input would need to be first segmented into coherent layers.

Additionally, complex curved strokes that self intersect may cause problems in the correspondence and interpolation steps. A simple solution, similar to what is proposed in Barla et al. [2005], is to cut these strokes into separate pieces. However, every piece would then behave independently, which may not be the desired animation effect.

Another limitation of our work lies in the selection of keyframes. As shown in Figure 9 the set of selected representative frames has a significant impact on the output animation. In particular, as the complexity of the animation increases (i.e. higher frequency motions), a larger number of keyframes is needed to preserve the motion. This effectively limits the number of smooth inbetween frames that we can create to reduce temporal noise.

Our frame inbetweening technique needs to be extended to provide smooth interpolation across multiple keyframes. Although the interpolating motions computed by the logarithmic spiral technique [Whited et al. 2010] are smooth between consecutive keyframes, the approach has the limitation that continuity is not necessarily preserved across longer sequences. However, multiple techniques have been proposed to preserve this continuity at the keyframes, while still interpolating them. In one approach [Powell and Rossignac 2008], subdivision is used to smoothly interpolate 3D poses by blending consecutive screw motions. A more recent approach [Rossignac and Vinacua 2011] also produces continuous motions as a blending of consecutive "steady affine motions". In 2D, both of these algorithms may be applied to logarithmic spirals with little modification.

Another possible area of exploration is the use of our noise reduction technique as a post-processing operation on sketchy animations synthesized using NPAR techniques.

References

BARLA, P., THOLLOT, J., AND SILLION, F. X. 2005. Geometric clustering for line drawing simplification. In *In Proceedings of the Eurographics Symposium on Rendering*, 183–192.

BAXTER, W., AND ANJYO, K.-I. 2006. Latent doodle space. *Computer Graphics Forum 25*, 3, 477–486.

BAXTER, W., BARLA, P., AND ANJYO, K. 2009. N-way morphing for 2D animation. *Computer Animation and Virtual Worlds (proc. CASA 2009) 20*, 79–87.

BÉNARD, P., COLE, F., GOLOVINSKIY, A., AND FINKELSTEIN, A. 2010. Self-similar texture for coherent line stylization. In *NPAR 2010: Proceedings of the 8th International Symposium on Non-photorealistic Animation and Rendering*, ACM, 91–97.

BORGEFORS, G. 1988. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence 10*, 849–865.

BOURDEV, L. 1998. *Rendering Nonphotorealiztic Strokes with Temporal and Arc-length Coherence*. Master's thesis, Brown University.

COCONU, L., DEUSSEN, O., AND HEGE, H.-C. 2006. Real-time pen-and-ink illustration of landscapes. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, ACM, NPAR '06, 27–35.

CURTIS, C. 1998. Loose and sketchy animation. In *Technical Sketch SIGGRAPH 1998*, ACM.

FU, H., TAI, C.-L., AND AU, O. K.-C. 2005. Morphing with laplacian coordinates and spatial-temporal texture. In *Proceedings of Pacific Graphics 2005*, 100–102.

GRABLI, S., DURAND, F., AND SILLION, F. X. 2004. Density measure for line-drawing simplification. In *Pacific Conference on Computer Graphics and Applications*, IEEE Computer Society, 309–318.

KALNINS, R. D., MARKOSIAN, L., MEIER, B. J., KOWALSKI, M. A., LEE, J. C., DAVIDSON, P. L., WEBB, M., HUGHES, J. F., AND FINKELSTEIN, A. 2002. Wysiwyg npr: drawing strokes directly on 3d models. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '02, 755–762.

KALNINS, R. D., DAVIDSON, P. L., MARKOSIAN, L., AND FINKELSTEIN, A. 2003. Coherent stylized silhouettes. In *Proceedings of SIGGRAPH 2003*, ACM, SIGGRAPH '03, 856–861.

LIE, D., CHEN, Q., YU, J., GU, H., TAO, D., AND SEAH, H. S. 2010. Stroke correspondence construction for vector-based 2d animation inbetweening. In *Proceedings of Computer Graphics International 2010*.

POWELL, A., AND ROSSIGNAC, J. 2008. Screwbender: Smoothing piecewise helical motions. *IEEE Comput. Graph. Appl. 28* (January), 56–63.

ROSSIGNAC, J., AND VINACUA, A. 2011. Steady affine motions and morphs. *ACM Transactions on Graphics (to appear)*.

SALISBURY, M. P., WONG, M. T., HUGHES, J. F., AND SALESIN, D. H. 1997. Orientable textures for image-based pen-and-ink illustration. In *Proceedings of the ACM SIGGRAPH Conference (SIGGRAPH-97)*, ACM Press, New York, 401–406.

SEAH, H., AND FENG, T. 2000. Computer-assisted coloring by matching line drawings. *The Visual Computer 16*, 269–304.

SHESH, A., AND CHEN, B. 2008. Efficient and dynamic simplification of line drawings. *Proceedings of Eurographics, Computer Graphics Forum 27*, 2, 537–545.

SÝKORA, D., DINGLIANA, J., AND COLLINS, S. 2009. As-rigid-as-possible image registration for hand-drawn cartoon animations. In *Proceedings of International Symposium on Non-photorealistic Animation and Rendering*, 25–33.

VELTKAMP, R. 2001. Shape matching: similarity measures and algorithms. In *Shape Modeling and Applications, SMI 2001 Intl. Conference on.*, 188–197.

WANG, Y., XU, K., XIONG, Y., AND CHENG, Z.-Q. 2008. 2D shape deformation based on rigid square matching. *Computer Animation and Virtual Worlds 19*, 3–4, 411–420.

WHITED, B., NORIS, G., SIMMONS, M., SUMNER, R. W., GROSS, M., AND ROSSIGNAC, J. 2010. Betweenit: An interactive tool for tight inbetweening. In *Proceedings of Eurographics, Computer Graphics Forum*.

WILSON, B., AND MA, K.-L. 2004. Rendering complexity in computer-generated pen-and-ink illustrations. In *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering 2004, Annecy, France, June 7-9, 2004*, ACM, 129–137.

WINKENBACH, G., AND SALESIN, D. H. 1994. Computer-generated pen-and-ink illustration. In *Proceedings of SIGGRAPH 1994*, ACM, 91–100.