Flow-Guided Stylized Keyframe Animation

Mark Robinson Browning

A Thesis

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Master of Science in Engineering

Recommended for Acceptance

by the Department of

Computer Science

Adviser: Adam Finkelstein

June 2013

Abstract

We present a method that combines hand-drawn artwork with fluid simulations to produce animated fluids in the visual style of the artist's drawings. Given a fluid simulation and a set of keyframes rendered by an artist in any medium, our system produces a set of in-betweens that visually matches the style of the keyframes and roughly follows the motion from the underlying simulation. Our method builds on recent advances in patch-based regenerative morphing and image melding to produce temporally coherent sequences with visual fidelity to the target medium. Because direct application of these methods results in motion that is generally not fluid-like, we augment them to produce motion closely matching that of the underlying simulation. We demonstrate our method with animations in a variety of visual styles. We also present initial experiments that suggest that our approach holds promise for other applications, including rotoscoping video (using optical flow instead of simulated velocity fields) and providing temporal coherence for artistic image processing filters.

Acknowledgements

I am grateful to my adviser, Adam Finkelstein, and collaborator, Connelly Barnes, for their ideas, guidance, and support throughout the course of this research, and for helping to prepare the paper upon which this thesis is based. Sam Ritter created many of the beautiful drawings that appear here. Suggestions and advice from members of the Princeton graphics lab helped shape the final direction of this work.

Vimeo users FKY, Squiver, and Eugenia Loli provided the boat, stars, and jellyfish video footage, respectively, under the Creative Commons license.

Contents

1 Introduction

Recent advances in computer simulation have allowed animators to enhance 3D animations with realistic motion for smoke, water, plants, cloth, or hair that would be painstaking to achieve by hand. These elements are typically not the visual focus of the animation, instead providing secondary motion to augment that of the primary subject of the scene. Even if artists could adjust every aspect of the pose of every hair on a character, for example, their efforts would be better spent on the movement of the character itself. Thus, simulation of realistic motion for natural phenomena has come to play an important role in the 3D animation pipeline. On the other hand, its use is relatively uncommon in 2D hand-drawn animation, because it is difficult to match the rendering of simulated phenomena to artists' stylized work; realistic smoke rendered over hand-drawn fire would look wrong.

This thesis introduces a method that combines hand-drawn artwork with fluid simulations to produce animated fluids in the visual style of the artist's drawings. Of course, it is possible for artists to animate fluids like smoke and water entirely by hand: water, smoke and fire appeared in hand-drawn animations before the use of computers [28, 26, 13]. The system we introduce is therefore designed primarily to divert artists' efforts from tedious replication of natural movement for these secondary elements so that they can instead focus on the major characters.

Creating fluid animation in an artist's chosen style poses several challenges. First, the workflow should be familiar to artists and provide them with sufficient control over a broad aesthetic range. Second, the computer-generated frames need to have visual coherence with the given style. Third, the motion in the scene should be both temporally coherent and plausibly fluid-like. To address these challenges, we choose a keyframe-based approach in which the artist draws some frames "rotoscoped" to match an underlying simulation, and the computer then synthesizes in-between

frames. This approach is familiar to artists and gives them control over the choice of keyframes, style, and medium. There remains a tension between the goals of faithfully reproducing the artist's chosen style and coherent motion in the scene, which we address by using a recent patch-based morphing technique called "regenerative morphing" [24, 9]. Regenerative morphs produce transitions between pairs of keyframes by optimizing for both visual and temporal coherence. Direct application of these methods produces motion that is generally not fluid-like and has no relation to the simulation from which they keyframes have been drawn, so we augment these methods to include motion priors that guide the optimization towards the motion of the underlying fluid simulation.

Regenerative morphs with these motion priors are stylistically and temporally coherent and match the guiding flow between pairs of keyframes. Simply concatenating these morphs between successive pairs of keyframes produces animations that exactly interpolate the given keyframes and, for some inputs, are visually pleasing. In other cases, however, misregistration of the keyframes with the underlying flow and the constraint of exact interpolation can produce sequences that exhibit derivative discontinuities around the keyframes. To address this problem, we extend the notion of the two-keyframe regenerative morph to a multi-keyframe morph that approximates the intermediate keyframes and parameterizes the tradeoff between interpolation and continuity. In this extended morph, each intermediate frame depends on every other frame, making the sequential optimization used by existing methods prohibitively time-consuming even for short animation sequences. We relax the sequential dependence and demonstrate that the resulting parallel optimization does not adversely affect the results. Using the parallelized method on a cluster with as many nodes as frames in the animation sequence, the time to compute the entire sequence is constant in the number of frames.

The primary contribution of this thesis is to demonstrate that it is possible to create coherent, stylized animations of fluids, using a workflow in which the style of automatic in-betweens is derived from hand-drawn keyframes and the motion is derived from a simulation. To our knowledge, we describe the first method that could be used for this purpose. Our approach builds on recent advances in patch based morphing, offering three crucial improvements to these methods for our application: (1) motion priors based on an underlying simulation, (2) a generalization to approximating multi-keyframe morphs that avoids temporal discontinuities around keyframes, (3) a parallelization of the multi-keyframe morph that allows animation sequences to be computed with reasonable latency on a cluster. Finally, in addition to demonstrating our method with fluid animations in a variety of visual styles, we describe initial experiments that indicate promise for this approach in two other applications: rotoscoping video (using optical flow instead of simulated velocity fields) and providing temporal coherence for artistic image processing filters.

2 Related Work

Fluid simulation has been an active area of research in computer animation. Our system relies on a plausible fluid simulation, both to guide the animation towards fluid-like motion and to give the artist a starting point for drawing keyframes. Pioneering work by Stam [27] introduced "stable fluids," a method that provides physically-plausible fluid motion at interactive frame rates, which is important for artistic control in applications like ours. Treuille et al. [29] demonstrated increased artistic control with a method that approximates user-provided keyframe simulation states, but which requires an expensive global optimization. Fattal and Lischinski [11] described a more efficient local approach to computing the forces that move the fluid from one keyframe to the next. The results shown in this thesis are based on

simulations produced using Stam's stable fluids, though the approach would naturally work with other simulators.

Our work builds on a thread of research for combining computer-generated elements with hand-drawn 2D animation, as in the methods of Corrêa et al. and Petrović et al. [8, 22], which combine textures and shadows with cel animation, respectively. The work of Jain et al. [16] incorporates simulated 3D elements into hand-drawn sequences by optimizing for unknown 3D parameters such as camera and depth. Our method complements this line of research by seeking to combine the plausible motion of a simulation with the look of hand-drawn art.

Researchers have also addressed non-photorealistic rendering of fluids, including liquid [10, 31, 32], smoke [14, 21, 23], and fire [12]. Largely to handle the challenge of temporal coherence in animation, these methods tend toward a particular cartoon-like aesthetic. In contrast, our work incorporates keyframes drawn by an animator, with two main benefits: the overall style is provided by example and can span a broad range of aesthetics, and the animator controls the look of chosen keyframes.

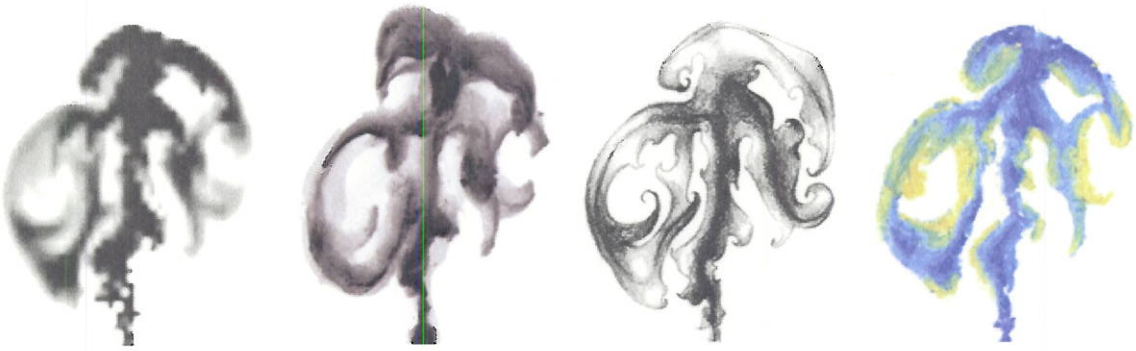
To construct in-betweens, our method synthesizes morphs between successive pairs of hand-drawn keyframes. Since their introduction by Beier and Neely [4], morphs have required substantial human intervention to establish feature correspondences and avoid ghosting artifacts [30]. Shechtman et al. [24] recently introduced regenerative morphs, which optimize over many possible feature correspondences to automatically generate smooth transitions that avoid ghosting. Darabi et al. [9] improved their method by expanding the search space and applying more sophisticated techniques to the synthesis of the in-betweens. Used without modification, these methods can provide high-quality stills in a morph between neighboring keyframes, but the apparent motion is not fluid-like. Our work builds on these methods, but restricts the motion of the corresponding features to follow the velocity field of a fluid simulation.

Researchers have developed a variety of methods to advect stylized imagery to follow an underlying flow in a scene. For example, Agarwala et al. [1] and Collomosse et al. [7] construct keyframed and temporally coherent stylized animation from video by a rotoscoping method in which drawn shapes track objects in the scene via optical flow. Bousseau et al. [6] address the problem of coherent advection of watercolor stroke texture, also following optical flow, for stylizing video. Kwatra et al. [17] address texture synthesis with temporal coherence on the surface of fluids. Recent work by Bénard et al. [5] generates example-based stylizations of 3D computer-generated animation by extending the image analogies method of Hertzmann et al. [15] to provide temporal coherence. Our approach is similar to these prior methods in that drawn features are advected to match a specified flow. The initialization step for our regenerative morph (Section 5) is similar to the bidirectional advection of Bousseau et al. Both the regenerative morphs on which we build and the animation method of Bénard et al. extend non-parametric methods for example-based synthesis to achieve temporal coherence in animation via optimization, though different goal functions are expressed. Finally, several aspects of the work of Bénard et al. rely on the source 3D geometry, which is unavailable in our setting.

While we address stylized rendering of physically plausible fluid motion, our methods could be complemented by other research on stylizing the motion itself, including the mid-level control for fluids of Barnat et al. [2], motion field texture synthesis of Ma et al. [20] and painted motion of Lockyer and Bartram [19].

3 Workflow

Our production workflow begins with a given fixed fluid simulation created for a particular shot. The main purpose of the simulation is to provide realistic motion for the construction of the in-between sequences. A secondary benefit is that it provides animators with reasonable keyframe targets, since these are known to have fluid-like



rendered simulation inkwash pen and ink oil pastel

Figure 3.1: A keyframe from the simulation of an upward-flowing jet of ink, drawn in multiple styles. Disparate features at multiple scales, including curls, sharp lines, textured gradients, and complex color blending, make many styles difficult for both the artist to draw and the morph to synthesize. Note that our method permits the artist's drawings to deviate substantially from the underlying simulation.

motion for the in-between sequences. Our experiments are based on Stam's stable fluids [27], but our method is general to any simulator that provides plausible flow. In particular, with an art-directable simulator like those of Treuille et al. [29] or Fattal and Lischinski [11], the animator could draw keyframes first, and the simulation would optimize to hit those frames, implying a somewhat modified workflow.

The animator chooses frames of the simulation to draw manually. An example of a simulated frame drawn in three different styles is shown in Figure 3.1. These drawn frames and the simulated velocity fields at each intermediate frame are supplied as input to the interpolation module, which outputs the in-between frames. If desired, the animator can then refine the output in specific places by modifying the set of keyframes until the resulting animation is satisfactory.

4 Synthesis by Regenerative Morphs

A successful keyframe interpolation will appear temporally coherent, follow the motion of the given fluid simulation, and reproduce the artist's style in each intermediate frame. These goals are difficult to satisfy simultaneously: for example,

simply advecting the keyframe pixels forwards and backwards according to the simulated flow and blending, as in Bousseau et al. [6], produces intermediates that are temporally coherent and faithful to the flow but distort the keyframe style for many kinds of artwork. This section describes recent developments in morphing and patch-based synthesis that address temporal coherence and fidelity to the keyframes. Sections 5 and 6 describe modifications to these methods that constrain the motion to match a desired flow.

To achieve both temporal coherence and fidelity to the keyframes in the interpolation, we build on two recent patch-based algorithms: regenerative morphing, introduced by Shechtman et al. [24], and its generalization via image melding, due to Darabi et al. [9]. These methods were developed based on observations about statistics in natural imagery, and while most of their applications to date have been photorealistic in nature, we find them to work well in our context and expect them to find broader use in non-photorealistic rendering.

The basis for these methods is the bidirectional similarity (BDS) measure described by Simakov et al. [25]:

$$d(S,T) = \frac{1}{N_S}\sum_{P\subset S}\min_{Q\subset T} D(P,Q) + \frac{1}{N_T}\sum_{Q\subset T}\min_{P\subset S} D(Q,P)$$

where S and T are the source and target images, N_I is the number of patches in image I, P and Q are patches in images S and T, respectively, and $D(\cdot,\cdot)$ is any patch distance measure. The first term captures completeness; it is minimized when the target contains as much information from the source as possible. The second term captures coherence; it is minimized when the target contains as few artifacts that do not appear in the source as possible.

The regenerative morph is then computed as a minimization over all interpolated frames of the following objective function:

$$E(T_{1\ldots K}, S_1, S_2) = \sum_{k=1}^{K}((1-\alpha)d(T_k, S_1) + \alpha d(T_k, S_2) + \beta d(T_k, T_{k-1}) + \beta d(T_k, T_{k+1})) \tag{4.1}$$

where S_1 and S_2 are the keyframes, $T_{1\ldots K}$ are the interpolated frames, $T_0 = S_1$, $T_{K+1} = S_2$, $\alpha = \frac{k}{K+1}$, and β parameterizes the tradeoff between the similarity of the interpolated frames to the keyframes and the temporal coherence of the sequence.

The objective is minimized as follows. First, the optimization is initialized to a crude morph at a coarse scale, typically by cross-fading between keyframes or computing nearest neighbor offsets between keyframes and constructing each intermediate frame as an average of the patches at the linearly interpolated offset locations. Then, at multiple scales, the algorithm sweeps sequentially forward and backward over the intermediate frames. For each frame, a search step computes nearest neighbor fields between the frame and its neighbors and the two keyframes using the Generalized PatchMatch algorithm [3]. Then, in a voting step, the frame is reconstructed as a weighted combination of these nearest neighbor patches. Several iterations of the search and voting steps are typically necessary for convergence. To ensure a coherent sense of motion, patch searches between adjacent frames are constrained to a small window around the location of the target patch. For additional details, refer to Darabi et al. [9] and Shechtman et al. [24].

5 Motion Priors

Regenerative morphing produces compelling transitions between keyframes, but the motion can appear arbitrary because it is guided by the nearest neighbor matches computed during the search step. In contrast, animating a fluid requires that

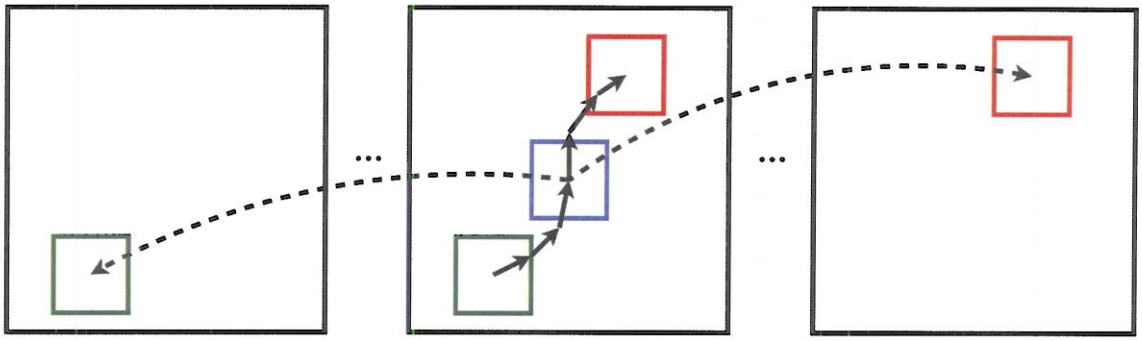


Figure 5.1: Initialization: each patch of the in-between is advected forward and backward to find correspondences in the two keyframes. The in-between is created by voting using these patches.

the apparent motion correspond to the underlying simulated flow. We satisfy this constraint by modifying the algorithm in two ways.

Initialization

First, we initialize the morph with a warp that matches the fluid flow. In a given intermediate frame, each pixel is advected forward along the simulated velocity to the second keyframe and backward to the first keyframe. Two images are constructed by voting using these neighbor fields. The initialization is produced by combining these images, as in [9], with linear blending weights according to the frame's position in the sequence. This is shown in Figure 5.1. The blue box represents a patch in an intermediate frame. The solid arrows represent the forward and backward advection paths from the blue patch. The dotted arrows indicate that the voting uses the patches in the keyframes corresponding to the forward and backward advected positions (red and green boxes, respectively).

Temporal Constraints

The second modification is to constrain the patch searches between adjacent frames according to the simulated flow. In particular, when computing $d(T_k, T_{k-1})$ and

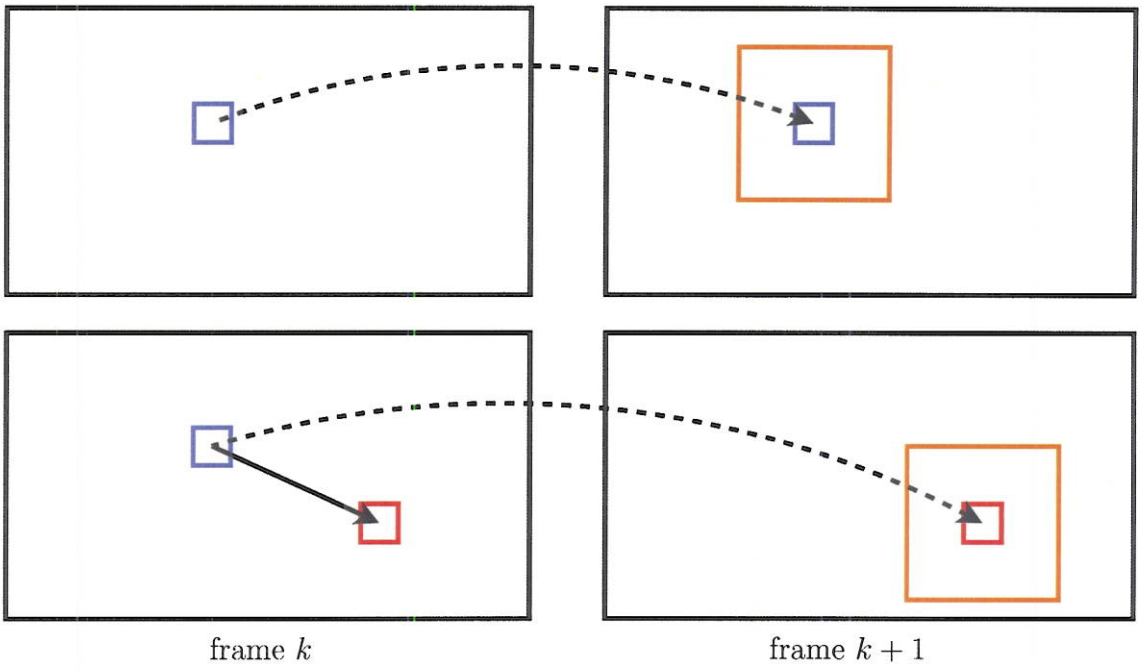


Figure 5.2: Motion priors. *Top:* in a standard regenerative morph, temporal coherence is achieved by restricting patch searches between adjacent frames to a small window. *Bottom:* in our method, these constraint windows are centered around the advected location to ensure that the motion follows the flow.

$d(T_k, T_{k+1})$, instead of constraining the search to a small fixed-size window around the location of the target patch, we constrain it to a small window around the target location after it is advected in the direction of the source image. This is shown in Figure 5.2. The blue boxes represent a patch in the current frame. Without our modification, the patch search is constrained to a small window (orange box) around this patch's location in the source image. With it, the constraint window is centered on the advected position (red box). A new parameter p specifies the width of this constraint window. The degree to which the advected constraint window aligns with the standard constraint window depends on the fluid velocity at the target position and the value of p.

6 Continuous Multi-Keyframe Morphs

For some inputs, simply concatenating successive two-keyframe sequences produced by the above method produces acceptable results that interpolate the given keyframes. For complex flows and artistic styles, however, this approach can produce discontinuous "pulsing" artifacts near the keyframes. One of the primary sources of such discontinuity is misregistration of the rotoscoped input with the underlying flow. Because each keyframe is supplied independently, and because it is difficult for the artist to have a completely coherent sense of the motion between keyframes, a region of given texture and color may not match the texture and color of the region to which it is advected in an adjacent keyframe. Thus, to preserve feature correspondences and interpolate the keyframes, the morphs of Section 5 tend to advect patches in directions that do not match the desired flow. To address this problem, we generalize the regenerative morph to approximate over multi-keyframe inputs and parameterize the tradeoff between interpolation and continuity at the keyframes.

Modified Objective Function

We generate the multi-keyframe morph by optimizing a modified objective function:

$$E(T_{1\ldots M}, S_{1\ldots N}) = \sum_{m=1}^{M}\Big(\Big(\sum_{n=1}^{N}\hat{w}(m,n)d(T_m,S_n)\Big)+ \\ +\beta d(T_m,T_{m-1})+\beta d(T_m,T_{m+1})\Big) \tag{6.1}$$

where m ranges over the entire sequence of intermediate frames $T_1 \ldots T_M$ between the first and Nth keyframes and $\hat{w}(m,n)$ is the normalized weight of keyframe n in the synthesis of frame m.

There are two differences between this objective function and (4.1). First, the optimization is performed over the entire multi-keyframe sequence. Thus intermediate

input keyframe approximated frame difference

Figure 6.1: The approximating multi-keyframe morph permits the keyframe to change to better match the flow without sacrificing fidelity to the artist's input. The difference image is the L_1 distance between the two RGB images, where white is 0 and black is the maximum difference between the two images.

keyframes are no longer constrained to match the input exactly, but instead share the soft constraints imposed on all other synthesized frames, namely, similarity in the BDS sense to the input keyframes and fidelity of the motion between adjacent frames to the given flow. By relaxing the interpolation constraint, we reduce the undesired motion required to transition from keyframe to keyframe while still respecting the overall style of the artist. The effect of approximating instead of interpolating is illustrated in Figure 6.1. The approximated keyframe is visually similar to the input keyframe, but with minor adjustments due to the temporal influence of neighboring frames.

Second, the source similarity weighting function has been generalized to include contributions from multiple keyframes. In our experiments, we found that setting $\hat{w}$ to be constant over a radius of 2-3 neighboring keyframes or using a smooth step function produced more continuous results than a simple alpha-blend of the two nearest keyframes.

Velocity Modulation

The second way in which we address discontinuities at the keyframes is based on the observation that motion that does not match the given simulation is most noticeable in regions with little underlying flow. In regions where the underlying flow is large, the extraneous motion required to preserve feature correspondences between keyframes is comparatively small. We therefore modulate the size p of the prior motion constraint at each patch location (i, j) by the magnitude of the velocity field at that location:

$$p_{i,j} = (1 - \gamma)p + \gamma f(|\mathbf{v}_{i,j}|)p \tag{6.2}$$

where $f : \mathbb{R}_{\geq 0} \to [0, 1]$ is some monotonically increasing function and γ is a parameter between 0 and 1 that controls the degree to which the modulation influences the prior size. For flows with very uneven distributions of velocity magnitudes, it may be desirable to modulate the prior size by histogram matching, for example, but for our inputs we found that

$$f(|\mathbf{v}_{i,j}|) = \frac{|\mathbf{v}_{i,j}|}{\max\limits_{i,j} |\mathbf{v}_{i,j}|}$$

worked well. When γ is close to 0, this modulation has little effect, which produces the discontinuous motion required to preserve keyframe correspondences. When γ is close to 1, the motion is more continuous, at the cost of additional fading, blurring, and approximation where the keyframes are misregistered with the underlying flow.

7 Implementation

Parallelization

To produce an animation by concatenating sequences of morphs between successive pairs of keyframes, as described in Section 5, it is trivially possible to compute each

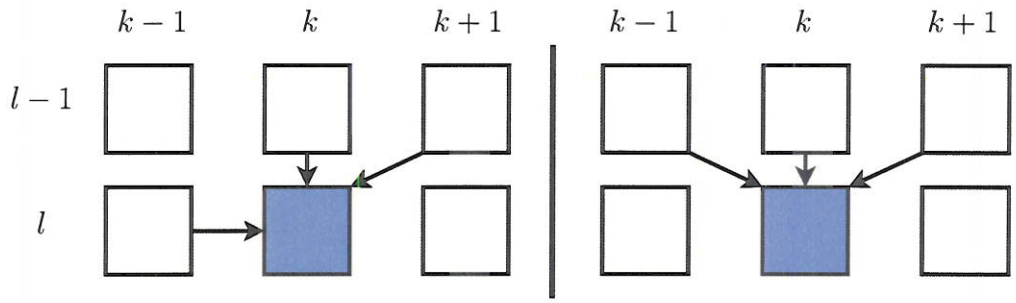


Figure 7.1: Parallelization. *Left:* in the original regenerative morphing algorithm, target frames (blue) are reconstructed by sweeping sequentially over each temporal iteration. Arrows indicate the frames that are used to reconstruct the target frame. *Right:* in our parallelized implementation, frames in a given iteration depend only on frames from the previous iteration.

parallel sequential difference

Figure 7.2: These frames demonstrate that parallelizing the morph does not noticeably degrade the quality of the results. Small differences between the frames are noticeable only upon close inspection. Difference image as in Figure 6.1.

morph independently and in parallel. In this case, due to the sequential sweeps across the frames at each scale, the time to completion depends on the number of frames between the most distant pair of keyframes. This sequential dependence is illustrated in Figure 7.1-left: during forward sweep l, frame k depends on the synthesized frame $k-1$ from the same sweep l (the direction of dependence is reversed in a backward sweep).

The continuous multi-keyframe morph developed in Section 6 results in an optimization that is global over the entire animated sequence, for which these sequential sweeps would be prohibitively slow to compute. We therefore parallelize

the implementation by breaking the dependence between adjacent frames in the same sweep, as illustrated in Figure 7.1-right. That is, in our parallel implementation, frame k in sweep l depends only on frames from sweep $l-1$ (and the keyframes). Thus our implementation constructs all frames in a given temporal iteration independently and simultaneously on a cluster.

In principle, this parallelization reduces the rate at which information can propagate between frames, but in our experiments, we found little noticeable difference between the output of the two implementations, as shown in Figure 7.2. Moreover, we found that the energy in the optimization (6.1) is consistently the same to within four significant digits.

Two-Pass Morph

The methods discussed so far each contribute to guiding the regenerative morph to match the underlying fluid simulation and to improving continuity in multi-keyframe sequences. The various combinations of these methods represent a large space of possible morphs. In our experiments, we found that the quality of results depends not only on the individual parameter settings, but also on the combination of these settings. For example, combining the approximating multi-keyframe morph with velocity modulation improves the continuity of our results, but applied independently, neither method results in comparable improvements. We found the following two-pass morph to work well across a variety of inputs.

First, we perform an initialization pass that constructs an approximating multi-keyframe morph using motion priors, as described above. The keyframe weighting function in (6.1) is the same alpha-blend used in (4.1), and the motion prior size p is not modulated by velocity. These parameters are chosen to maximize the fidelity of the reconstructed frames to the keyframes. The purpose of this pass is to produce a set of high-quality frames, which may lack continuity, for use as input to a second

"smoothing" pass. In the second pass, we treat each output frame from the first pass as a keyframe and perform the approximating multi-keyframe morph with the weighting function set to be constant over a radius of 2-3 keyframes, and modulate the prior size by velocity, as in (6.2), with γ set to 1. The purpose of this pass is to minimize the discontinuity artifacts in the first pass without sacrificing too much sharpness in the intermediates.

8 Results and Discussion

Appendix A shows selected frames from two different simulations animated in five distinct visual styles and media, demonstrating the broad range of effects that can be reproduced. Figure A.1 shows a simulation of a jet of ink animated in the styles of Figure 3.1. Figure A.2 shows frames drawn from a simulation of two opposing jets of ink that produce a complex collision. In each case, the shapes and textures at multiple scales are generally faithfully replicated in the synthesized in-betweens.

Both of these examples are drawn from sequences of 71 simulation frames. The artist drew a keyframe for every tenth frame, and the system produced nine in-betweens for every consecutive pair of keyframes. At this ratio, the artist's workload was reduced by at least a factor of ten over drawing every frame of the animation – a clear savings in mechanical labor alone. Moreover, to achieve the same level of temporal coherence entirely by hand might impose a further burden, as these drawings were made without particular regard for the previous or next keyframe. It is natural to wonder whether the artist could have drawn even fewer keyframes – every twentieth frame, for example. We performed preliminary experiments and found that results for 20-frame spans were visibly inferior to the results for 10-frame spans shown here. These experiments were not exhaustive in the large space of visual styles and optimization parameters, so it is plausible that longer sequences might be

made to work. Regardless, we found the 10:1 ratio to be a good compromise between visual fidelity and labor savings.

For some styles, we experimented with a range of patch sizes (5,7,9,11) and motion prior sizes (1,3,5,7). We found the method to be robust across this range, producing different but acceptable results in most cases. The patch size influences the performance of the optimization (doubling patch size roughly doubles computation time) and sets the approximate scale of features that are matched via the bidirectional similarity metric. Larger patches more faithfully replicate the features of the keyframes, at the cost of a coarser match to the guiding flow. The size of the motion prior similarly trades off between fidelity to the visual style of the keyframes and fidelity to the fluid motion: larger prior sizes allow the patch search to find better matches, at the cost of more extraneous motion. All of the fluid results shown here use patch size 5 and prior size 5 at image resolution 400×400. For this patch size, with square frames dimension 200, 400, 600 and 800 pixels, the algorithm runs for about 10, 30, 100 and 240 minutes, respectively, on a heterogeneous cluster in which a 2.8Ghz 4-core AMD processor is typical. These timings match a cubic dependence in the linear dimension of the image. Note that since our algorithm is parallelized by frame, these times are constant in the number of frames M in the full sequence, provided M is less than the number of nodes in the cluster.

We found that the optimization energy in equation (4.1) was consistently lower using our flow-guided morphs than the regenerative morph of Darabi et al. [9]. We believe that this is because the given flow is a good prior for plausible motion between the rotoscoped keyframes.

While the methods developed here improve significantly on the results obtained by naïve application of previous algorithms, some visual artifacts remain. The in-betweens sometimes fail to reproduce fine-scale, high-contrast aspects of textures, such as the fine stippling of the the pen and ink style shown in Figure 3.1. These

differences are often hard to notice in animation, but can clearly be seen in the stills. It is possible that morphs run at higher resolution with larger patch sizes would address this problem, at the cost of increased computation. Also, when the morph fails to find good correspondences, it often resorts to fading in (or out) some feature that an artist might have been able to treat more gracefully by hand. This might be addressed by more sophisticated optimizations that find even better patch correspondences. In our experiments, the improvements of Darabi et al. [9] to the search, sampling, and reconstruction methods used in Shechtman et al. [24] yielded dramatically better results, so we expect that further improvement in this direction will prove fruitful. Finally, we note that in some animations the regenerative morph produces a faint, diffuse halo around high-contrast areas. We believe that this is a result of the gradient weighting in the image melding approach, and could probably be removed by changing parameter settings (or more easily by straightforward post-production).

9 Extensions

In developing the keyframe interpolation methods described in earlier sections, our goal was to produce plausible motion for stylized animations of fluids. More generally, however, our inputs are simply a set of 2D keyframes and motion fields that describe gross motions between them. We therefore speculate that our method may be suitable for a variety of other inputs, including rotoscoped video with optical flow and rotoscoped 3D animation with flow derived from the underlying geometry. This section describes preliminary experiments with stylization of video using both hand-drawn artwork and generic image processing filters from Adobe Photoshop. The results suggest that this method may be a viable approach for stylizing video, though several challenges remain to be addressed in future work.

Rotoscoped Video

First, we consider the workflow described in Section 3, instead starting from video. Rather than velocity fields from a simulation, we rely on optical flow computed using the approach of Liu [18]. The artist rotoscopes a few keyframes from the video, and we apply the morphing methods of Sections 4-6 to produce the in-betweens. Figure B.1 and Figure B.2 show example frames from two videos. Our method produces good results for keyframes spaced 10-20 frames apart. In the example in Figure B.1, the boat translates between keyframes with little distortion, and the fluid motion of the waves is conveyed in the drawn water, especially in the specular highlights. Similarly, the stars in the night sky in Figure B.2 rotate convincingly. Two primary artifacts are visible. First, near occlusion and disocclusion boundaries, edges are prone to jumping forward or lagging behind, which is due to discontinuities in the optical flow causing patches to be advected across the boundaries. For videos from which high-quality optical flow can be extracted, this problem can be mitigated by initializing the morph using small (1-3 pixel) patches to reduce overlap at the boundaries, but we anticipate that more sophisticated techniques, such as the occlusion indicators used in [5], would further improve our results. Second, discontinuities in the optical flow of the output near keyframes are sometimes apparent. The techniques employed in the smoothing pass discussed in Section 7 can be applied here as well, though in our experiments they proved less effective with these inputs than with fluids.

Filtered Video

Another extension of our method is in the application of single-frame image filters to video. The naïve approach of independently filtering each frame of the video generally produces temporally incoherent results due to randomness and sensitivity to the input. Using the multi-keyframe approximating morph developed in Section 6, however, we

are able to construct a temporally coherent sequence, in which each frame is similar to the original filtered frames in the BDS sense. In particular, we compute the morph by treating every nth filtered frame as a keyframe. For larger values of n, the method is a special case of the rotoscoping application discussed above, and the results share that application's advantages and disadvantages. Smaller values of n reduce artifacts associated with optical flow, but produce more strongly conflicting constraints on the intermediate frames. For highly-structured filters, such as the chalk and charcoal filter shown in Figure B.4, this can produce some "swimming" artifacts, as the structural elements are forced to shift to accommodate the constraints of nearby keyframes. Nevertheless, this approach can generate results similar to the "watercolorization" method of Bousseau et al. [6], while their method would fail for filters such as chalk and charcoal.

10 Conclusions and Future Work

This thesis demonstrates that it is possible to create coherent, stylized animations of fluids by using a workflow in which the style and motion of automatic in-betweens are derived from hand-drawn keyframes and a simulation, respectively. Our results would have been arduous and time-consuming to produce by hand, and perhaps even impossible to animate manually with a similar degree of temporal coherence. We demonstrate that our method works on a broad range of artistic styles that would be difficult to reproduce programmatically.

The approach we describe is artist-friendly in that it relies on a familiar keyframe-based workflow. Of course, the interpolating morph described in Sections 4 and 5 gives the artist complete control of the look at the keyframes. The approximating scheme of Section 6 provides nearly the same level of control, because it tends to synthesize frames that are only slightly different from the corresponding input keyframes. Finally, while the simulation is useful as a guide for the artist, the hand-

drawn frames can nevertheless deviate from it without substantially degrading the quality of the results, thereby supporting flexibility of composition.

As mentioned in Section 1, there is tension between temporal coherence, fidelity to the medium, and fidelity to the simulated motion. We have cast this problem as an optimization in which the objective captures temporal coherence and fidelity to the medium and the optimization procedure is constrained to search for a local minimum that respects the simulated motion. Our output should exhibit as much of each these desirable qualities as possible, but it is impossible to compare our results against some fundamental limit, since the weight of each of these qualities is a matter of preference that depends on the style and context of the animation. Thus, while our results improve upon simpler methods, it is difficult to estimate how much room for improvement remains. Also, as mentioned in Section 8, the maximum length of a morph that can be obtained by these methods while maintaining acceptable results remains to be seen. Again, this is likely to depend on the style and context of the animation.

We have used simulations based on Stam's stable fluids [27], but we believe our method would nicely complement simulations that focus on artistic control, such as those of Treuille et al. [29] and Fattal and Lischinski [11], which would necessitate a modified workflow. Moreover, while we have focused on 2D fluid simulations, we believe that the broad ideas presented in this work could be adapted to other kinds of simulations, including 3D fluid, hair, and cloth. A major challenge would be to address depth and occlusion changes in this context.

Finally, we believe that this work demonstrates the first application of regenerative morphs in non-photorealistic rendering, and that this general technique is likely to be broadly applicable in areas that involve stylization and artistic control. Section 9 presents preliminary results that indicate promise for stylization of video. We believe that the method may also extend to 3D keyframe animation.

A Fluid Output

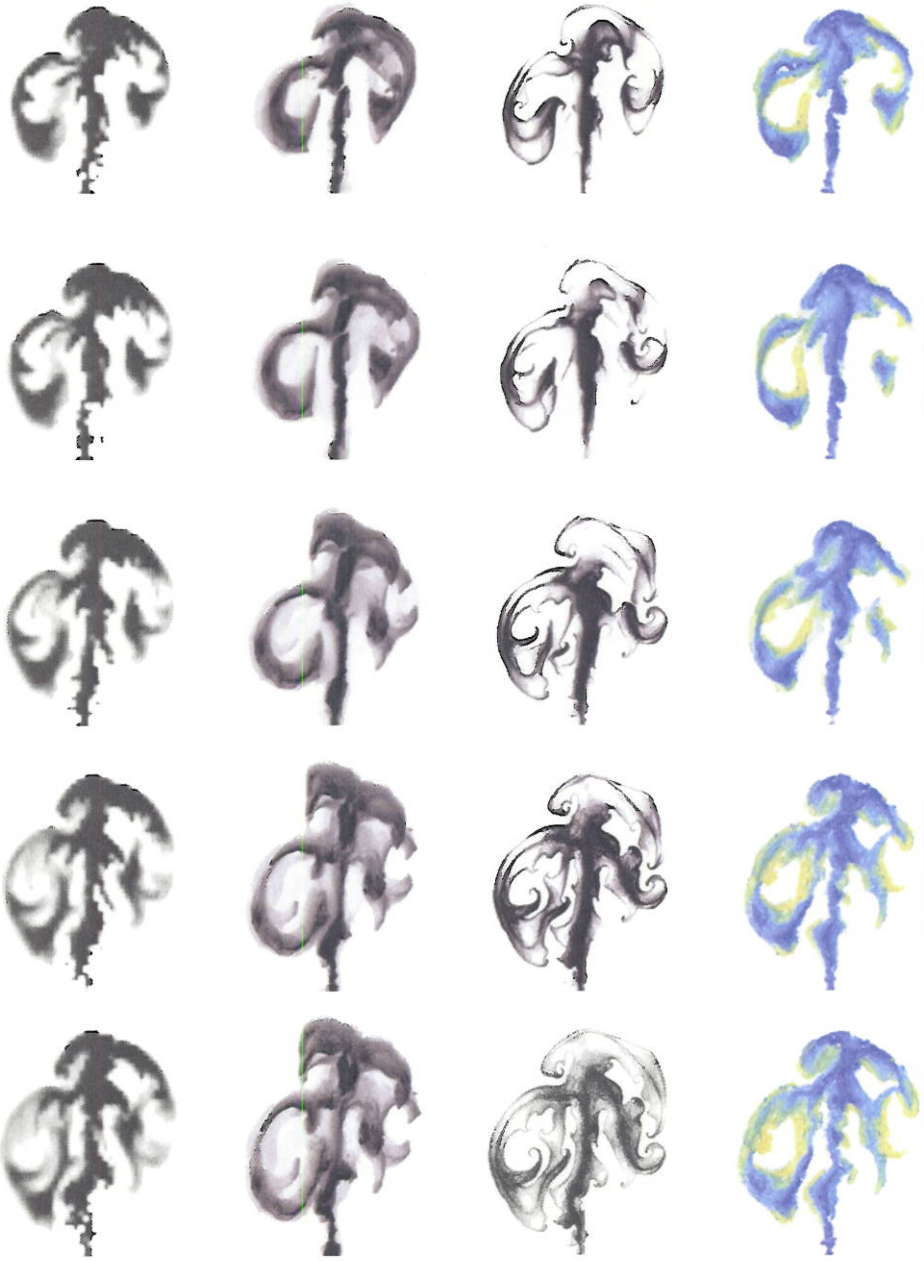


Figure A.1: A sequence of synthesized frames in the styles shown in Figure 3.1. Every second frame shown.

Figure A.2: A simulation of opposing jets of ink (left) rendered in two new styles. Every second frame shown. The interaction at the interface between the opposing jets of ink creates interesting blending effects.

B Rotoscoped and Filtered Video Output

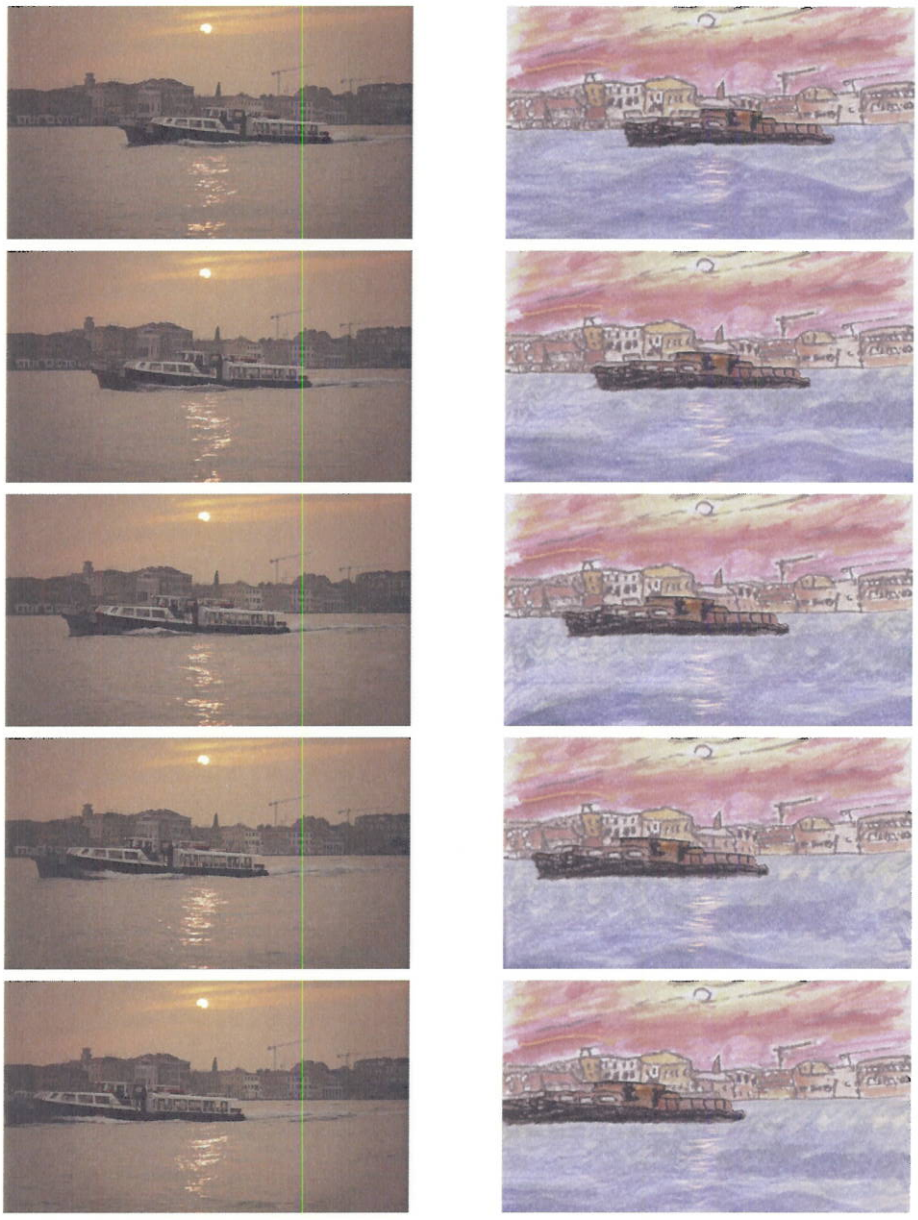


Figure B.1: Video of a boat rotoscoped in a complex style. Every eighth frame shown. When animated, the specular highlights move convincingly according to to the motion of the waves.

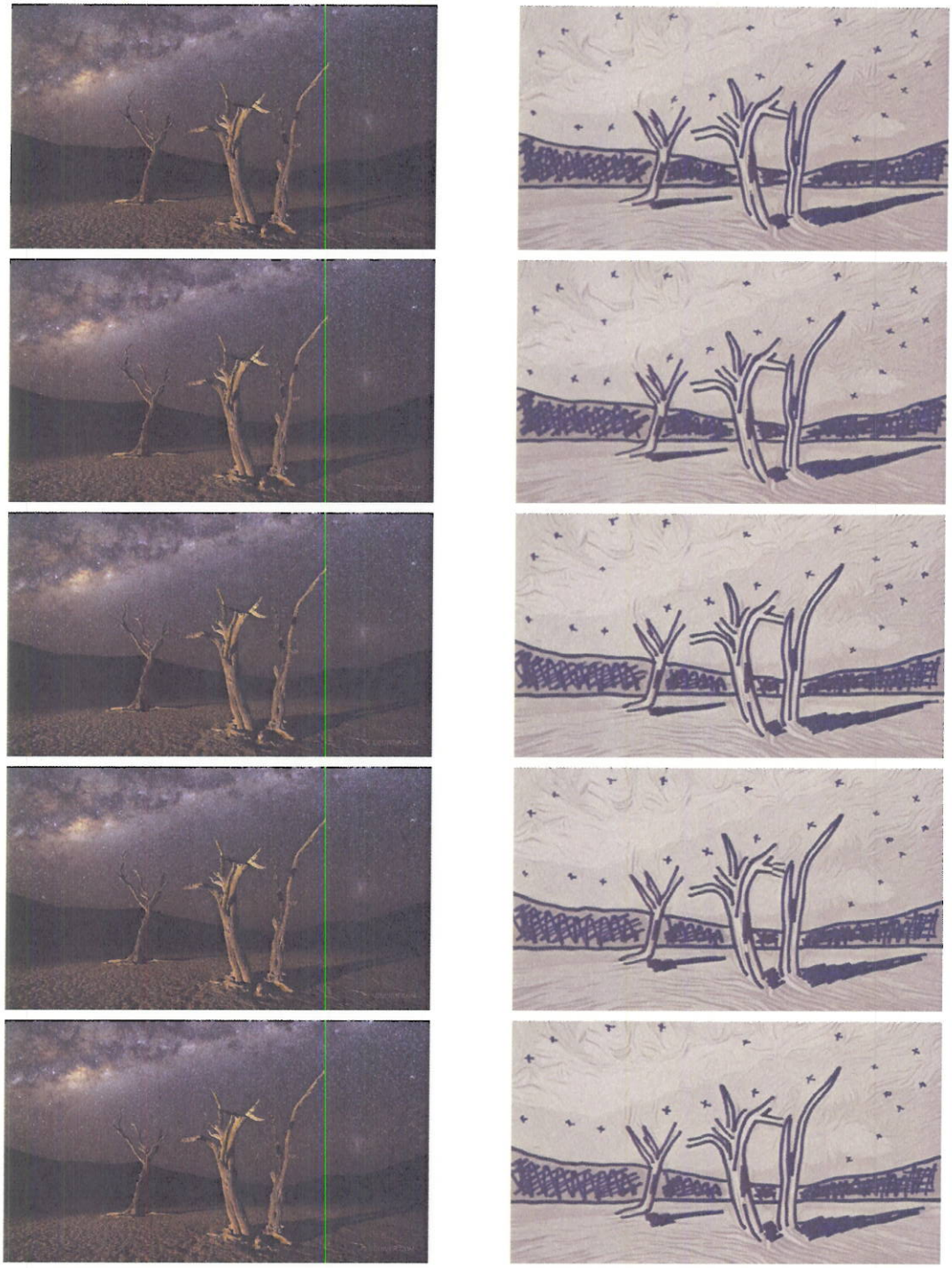


Figure B.2: Time-lapse footage rotoscoped in a simple style. Every eighth frame shown. Synthesized frames preserve the artist's style and exhibit correct motion, as in the rotation of the stars, for example.

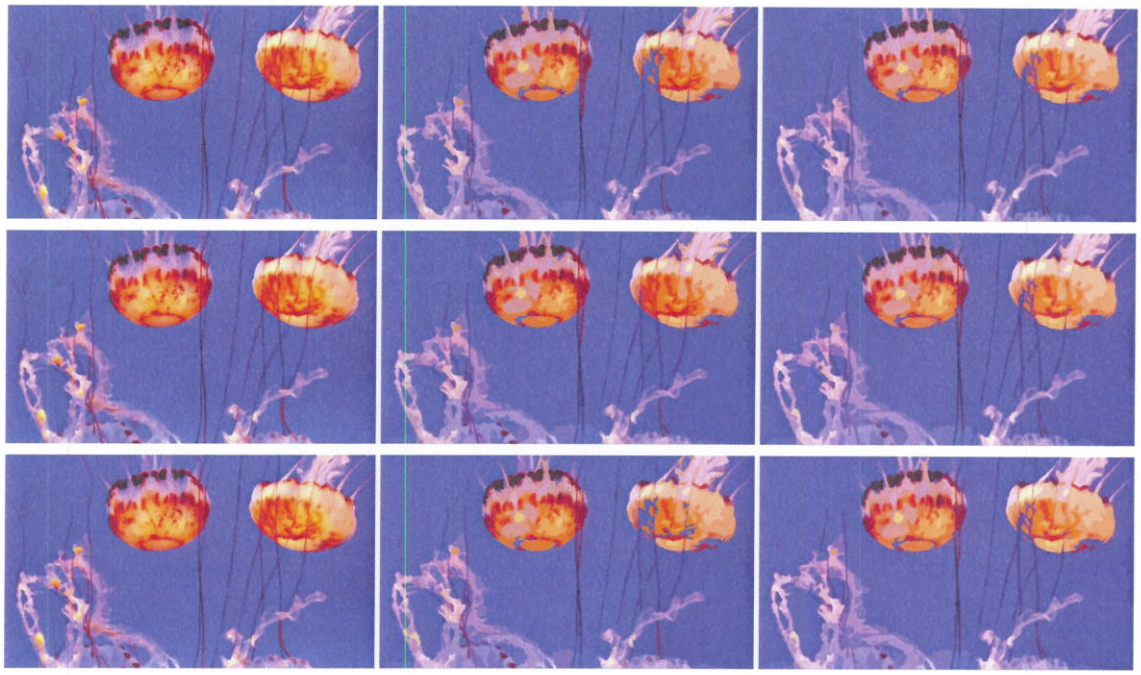


Figure B.3: *Left:* unfiltered video frames. *Middle:* each frame filtered independently using Adobe Photoshop's cutout filter. *Right:* filtered frames after applying our method. Our method changes each frame to better match its neighbors while preserving the effect of the filter, which can be seen in the blue regions in the filtered versions of the upper right jellyfish.

Figure B.4: *Left:* a frame from the video in Figure B.2 filtered using Adobe Photoshop's chalk and charcoal filter. *Right:* the same frame after applying our method. The style of the filter is preserved while improving temporal coherence, even for highly-structured filters such as this.

Bibliography

[1] Aseem Agarwala, Aaron Hertzmann, David H. Salesin, and Steven M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph.*, 23(3):584–591, 2004.

[2] Alfred Barnat, Zeyang Li, James McCann, and Nancy S. Pollard. Mid-level smoke control for 2d animation. In *Proceedings of Graphics Interface 2011*, GI '11, pages 25–32, 2011.

[3] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized PatchMatch correspondence algorithm. In *European Conference on Computer Vision*, September 2010.

[4] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '92, pages 35–42, 1992.

[5] Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. Stylizing Animation By Example. *ACM Transactions on Graphics*, 32(4), July 2013.

[6] Adrien Bousseau, Fabrice Neyret, Joëlle Thollot, and David Salesin. Video watercolorization using bidirectional texture advection. *ACM Trans. Graph.*, 26(3), July 2007.

[7] John P. Collomosse, David Rowntree, and Peter M. Hall. Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):540–549, September 2005.

[8] Wagner Toledo Corrêa, Robert J. Jensen, Craig E. Thayer, and Adam Finkelstein. Texture mapping for cel animation. In *Proceedings of SIGGRAPH 98*, pages 435–446, July 1998.

[9] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image Melding: Combining Inconsistent Images using Patch-based Synthesis. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2012)*, 31(4), 2012.

[10] A.M. Eden, A.W. Bargteil, T.G. Goktekin, S.B. Eisinger, and J.F. O'Brien. A method for cartoon-style rendering of liquid animations. In *Proceedings of Graphics Interface 2007*, pages 51–55. ACM, 2007.

[11] Raanan Fattal and Dani Lischinski. Target-driven smoke animation. *ACM Trans. Graph.*, 23(3):441–448, August 2004.

[12] Fabian Di Fiore, Johan Claes, Fabian Di, Fiore Johan, Claes Frank, Frank Van Reeth, and Limburgs Universitair Centrum. A framework for user control on stylised animation of gaseous phenomena, 2004.

[13] Joseph Gilland. *Elemental Magic, Volume I: The Art of Special Effects Animation.* Focal Press, 2009.

[14] Haitao He and Duanqing Xu. Real-time cartoon animation of smoke. *Computer Animation and Virtual Worlds*, 16(3-4):441–449, 2005.

[15] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 327–340, 2001.

[16] Eakta Jain, Yaser Sheikh, Moshe Mahler, and Jessica Hodgins. Three-dimensional proxies for hand-drawn characters. *ACM Trans. Graph.*, 31(1):8:1–8:16, February 2012.

[17] Vivek Kwatra, David Adalsteinsson, Theodore Kim, Nipun Kwatra, Mark Carlson, and Ming C Lin. Texturing fluids. *Visualization and Computer Graphics, IEEE Transactions on*, 13(5):939–952, 2007.

[18] Ce Liu. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis.* PhD thesis, Massachusetts Institute of Technology, May 2009.

[19] Matt Lockyer and Lyn Bartram. The amotion toolkit: painting with affective motion textures. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, CAe '12, pages 35–43, 2012.

[20] Chongyang Ma, Li-Yi Wei, Baining Guo, and Kun Zhou. Motion field texture synthesis. *ACM Trans. Graph.*, 28(5):110:1–110:8, December 2009.

[21] M. McGuire and A. Fein. Real-time rendering of cartoon smoke and clouds. In *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, pages 21–26. ACM, 2006.

[22] Lena Petrović, Brian Fujito, Lance Williams, and Adam Finkelstein. Shadows for cel animation. In *Proceedings of ACM SIGGRAPH 2000*, pages 511–516, July 2000.

[23] A. Selle, A. Mohr, and S. Chenney. Cartoon rendering of smoke animations. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 57–60. ACM, 2004.

[24] Eli Shechtman, Alex Rav-Acha, Michal Irani, and Steve Seitz. Regenerative morphing. In *IEEE Conference on Computer VIsion and Pattern Recognition (CVPR)*, San-Francisco, CA, June 2010.

[25] Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. Summarizing visual data using bidirectional similarity. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008.

[26] Charles Solomon. *Enchanted Drawings: The History of Animation.* Random House, 1994.

[27] Jos Stam. Stable fluids. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 121–128, August 1999.

[28] Frank Thomas and Ollie Johnston. *Disney Animation: The Illusion of Life.* Abbeville Press, 1987.

[29] Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. Keyframe control of smoke simulations. *ACM Trans. Graph.*, 22(3):716–723, July 2003.

[30] G. Wolberg. Image morphing: a survey. *The visual computer*, 14(8):360–372, 1998.

[31] M. You, J. Park, B. Choi, and J. Noh. Cartoon animation style rendering of water. *Advances in Visual Computing*, pages 67–78, 2009.

[32] J. Yu, X. Jiang, H. Chen, and C. Yao. Real-time cartoon water animation. *Computer Animation and Virtual Worlds*, 18(4-5):405–414, 2007.