The 8th ACM/EG Expressive Symposium
C. Kaplan, A. Forbes, and S. DiVerdi (Editors)

EXPRESSIVE 2019

Video Motion Stylization by 2D Rigidification

Johanna Delanoy [1], Adrien Bousseau[1] and Aaron Hertzmann[2]

[1]Inria Sophia Antipolis - Méditerranée, Université Côte d'Azur
[2]Adobe Research

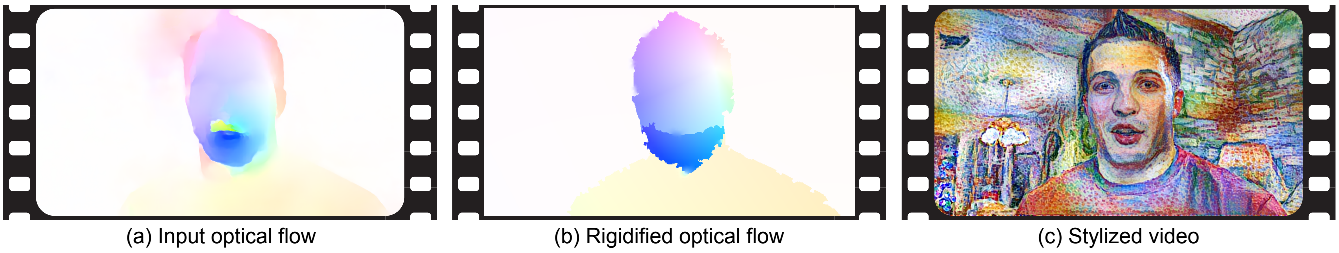


Figure 1: *Our method takes as input a video and its optical flow (a). We segment the video and optimize its pixel trajectories to produce a new video that exhibits piecewise-rigid motion (b). The resulting rigidified video can be stylized with existing algorithms (c) to produce animations where the style elements (brush strokes, paper texture) produce a strong sense of 2D motion.*

Abstract

This paper introduces a video stylization method that increases the apparent rigidity of motion. Existing stylization methods often retain the 3D motion of the original video, making the result look like a 3D scene covered in paint rather than a 2D painting of a scene. In contrast, traditional hand-drawn animations often exhibit simplified in-plane motion, such as in the case of cut-out animations where the animator moves pieces of paper from frame to frame. Inspired by this technique, we propose to modify a video such that its content undergoes 2D rigid transforms. To achieve this goal, our approach applies motion segmentation and optimization to best approximate the input optical flow with piecewise-rigid transforms, and re-renders the video such that its content follows the simplified motion. The output of our method is a new video and its optical flow, which can be fed to any existing video stylization algorithm.

CCS Concepts

• ***Computing methodologies*** → ***Non-photorealistic rendering;*** *Motion processing;*

1. Introduction

The goal of *video stylization* is to give a video the look of having been created with an artistic medium, such as oil painting or watercolor. Past research in non-photorealistic animation has worked hard to ensure "temporal coherence", generally taken to mean avoiding flickering artifacts, while also following optical flow [HE04, Lit97, HP00, BCK*13, BNTS07, OH12, SED16, RDB18]. We believe that some of the most recent methods have become *too successful* at it: too much temporal coherence creates the uncanny and unappealing effect of a 3D world covered in paint, rather than of a painting of a 3D world (*e.g.* [SED16, RDB18]). Some previous works have injected noise into animation in the quest for a more hand-made look [FLJ*14, KP11, FJS*17]. Our work explores a different avenue – inspired by traditional cut-out and multi-plane animation – to create *motion* that looks hand-drawn, rather than being *too faithful* to the input.

We complement existing methods by introducing *motion rigidification*, which consists of deforming a video so that its motion becomes as piecewise-rigid as possible in image space. When advected along our modified optical flow, style elements undergo 2D rigid transforms and uniform scaling rather than tracking 3D trajectories. We enforce similarity (rotation, translation, and scaling) rather than strict rigidity, since scaling is necessary to model objects that move away or toward the observer. The resulting stylized videos exhibit a very "2D look;" this look is reminiscent of traditional cut-out animations like *Charlie and Lola* and *Village of Id-*

iots where objects are animated by moving their parts rigidly from frame to frame, and by replacing the parts when they deform significantly.

Our approach is inspired by the work of Breslav et al. [BSM*07], who hypothesize that, since style elements are traditionally drawn in 2D, they should move in 2D to preserve their hand-drawn appearance. However, their approach was limited to a very specific type of texture-mapped 3D rendering. In addition, while their method changes the motion of the stylization texture, it keeps the underlying object unchanged, which yields motion discrepancies at silhouettes. We build on their approach and generalize it to arbitrary videos and to any stylization algorithm that takes an optical flow as input.

Our solution includes three main components:

- A motion segmentation algorithm that decomposes a video into near-rigid pieces. Users can control the segmentation with scribbles, for instance to capture motions that are subtle yet contribute to the intended story.
- A motion optimization algorithm that warps pixel trajectories such that they form as-rigid-as-possible segments while deviating as-little-as-possible from the original trajectories.
- A video re-rendering algorithm that synthesizes a video whose motion conforms with prescribed pixel trajectories. The output of our method is thus a new video aligned with its rigidified optical flow, which can be used as input to any video stylization method.

We demonstrate the effectiveness of our method by rigidifying videos with complex motions (animals, humans, natural scenes), which we subsequently stylize with a recent by-example style transfer algorithm [GEB16, RDB18].

2. Related work

Video stylization has been an active topic in Non-Photorealistic Rendering for more than two decades [Lit97, Mei96], as surveyed by Bénard et al. [BBT11], Kyprianidis et al. [KCWI13], and Rosin and Collomosse [RC13]. We first discuss the main approaches to stylize videos, before discussing related methods on motion estimation and processing.

Video stylization. The earliest methods for stylized animation targeted oil painting, where individual brush strokes are clearly visible [Mei96, Lit97]. The most common strategy to produce such a style consists in distributing brush strokes to cover the first frame of the animation, moving the strokes to the next frame using optical flow, and removing or adding strokes to avoid overlaps and gaps [Lit97, HP00, HE04]. These approaches have later been extended to styles like watercolor by advecting [BNTS07] or filtering [KP11] a stylization texture from frame-to-frame. Recent methods employ by-example texture synthesis to handle an even wider range of styles [BCK*13, BBRF14, SED16, FJS*17, RDB18]. These methods cast the synthesis as a global optimization that strives to reproduce the appearance of an exemplar while maintaining temporal coherence along optical flow. However, enforcing temporal coherence too strictly results in rather artificial results, which motivated Fišer et al. [FLJ*14, FJS*17] to inject randomness in the synthesis to mimic the temporal noise of traditional animations. Our work is largely complementary to all these methods, since our goal is not to improve how the stylization follows the video motion, but rather to modify that motion to look more hand-drawn.

Our approach follows the idea of Breslav et al. [BSM*07], who stylizes 3D animations using 2D patterns that approximate object motion with similarity transforms. Their main idea and results are highly inspirational, but the specific approach they took has numerous limitations. In particular, their method only applies to textured 3D models, with predefined texture segmentation. In addition, their method does not modify the motion of the underlying 3D objects, resulting in visible sliding of the patterns along silhouettes where the input and modified motion differ significantly. Our approach addresses these limitations to produce rigidified videos that are compatible with a large body of existing video stylization methods.

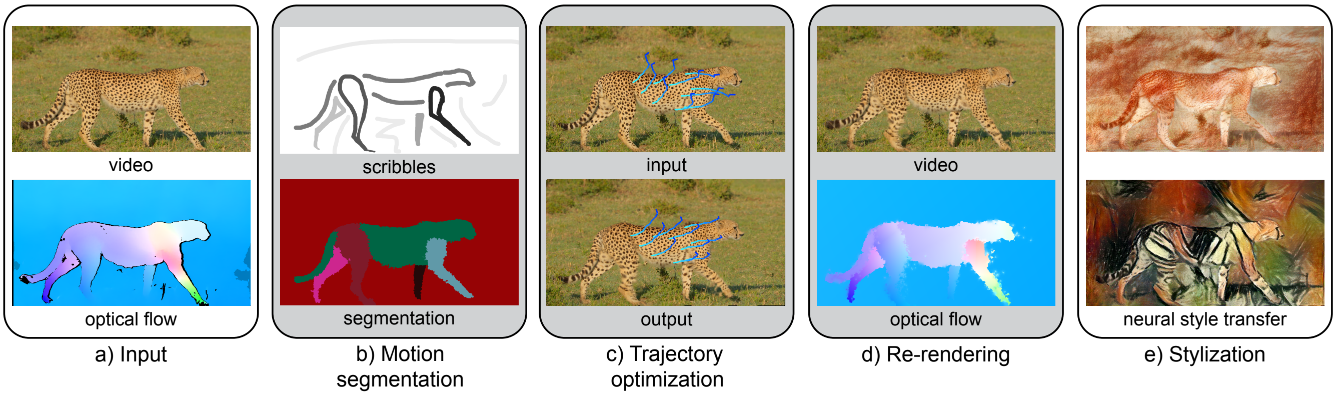
Our approach is inspired by the 2D motion produced by traditional animation techniques, such as paper cut-out. Barnes et al. [BJS*08] described an animation system dedicated to this technique, where users animate characters made of one or several rigid parts. Each part is rendered with a constant texture, which mimics traditional animations where the same piece of paper is moved from frame to frame. In contrast, we designed our method to best preserve the original appearance of the input video, including temporal variations of texture and shading within each rigid piece. We leave the choice of abstracting away such variations to the subsequent stylization algorithms that can be applied on our output.

Motion estimation and processing. While we aim at simplifying motion in a video, several methods aim at *magnifying* motion [LTF*05, WRS*12, WRDF13]. In particular, our method follows the main processing steps of Liu et al. [LTF*05] – motion segmentation, motion modification, and video re-rendering. However, our implementation of the two first steps differs. For motion segmentation, Liu et al. group correlated trajectories, while we group pixels that follow the same similarity transforms. For motion modification, Liu et al. simply apply a scaling factor on the motion vectors, while we optimize for new trajectories that are as-rigid-as-possible while staying close to the input. Closer to our application domain, Collomosse et al. [CRH05], Lee et al. [LYKL12] and Wang et al. [WDAC06] magnify motion in videos to reproduce the classical “squash and stretch” effects of cartoon animations, while Dvorožňák et al. [DLKS18] transfer these effects from a hand-drawn exemplar. Finally, local rigidity has been used as a regularizer in optical and scene flow computation [VSR13, YL15], which is complementary to our goal of modifying the video to achieve rigid motion.

3. Overview

Given a video and its optical flow, our goal is to generate a new video and optical flow such that

- The new video is composed of large segments that follow similarity transforms from frame to frame,
- The pixel trajectories in the new video are close to the pixel trajectories in the original video,
- The new video and its optical flow are well aligned.



a) Input

b) Motion segmentation

c) Trajectory optimization

d) Re-rendering

e) Stylization

Figure 2: *Overview of our method. Given an input video and its optical flow (a), we first employ interactive segmentation to decompose the video into parts that approximately move rigidly (b). We then optimize pixel trajectories such that they remain close to the input trajectories, while being as rigid as possible (c, magnified for visualization). The optimized trajectories are then used to re-render the video and its optical flow (d), which can serve as input to any existing stylization algorithm (e).*

We address these objectives as three separate computational steps — motion segmentation, trajectory optimization, and video re-rendering (Figure 2).

We first cast motion segmentation as a labeling problem, where pixels of a frame receive the same label if their optical flow is approximated well by the same similarity transforms (Section 4). Our formulation includes spatial and temporal smoothness terms to favor the emergence of large segments that move coherently during multiple frames. Since the number and shape of the segments greatly impact the outcome of our method, we provide artistic control on this step by means of user scribbles.

However, while the similarity transforms found for each segment only introduce subtle deviations from the original optical flow, accumulating these deviations over multiple frames would yield significant drift of the video content. We address this issue in a second step, where we track pixels along extended sequences and optimize the resulting trajectories to best satisfy the local rigid motion while minimizing global drift (Section 5).

Our last step consists in warping the video according to the displacement of the optimized trajectories. Note that, unlike conventional methods that estimate optical flow *from* a given video, this step entails generating a new video that follows the given flow. Once re-rendered, our rigidified video is ready to be processed by any existing video stylization algorithm.

4. Motion segmentation

The first step of our method takes as input a video and its optical flow and segments it into parts such that, for each frame of the video, the optical flow within each part is well approximated by a similarity transform. A similarity transform $\mathbf{S}$ is composed of a rotation matrix $\mathbf{R}$, a translation vector $\mathbf{t}$, and a uniform scaling s. We formulate this segmentation as a labeling problem, where each label $\ell \in \mathcal{L}$ is associated with a series of similarity transforms over all frames, $\{\mathbf{S}_\ell^t = (\mathbf{R}_\ell^t, \mathbf{t}_\ell^t, s_\ell^t)\}_{t\in(1\ldots T)}$. The output of this step is a spatio-temporal label map, which assigns each pixel of each frame to one of the labels, each of which has an associated similarity transform (Figure 2b). We use a fixed number of labels, specified by the user scribbles (Section 4.2).

Since the optical flow of real-world videos is often inaccurate at objects boundaries, we achieve more precise segmentations by complementing the per-frame motion models with a color model for each segment. We use a Gaussian Mixture Model (GMM) with 5 Gaussians to represent the color distribution of a label over all frames, with scale, mean and variance parameters denoted as $(\alpha_\ell^g, \mu_\ell^g, \sigma_\ell^g)_{g\in(1\ldots5)}$.

We first discuss how to evaluate the quality of a given configuration of unknowns, before explaining how we find high-quality configurations using an optimization algorithm that iterates between assigning pixels to labels, and updating the motion and color parameters of each label given their assigned pixels. While we describe our algorithm in terms of pixels, we detail at the end of Section 4.3 how we accelerate the optimization by working on superpixels.

4.1. Energy formulation

In what follows, we denote $I^{t\in T}$ the input video frames and $\mathbf{f}^t$ the optical flow from frame t to the next. Each pixel i in frame t has an initial position $\mathbf{u}_i^t$, such that $\mathbf{u}_i^{t+1} = \mathbf{u}_i^t + \mathbf{f}_i^t$.

We define the quality of a given labeling L with an energy composed of two terms. The first term measures how well the similarity transforms and Gaussian Mixture Model of a label approximate the optical flow and color of pixels assigned to that label. For a given pixel i, frame t and label ℓ, we express the term as

$$\begin{aligned} E_{\text{fit}}(\mathbf{u}_i^t, \ell) \;=\; & \left|\left|(\mathbf{u}_i^t + \mathbf{f}_i^t) - (\mathbf{R}_\ell^t s_\ell^t \mathbf{u}_i^t + \mathbf{t}_\ell^t)\right|\right|^2 \\ & - w_{\text{color}} \log\Big(\sum_g \alpha_\ell^g G(I^t(\mathbf{u}_i), \mu_\ell^g, \sigma_\ell^g)\Big), \qquad (1) \end{aligned}$$

where G denotes the normal distribution and w_{color} balances the contribution of the motion and color models.

The second term encourages large, uniform segments by penalizing the assignment of different labels to neighboring pixels that share similar colors and motion

$$E_{\text{smooth}}(\mathbf{u}_i, \mathbf{u}_j, \ell_i, \ell_j) = \delta(\ell_i \neq \ell_j)\left[\exp\left(-\frac{D_c(i,j)}{2\beta_c}\right) + \exp\left(-\frac{D_f(i,j)}{2\beta_f}\right)\right], \quad (2)$$

where $D_c(i,j) = ||I(\mathbf{u}_i) - I(\mathbf{u}_j)||^2$ measure the color difference and $D_f(i,j) = ||\mathbf{f}_i - \mathbf{f}_j||^2$ the optical flow difference between pixels i and j. The indicator function $\delta(\ell_i \neq \ell_i)$ equals 1 when the labels ℓ_i and ℓ_j differ, 0 otherwise, and the exponential decreases quickly as the color and motion differences increase. We follow Rother et al. [RKB04] to compute the weights β_c and β_f as the average color and optical flow differences of the video, computed over all pixel neighborhoods. In practice, we evaluate E_{smooth} on a spatio-temporal neighborhood to also encourage temporal coherence of the segmentation, as detailed in Section 4.3.

We balance these two terms with dedicated weights to obtain the energy of a given labeling L over the entire video sequence

$$\begin{aligned} E_{\text{segment}}(L) = & \sum_i \sum_t w_{\text{fit}} E_{\text{fit}}(\mathbf{u}_i^t, \ell_i^t) \\ & + \sum_{j \in \mathcal{N}_i^t} w_{\text{smooth}} E_{\text{smooth}}(\mathbf{u}_i^t, \mathbf{u}_j, \ell_i^t, \ell_j), \end{aligned} \quad (3)$$

where ℓ_i^t denotes the label assigned to pixel i in frame t, and $\mathcal{N}_i^t$ denotes its spatio-temporal neighborhood.

4.2. User guidance

The energy formulation outlined above solely measures the quality of a segmentation based on geometric criteria (fitness and smoothness). However, the quality of a segmentation also often depends on artistic goals. For example, users may want to approximate background objects with a single segment, yet decompose a foreground object in several pieces to better capture subtle motions. Similarly, users may choose to segment each leg of an animal separately to prevent one of the legs to appear "fixed" to the body, even if that leg only moves slightly. In addition, the segmentation algorithm can be sensitive to errors in the optical flow or to low-contrast object boundaries. We enable user control and correction by incorporating scribbles in our segmentation algorithm. Each scribble is assigned a color that represents a label, such that pixels scribbled with the same color should end up in the same segment, while pixels scribbled with a different color should be in separate segments. We achieve this behavior by over-writing the fitting term on scribbled pixels

$$E_{\text{fit scribble}}(\mathbf{u}_i^t, \ell) = w_{\text{scribble}} \delta(\ell \neq \ell_s) \quad (4)$$

with ℓ_s the label of the scribble. The weight w_{scribble} balances the strength of the user annotations against the other terms of the optimization.

The different scribble colors implicitly define the set of labels $\mathcal{L}$ considered by the optimization. We also experimented with automatic segmentation and a variable number of labels, using a so-called *label cost* to encourage the use of as few labels as possible [DOIB12]. However, we achieved our best results with user guidance. In practice, we only require users to provide scribbles in a few keyframes of their choice, and we propagate these scribbles over the entire video by tracking the scribbled pixels until they get occluded.

4.3. Optimization

The energy we defined depends on two sets of variables – the assignment of pixels to labels, ℓ_i^t, and the similarity transforms and Gaussian Mixture Models associated to each label, parameterized by $\mathbf{S}_\ell^t$ and $(\alpha_\ell^g, \mu_\ell^g, \sigma_\ell^g)$ respectively. We solve for values of these variables that approximately minimize $E_{\text{segment}}(L)$ using the PEARL algorithm, which is a general optimization method for multi-model fitting [IB12, DOIB12]. In a nutshell, the algorithm alternates between assigning observations to labels using a fixed set of models, and updating the model parameters of each label to best fit the observations assigned to it (Algorithm 1). At each iteration, the assignment of labels is performed using the α-expansion algorithm[†]. We performed 3 such iterations for all results, which was sufficient to converge in our experiments. The main challenge in applying PEARL in our context is to properly initialize and update the model parameters to capture the complex motion of real-world objects over multiple frames.

Algorithm 1 PEARL algorithm [DOIB12] applied to our motion segmentation problem.

```
1: Initialize similarity transforms and GMMs for the set of label candidates $\mathcal{L}$
2: Run $\alpha$-expansion to compute the optimal labeling $L$ according to $E_{\text{segment}}(L)$ (Equation 3), using fixed label candidates $\mathcal{L}$
3: Update the similarity transforms and GMMs of the label candidates $\mathcal{L}$ to best fit the optical flow and color distribution within each segment of $L$
4: Goto 2
```

Initializing the motion and color models. Since each scribble color corresponds to a unique label, we initialize the Gaussian Mixture Model and similarity transforms of each label from its scribbled pixels tracked along the video. Given a set of such scribbled trajectories, we use the least-squares formulation described by Breslav et al. [BSM*07] to fit a similarity transform on the optical flow displacements within each frame, and use the *OpenCV* [Bra00] implementation of Gaussian Mixture Models to fit a color distribution on the colors gathered from all frames. Finally, when the trajectories of the scribbled pixels start after the first frame of the video, or end before the last frame, we initialize the similarity transforms of the missing frames with the transforms obtained at the closest frames.

[†] Code for multi-label segmentation available at https://vision.cs.uwaterloo.ca/code

Updating the motion and color models. Each labeling iteration of the PEARL algorithm forms segments by assigning pixels to labels. Our goal is then to use the optical flow and color values of each segment to update the motion and color models of the corresponding label. However, a given label may only occur in a subset of the video frames; while each label needs a color and motion model in every frame to be used as candidates for the next labeling iteration. Our solution is to extend the segment to other frames by tracking each of its pixel along the forward and backward optical flows. We then update the model parameters using the same least-squares and GMM fitting as for the initialization. Finally, in the event where all pixel trajectories of a segment end before reaching some of the frames, we update the similarity transforms of such frames with the transforms obtained at the closest frames.

Implementation details. In practice, we accelerate the evaluation of $E_{\text{segment}}(L)$ by computing the labeling on a graph of superpixels rather than on the pixel grid. As a downside, working with superpixels reduces temporal coherence of the segmentation since superpixels are computed in each frame independently. We use the average color and optical flow values over superpixels to compute the color and motion difference terms D_c and D_f, and consider two superpixels to be spatial neighbors if they share a boundary, and temporal neighbors if they are connected by at least one optical flow vector. We also introduce a weight on the term E_{smooth} for temporal neighbors according to the number of optical flow connections they share, $w_{\text{temporal}}(i,j) = \frac{\sum_{p\in\mathcal{S}_i}\sum_{q\in\mathcal{S}_j}\delta_{\mathbf{f}}(p,q)}{\min(|\mathcal{S}_i|,|\mathcal{S}_j|)}$, with $\mathcal{S}_i$ and $\mathcal{S}_j$ neighboring superpixels and $\delta_{\mathbf{f}}(p,q)$ equals 1 when pixel p and q are connected by the optical flow, 0 otherwise. Finally, we consider that a superpixel is covered by a scribble if 25% of its pixels are covered by that scribble.

Our implementation is based on the Flownet 2.0 optical flow algorithm [IMS*17] and on SEEDS superpixels [VdBBR*12]. We detect occlusions by checking the consistency of the forward and the backward flow, as described by Sundaram et al. [SBK10] (Equation 5 in their paper).

5. Trajectory optimization

Our segmentation algorithm recovers one similarity transform per segment, per frame. However, applying these transforms in sequence results in significant drift, as approximation errors accumulate from frame to frame. The second step of our approach is to optimize pixel trajectories over the video to best reproduce the similarity transforms found at each frame, while keeping pixels close to their original trajectories. As an additional benefit, balancing rigidity of the output with fidelity to the input offers a continuum of solutions, ranging from the original video all the way to a highly rigidified video.

Our approach starts by tracking pixels along the video optical flow to create their trajectories. We adopt a greedy scheme where we start a trajectory for every pixel of the first frame, and then for every pixel of subsequent frames that is not traversed by any existing trajectory. We repeat this process in reverse order, starting from the last frame and progressing towards the first. These two passes over the video provides us with a large set of, occasionally redundant, trajectories. We then order the trajectories by length and select them one by one until all pixels of the video are traversed by at least one trajectory. We end up with N trajectories $\mathbf{U}_{i=1\cdots N}$, each tracking a pixel $\mathbf{u}_i$ over a continuous subset of the frames, *i.e.* $\mathbf{U}_i = (\mathbf{u}_i^{t-m}, ..., \mathbf{u}_i^t, ..., \mathbf{u}_i^{t+n})$. We next optimize for new trajectories $\hat{\mathbf{U}}_i$ according to two energy terms.

The first term measures the deviation of each trajectory from the similarity transforms of the segments it traverses, similar in spirit to as-rigid-as-possible energies used for image and surface deformation [SMW06, SA07]

$$E_{\text{rigid}}(\hat{\mathbf{U}}_i) = \sum_t \left|\left|\hat{\mathbf{u}}_i^{t+1} - (\mathbf{R}_\ell^t s_\ell^t \hat{\mathbf{u}}_i^t + \mathbf{t}_\ell^t)\right|\right|^2 \tag{5}$$

where the sum runs over all frames of the trajectory, and we use the shorthand $\ell = \ell_i^t$ for clarity.

The second term measures the deviation of each trajectory from its original position

$$E_{\text{anchor}}(\hat{\mathbf{U}}_i) = \sum_t \left|\left|\hat{\mathbf{u}}_i^t - \mathbf{u}_i^t\right|\right|^2. \tag{6}$$

Combining the two terms gives an energy over all trajectories

$$\begin{aligned} E_{\text{trajectories}}(\hat{\mathbf{U}}) &= \sum_{i=1\cdots N} w_{\text{rigid}} E_{\text{rigid}}(\hat{\mathbf{U}}_i) \\ &+ \ w_{\text{anchor}} E_{\text{anchor}}(\hat{\mathbf{U}}_i). \end{aligned} \tag{7}$$

After optimization, we generate the output optical flow $\hat{\mathbf{f}}_i^t$ by splatting the vector $(\hat{\mathbf{u}}_i^{t+1} - \hat{\mathbf{u}}_i^t)$ for each pixel along each optimized trajectory. Similarly, we generate a warping field $\mathbf{w}_i^t$ by splatting the vector $(\mathbf{u}_i^t - \hat{\mathbf{u}}_i^t)$, which we will use to render a new video aligned with the optimized optical flow (Section 6). Since the optimized trajectories may not traverse all pixels of the output, we diffuse the splatted values to empty pixels.

Implementation details. Equation 7 corresponds to a linear least-squares energy, which we minimize by solving the corresponding sparse linear system using Eigen [GJ*10]. Like with the segmentation, we speed-up computation and improve robustness to noise by performing the above optimization over superpixels rather than pixels, where we select trajectories such that each superpixel is traversed by at least one trajectory. However, since this strategy results in a much sparser set of trajectories, diffusing the splatted optical flow and warp vectors produces blurry vector fields. We address this issue by first generating a new segmentation $\hat{L}$, where we assign to each superpixel the most frequent label among the trajectories traversing that superpixel. We then use this segmentation to stop the diffusion at borders between superpixels of different labels. Note that $\hat{L}$ is only a proxy for the segmentation of the output video, since the superpixels are computed on the input rather than on the unknown output. Nevertheless, we found that this approximation improves results compared to using the input segmentation, or no segmentation at all.

6. Rendering

We are now equipped with a rigidified optical flow $\hat{\mathbf{f}}_i^t$, along with a warping field $\mathbf{w}_i^t$ that indicates how to distort the input frames

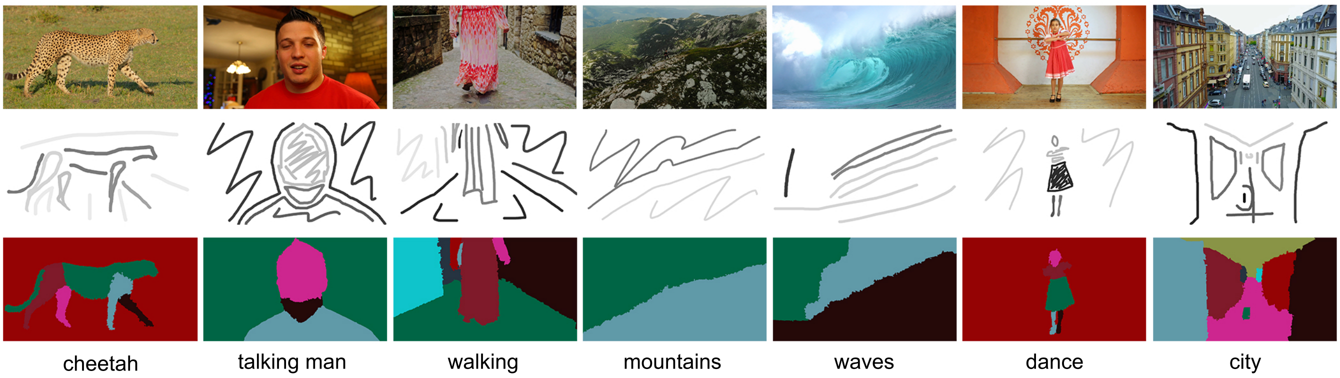


Figure 3: *Example scribbles and motion segmentation for each sequence in our results.*

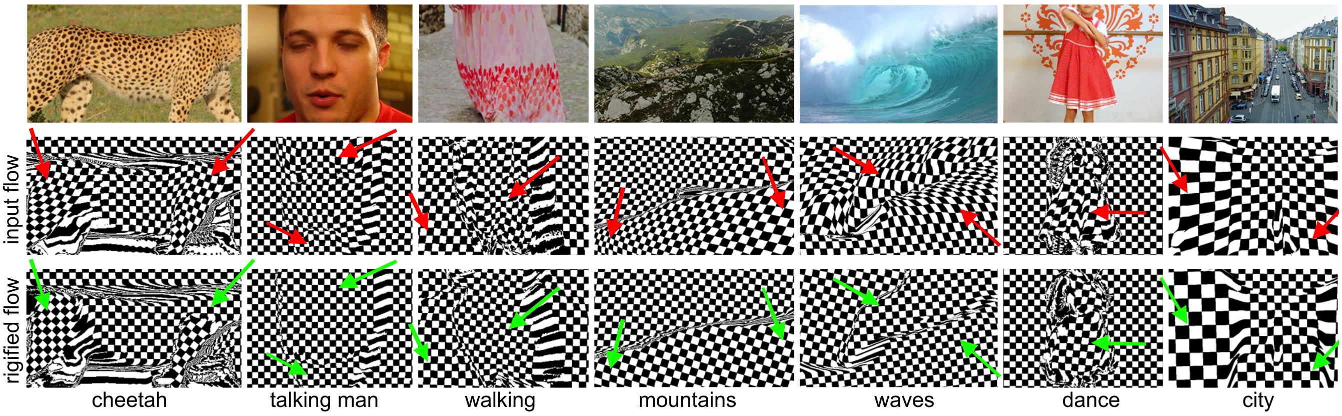


Figure 4: *Advecting a checkerboard texture along the video quickly reveals distortions due to non-rigid motion (middle row). Our method better preserves the shape of the checkerboard pattern (bottom row). See our supplemental materials for animated versions of this visualization.*

to align them with the new flow. Specifically, we render each new frame $\hat{I}^t$ by looking up, for each pixel $\hat{\mathbf{u}}_i^t$, the color of pixel $\hat{\mathbf{u}}_i^t + \mathbf{w}_i^t$ in the original frame I^t.

7. Results

We applied our approach on videos with varied motion, including deformable animals and characters (walking cheetah, talking man, walking woman, dancing girl), fluids (waves), and out-of-plane motion (camera rotating around a mountain or following a street). Figure 3 shows one frame for each of these sequences, along with user scribbles and the resulting motion segmentation.

Our results demonstrate several different effects, which were produced as a function of the input video and the user scribbles that we provided. For example, we assigned the jaw of the talking man to a different segment than the remaining of his face, which results in a cut-out motion similar to how Canadians are animated in *South Park*. We also purposely separated the legs of the cheetah from its body to achieve a puppet-like animation, or the head of the dancing girl from her torso for a similar effect. Our method also applies to non-articulated objects, such the depth layers of the mountain sequence, or the ground and walls of the walking sequence. In such cases, our method approximates rigid out-of-plane motions by 2D translations and scaling, as is traditionally done in multi-plane cell animation. Finally, the wave sequence illustrates an extreme case of non-rigid motion. Our method approximates the complex motion as a series of simple ones, which results in visible discontinuities at motion borders. These discontinuities can be attenuated by reducing the weight w_{rigid} in Equation 7.

Figure 4 visualizes the rigidity of an output video by showing how a checkerboard texture evolves as it is advected along its optical flow. Note how the squares of the checkerboard retain their shape in successive frames, while they quickly distort when advected along the original video, revealing the 3D shape of the underlying objects. The only distortions that remain visible in our results occur at disocclusions, where our implementation of texture advection stretches the texture to cover the gaps.

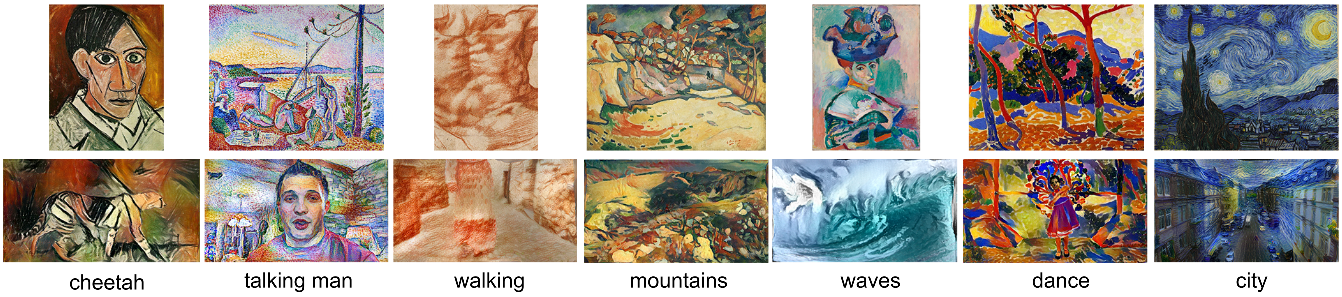


Figure 5: *We use neural style transfer [GEB16, RDB18] to render videos in various styles. We only transferred the luminance for the* waves *sequence.*

We strongly encourage readers to look at our accompanying videos to judge the effect of our method during animation. In particular, while we provide the intermediate warped videos as supplemental material, the benefit of our approach is best appreciated on side-by-side comparisons between stylizations of original sequences and stylizations of our rigidified versions. For stylization, we use the method by Ruder et al.[‡] [RDB18], which incorporates temporal coherence constraints to the successful neural style transfer algorithm of Gatys et al. [GEB16]. We used this approach to transfer the style of famous painters, as illustrated in Figure 5.

Limitations. Our focus in this work is on the style of motion, rather than on automated video analysis. Our method is sensitive to errors in the input optical flow, which can impact the motion segmentation and optimization. We used extra scribbles to both indicate the desired style and to correct such errors. The optical flow and segmentation algorithms also produce ragged object boundaries, which creates artifacts in the warped video. However, these artifacts are largely hidden in the final stylized result. Given the dizzying pace of advances in computer vision at present, we believe that it should be easy to considerably improve these aspects of our method.

The term E_{anchor} of our trajectory optimization typically results in a warping field of small magnitude, which makes our simple image warp sufficient in most cases. Nevertheless, stretching or fold-over artifacts can occur in areas where two segments move in opposite directions by several pixels, as shown in Figure 6. A potential solution to this limitation would be to assign a depth order to each segment and in-paint holes that appear between segments or along image borders using texture synthesis, as done by Liu et al. for motion magnification [LTF*05].

Parameter settings and timings. All our videos have a resolution of around 800 × 450 pixels, which we segmented into 5600 superpixels, each covering around 60 pixels. We kept all parameters fixed for our tests. In particular, we used $w_{\text{color}} = 0.01$ to balance the color and optical flow terms of the segmentation, $w_{\text{fit}} = 80$, $w_{\text{smooth}} = 15$, $w_{\text{scribble}} = 10000$ to treat scribbles as hard constraints, $w_{\text{anchor}} = 1e-6$ and $w_{\text{rigid}} = 1$ to achieve a near rigid output. We also experimented with smaller values of w_{rigid}, but the resulting effects were too small to be noticeable.

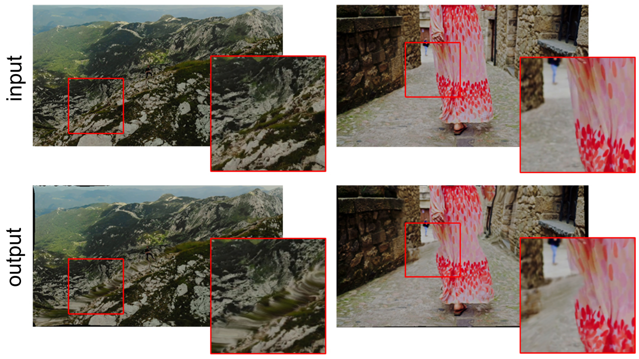


Figure 6: *Our simple image warp can produce stretching (left) or fold-over artifacts (right) in the presence of strong displacement between neighboring segments. User-provided depth ordering and in-painting would be needed to handle such cases.*

Table 7 details the time spent for each step of our method, for each of our results on a desktop computer equipped with an Intel Xeon E5-2630 CPU (20 cores) and 48GB of memory . The most expensive part is the motion segmentation, which takes around 4 seconds per frame on average. Significant time is also spent on diffusing the optical flow and warp vectors to all pixels (around 2 seconds par frame), which could be greatly accelerated by using a GPU solver.

8. Conclusion

Despite decades of research in non-photorealistic rendering, motion stylization has received significantly less attention than appearance stylization. As a result, while stylization algorithms can now make individual frames look much like paintings, stylized videos often move too realistically compared to traditional hand-drawn animations. Often, they make the world appear covered in paint. Un-

[‡] https://github.com/manuelruder/artistic-videos

Sequence	# frames	# scribbled keyframes	# labels	segmentation (s)	optimization (s)	diffusion (s)
cheetah	230	6	6	1050	57	529
talking	280	7	4	842	45	375
walk	250	4	8	1375	108	866
mountains	200	1	2	423	26	319
dance	200	10	6	937	33	649
waves	180	4	3	589	40	549
street	220	5	9	1438	38	361

Table 1: *Timings for some of our results. The computational cost of the segmentation is roughly linear with the number of frames and labels, while the remaining steps are linear with the number of frames. Motion segmentation dominates the cost, followed by the diffusion of optical flow and warping vectors.*

fortunately, there are very few reference points in traditional animation to inspire innovative algorithms. Most hand-painted and hand-drawn animations use very simple strategies for creating motion, such as redrawing every frame, or moving paper cut-outs. Hence, to some extent, every non-photorealistic animation algorithm creates a new style of motion.

In this spirit, our goal in this paper is to explore a new style of motion, based on identifying problems with existing motion styles, and taking inspiration from the traditional cut-out animation style as well as from the seminal work by Breslav et al. [BSM*07]. Our output videos produce a strong sense of 2D motion, as if individual parts were moved around in the image plane. Since our method outputs a new video and its optical flow, it is compatible with any existing stylization algorithm.

Our method employs motion segmentation to decompose the video into near-rigid parts, and we found that different segmentations of the same video can result in very different stylizations, which motivated us to provide user control on this step. In the future, we believe that a combination of better computer vision algorithms and new design principles inspired by real-world animations could help automate motion simplification.

Acknowledgements

This research was supported by research and software donations from Adobe. The authors are grateful to Inria Sophia Antipolis - Méditerranée "Nef" computation cluster for providing resources and support.

References

[BBRF14] Browning M., Barnes C., Ritter S., Finkelstein A.: Stylized keyframe animation of fluid simulations. In *Symposium on Non-Photorealistic Animation and Rendering - NPAR '14* (2014).

[BBT11] Bénard P., Bousseau A., Thollot J.: State-of-the-art report on temporal coherence for stylized animations. *Computer Graphics Forum 30*, 8 (December 2011), 2367–2386.

[BCK*13] Bénard P., Cole F., Kass M., Mordatch I., Hegarty J., Senn M. S., Fleischer K., Pesare D., Breeden K.: Stylizing Animation By Example. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (July 2013), 119:1–119:12.

[BJS*08] Barnes C., Jacobs D. E., Sanders J., Goldman D. B., Rusinkiewicz S., Finkelstein A., Agrawala M.: Video puppetry: A performative interface for cutout animation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 27*, 5 (Dec. 2008), 124:1–124:9.

[BNTS07] Bousseau A., Neyret F., Thollot J., Salesin D.: Video watercolorization using bidirectional texture advection. *ACM Transaction on Graphics (Proc. SIGGRAPH) 26*, 3 (2007).

[Bra00] Bradski G.: The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[BSM*07] Breslav S., Szerszen K., Markosian L., Barla P., Thollot J.: Dynamic 2D Patterns for Shading 3D Scenes. *ACM Transaction on Graphics (Proc. SIGGRAPH) 26*, 3 (2007).

[CRH05] Collomosse J. P., Rowntree D., Hall P. M.: Rendering cartoon-style motion cues in post-production video. *Graphical Models 67*, 6 (2005), 549–564.

[DLKS18] Dvorožňák M., Li W., Kim V. G., Sýkora D.: ToonSynth: Example-based synthesis of hand-colored cartoon animations. *ACM Transactions on Graphics (Proc. SIGGRAPH) 37*, 4 (2018).

[DOIB12] Delong A., Osokin A., Isack H., Boykov Y.: Fast Approximate Energy Minization with Label Costs. *International Journal of Computer Vision (IJCV) 96*, 1 (2012), 1–27.

[FJS*17] Fišer J., Jamriška O., Simons D., Shechtman E., Lu J., Asente P., Lukáč M., Sýkora D.: Example-based synthesis of stylized facial animations. *ACM Transactions on Graphics (Proc. SIGGRAPH) 36*, 4 (2017).

[FLJ*14] Fišer J., Lukáč M., Jamriška O., Čadík M., Gingold Y., Asente P., Sýkora D.: Color me noisy: Example-based rendering of hand-colored animations with temporal noise control. *Computer Graphics Forum (Proc. EGSR) 33* (2014).

[GEB16] Gatys L. A., Ecker A. S., Bethge M.: Image style transfer using convolutional neural networks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (June 2016).

[GJ*10] Guennebaud G., Jacob B., et al.: Eigen v3. http://eigen.tuxfamily.org, 2010.

[HE04] Hays J., Essa I. A.: Image and video based painterly animation. In *International Symposium on Nonphotorealistic Animation and Rendering (NPAR)* (2004).

[HP00] Hertzmann A., Perlin K.: Painterly rendering for video and interaction. In *International Symposium on Nonphotorealistic Animation and Rendering (NPAR)* (2000).

[IB12] Isack H., Boykov Y.: Energy-based geometric multi-model fitting. *International Journal of Computer Vision 97* (April 2012), 123–147.

[IMS*17] Ilg E., Mayer N., Saikia T., Keuper M., Dosovitskiy A., Brox T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[KCWI13] Kyprianidis J. E., Collomosse J., Wang T., Isenberg T.: State of the art: A taxonomy of artistic stylization techniques for images and video. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 19*, 5 (2013), 866–885.

[KP11] KASS M., PESARE D.: Coherent noise for non-photorealistic rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH) 30*, 4 (July 2011), 30:1–30:6.

[Lit97] LITWINOWICZ P.: Processing images and video for an impressionist effect. In *SIGGRAPH* (1997), pp. 407–414.

[LTF*05] LIU C., TORRALBA A., FREEMAN W. T., DURAND F., ADELSON E. H.: Motion magnification. *ACM Transactions on Graphics (Proc. SIGGRAPH) 24*, 3 (2005).

[LYKL12] LEE S.-Y., YOON J.-C., KWON J.-Y., LEE I.-K.: Cartoonmodes: Cartoon stylization of video objects through modal analysis. *Graphical Models 74*, 2 (2012), 51 – 60.

[Mei96] MEIER B. J.: Painterly rendering for animation. In *SIGGRAPH* (1996), pp. 477–484.

[OH12] O'DONOVAN P., HERTZMANN A.: Anipaint: Interactive painterly animation from video. *IEEE Transactions on Visualization and Computer Graphics (TVCG) 18*, 3 (2012), 475–487.

[RC13] ROSIN P., COLLOMOSSE J.: *Image and Video-Based Artistic Stylisation*. Springer, 2013.

[RDB18] RUDER M., DOSOVITSKIY A., BROX T.: Artistic style transfer for videos and spherical images. *International Journal of Computer Vision* (2018).

[RKB04] ROTHER C., KOLMOGOROV V., BLAKE A.: "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (Proc. SIGGRAPH) 23*, 3 (2004), 309–314.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proc. EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing* (2007), pp. 109–116.

[SBK10] SUNDARAM N., BROX T., KEUTZER K.: Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV)* (Sept. 2010), Lecture Notes in Computer Science, Springer.

[SED16] SELIM A., ELGHARIB M., DOYLE L.: Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics (Proc. SIGGRAPH) 35*, 4 (July 2016), 129:1–129:18.

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006), 533.

[VdBBR*12] VAN DEN BERGH M., BOIX X., ROIG G., DE CAPITANI B., VAN GOOL L.: Seeds: Superpixels extracted via energy-driven sampling. In *European conference on computer vision* (2012), Springer, pp. 13–26.

[VSR13] VOGEL C., SCHINDLER K., ROTH S.: Piecewise rigid scene flow. In *IEEE International Conference on Computer Vision (ICCV)* (2013).

[WDAC06] WANG J., DRUCKER S. M., AGRAWALA M., COHEN M. F.: The cartoon animation filter. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006), 1169–1173.

[WRDF13] WADHWA N., RUBINSTEIN M., DURAND F., FREEMAN W. T.: Phase-based video motion processing. *ACM Transaction on Graphics (Proc. SIGGRAPH) 32*, 4 (2013).

[WRS*12] WU H.-Y., RUBINSTEIN M., SHIH E., GUTTAG J., DURAND F., FREEMAN W. T.: Eulerian video magnification for revealing subtle changes in the world. *ACM Transaction on Graphics (Proc. SIGGRAPH) 31*, 4 (2012).

[YL15] YANG J., LI H.: Dense, accurate optical flow estimation with piecewise parametric model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).