Video Watercolorization using Bidirectional Texture Advection

Adrien Bousseau[1,3] Fabrice Neyret[2] Joëlle Thollot[3] David Salesin[1,4]
[1]Adobe Systems [2]LJK/IMAG-INRIA [3]INRIA Grenoble University [4]University of Washington
{Adrien.Bousseau|Fabrice.Neyret|Joelle.Thollot}@imag.fr salesin@adobe.com

Figure 1: A watercolorized video. *Original video (left), and its watercolorization (right).*

Abstract

In this paper, we present a method for creating watercolor-like animation, starting from video as input. The method involves two main steps: applying textures that simulate a watercolor appearance; and creating a simplified, abstracted version of the video to which the texturing operations are applied. Both of these steps are subject to highly visible temporal artifacts, so the primary technical contributions of the paper are extensions of previous methods for texturing and abstraction to provide temporal coherence when applied to video sequences. To maintain coherence for textures, we employ texture advection along lines of optical flow. We furthermore extend previous approaches by incorporating advection in both forward and reverse directions through the video, which allows for minimal texture distortion, particularly in areas of disocclusion that are otherwise highly problematic. To maintain coherence for abstraction, we employ mathematical morphology extended to the temporal domain, using filters whose temporal extents are locally controlled by the degree of distortions in the optical flow. Together, these techniques provide the first practical and robust approach for producing watercolor animations from video, which we demonstrate with a number of examples.

Keywords: Non-photorealistic rendering, abstract stylization, animated textures, temporal coherence.

1 Introduction

Watercolors have two essential characteristics that make them especially beautiful and evocative. They have distinctive textures — including those due to separation and granulation of pigments, as well as the texture of the paper itself, which shows through on account of watercolors' transparency. And they have an ability to suggest detail through abstract washes of color. In recent years, researchers in computer graphics have found ways to reproduce much of watercolors' characteristic appearance, taking an image as input and producing a watercolor-like illustration as output in a process known as "watercolorization" [Curtis et al. 1997; Lum and Ma 2001; Van Laerhoven et al. 2004; Bousseau et al. 2006]. However, the same characteristics that give watercolors their distinctive charm are nonetheless quite difficult to reproduce in any temporally coherent fashion, so the analogous process of taking a *video* as input and producing a watercolor-like *animation* as a result has remained a longstanding and elusive goal. In this paper, we show how this goal can be achieved to a large extent.

We use two different approaches to achieve temporal coherence. To preserve coherence of the watercolor texture itself, we advect a set of pigmentation textures according to an optical flow field computed for the video. And to produce a temporally coherent abstraction of the video, we use 3D mathematical morphology operations to create spatially and temporally smooth (i.e., non-flickering), simplified areas to render with watercolor textures.

The major technical contribution of the paper is the combination and application of these two existing algorithmic techniques, *texture advection* and *mathematical morphology*, to a new domain.[1] Moreover, the successful adaptation of these techniques to this new domain has required extending them in a number of ways. For advection, we propose a new scheme that involves simultaneously advecting two different texture fields, in forward and reverse directions through the video, and optimizing their combination on a per-pixel basis so as to yield minimal texture distortion — even in areas of disocclusion, which are notoriously difficult to handle with previous approaches. For abstraction via mathematical morphology, we extend traditional approaches by also filtering over time, and by taking into account errors and distortions in optical flow to control the temporal extent of the 3D filters. Together, these contributions provide the first practical and reasonably robust approach for producing realistic watercolor animations from video sequences.

In the rest of this paper, we will briefly survey related work (Section 2), describe our approach to texture advection in detail (Section 3), present our extensions of mathematical morphology for abstraction (Section 4), demonstrate our results (Section 5), and finally discuss areas for future work (Section 6).

[1]Advection has previously been used for visualizing fluids and flow, while mathematical morphology has typically been used for image denoising.

2 Related work

A significant body of research has been devoted to creating watercolor rendering for static images [Small 1991; Curtis et al. 1997; Lum and Ma 2001; Lei and Chang 2004; Van Laerhoven et al. 2004; Johan et al. 2005; Luft and Deussen 2006; Bousseau et al. 2006]. Commercial tools, like Photoshop™and The GIMP, also provide ways of creating watercolor-like renderings from images.

In general, the difficulty with extending non-photorealistic rendering (NPR) techniques from static to moving images is that, without careful consideration to temporal coherence, the resulting animations exhibit one of two problems: either the illustration effects remain fixed in place across the image, an unwanted artifact that has become known as the "shower door" effect; or the illustration effects exhibit no temporal coherence whatsoever, randomly changing in position and appearance from frame to frame, which can be even more visually distracting.

Several techniques have been developed to combat these problems in various NPR styles, although most of this work concerns the problem of animated 3D scenes, in which geometry information is available. For watercolorization, these approaches rely on texture mapping in 3D [Lum and Ma 2001], or on the distribution of discrete 2D primitives over surfaces [Luft and Deussen 2006; Bousseau et al. 2006]. Similarly, most of the work on video stylization has been in the realm of primitive-based rendering, where discrete paint strokes [Litwinowicz 1997; Hertzmann and Perlin 2000; Hays and Essa 2004; Collomosse et al. 2005] or other primitives [Klein et al. 2002] are applied to create the stylization. Applying such methods to watercolor raises the difficult question of finding a set of 2D primitives that, once combined, produces a continuous texture (typically pigments or paper). In that spirit, Bousseau et al. [2006] has proposed to use a Gaussian kernel as a primitive for watercolor effects, but was restricted to Perlin noise. We rather evolve the texture globally, allowing us to deal with any scanned texture.

A related problem has been addressed by Cunzi et al. [2003], who use a "dynamic canvas" to preserve the appearance of a 2D background paper texture while rendering an interactive walkthrough of a static 3D scene. Our work takes inspiration from theirs to perform a similar kind of dynamic adaptation of our watercolor textures, while at the same time tracking dynamic object motions in a changing video scene. Our work also shares some similarity to Fang's "RotoTexture" [Fang 2006], in that both attempt to provide textures that track dynamic scenes; however, Fang's work is concerned with maintaining the appearance of 3D textures rather than 2D. Other work in scientific visualization [Max and Becker 1995; Jobard et al. 2001], fluid simulation [Stam 1999; Neyret 2003], and artistic image warping [Sims 1992] shares the goal of evolving texture along a 2D vector field. Our work builds on the advection approach that these schemes introduced.

A significant body of research has been concerned with the issue of abstraction of video as well. Winnemöller et al. [2006] presented a method to smooth a video using a bilateral filter, which reduces the contrast in low-contrast regions while preserving sharp edges. We use a similar approach to produce large uniform color areas, and we go a step further in simplifying not just the color information but the 2D scene geometry as well. In "video tooning," Wang et al. [2004] use a mean-shift operator on the 3D video data to cluster colors in space and time. To smooth geometric detail, they employ a polyhedral reconstruction of the 3D clusters followed by mesh smoothing to obtain coarser shapes. Collomosse et al. [2005] use a similar type of geometric smoothing involving fitting continuous surfaces to voxel objects. Such high-level operations are usually computationally expensive and may sometimes require user input to produce convincing results. Our approach, by contract, uses simple low-level image processing for geometric as well as color simplification. In essence, we extend the approach of Bousseau et al. [2006], who use a morphological 2D filter to abstract the shapes of a still image and mimic the characteristic roundness of watercolor, by extending the morphological filter to the 3D spatiotemporal volume in a way that provides temporal coherence as well.

Finally, many image filtering operations have been extended to the spatiotemporal domain to process video: the median filter [Alp et al. 1990], average filter [Ozkan et al. 1993], and Wiener filter [Kokaram 1998] are all examples. A motion compensation is usually applied before filtering to avoid ghosting artifacts in regions of high motion. Recently, Laveau and Bernard [2005] proposed orienting a morphological structuring element along an optical flow path in order to apply these operators on videos. However, their filters have been developed in the context of noise removal, which requires filters of small support. Our application is more extreme in that it targets the removal of significant image features larger than a single pixel. To this end, we propose a type of adaptive structuring element that smoothes the appearance and disappearance of significant image features.

3 Texture advection

The "granulation" of watercolor pigments provides a large share of the richness of continuous tone techniques. Granulation is caused by pigment deposition on paper or canvas: the more pigments are deposited, the more the color is saturated. The effect is seen in both wet techniques like watercolor and dry techniques like charcoal or pastel. In addition, watercolors are transparent, and their transparency allows the texture of the paper itself to show through.

To achieve these texture effects, we follow the approach of Bousseau et al. [2006], who showed that for computer-generated watercolor, convincing visual results are obtained by considering a base color image C (e.g., a photograph) that is modified according to a grey-level pigment texture $P \in [0,1]$ at each pixel to produce a modified color C' according to the equation

$$C' = C\big(1-(1-C)(P-0.5)\big) \tag{1}$$

However, in order to create effective watercolor animations, this texture P must satisfy two competing constraints: On the one hand, it must maintain its appearance in terms of a more or less homogeneous distribution and frequency spectrum. On the other hand, it must follow the motion in the scene so as not to create a "shower door" effect.

To resolve this conflict we build on previous work on advected textures [Max and Becker 1995; Neyret 2003], classically used to depict flow in scientific visualizations. The general idea of such methods is to initialize the texture mapping on the first frame of an animation, and then evolve the mapping with the motion flow. In these methods the texture mapping is reinitialized whenever the statistical spatial properties of the current texture become too dissimilar to the original one.

We employ this same basic idea to our situation of applying texture to video, substituting optical flow for fluid flow. One significant complication, which arises quite frequently for videos but not for continuous fluid flows simulations, is the occurrence of *disocclusion boundaries*: places where new pixels of the background are revealed as a foreground object moves across a scene. Optical flow fields at disocclusions are essentially undefined: there is no pixel in the prior frame corresponding to these disoccluded boundary regions. Classical advected textures are designed for handling only continuous space-time distortions. In the absence of continuity, they tend to fail quite badly as shown Figures 3 and 4.

In order to handle disoccluded boundaries effectively, we introduce the notion of *bidirectional advection*: simultaneously advecting two texture layers in opposite directions in time — from the first frame of the video to the last, and from the last frame to the first. We use a combination of these two texture layers weighted at each pixel by the local quality of the texture from each direction. In the rest of this section we describe our algorithm and its properties.

3.1 Advection computation

In the following, we will use $\mathbf{x} = (x, y)$ for screen coordinates, and $\mathbf{u} = (u, v)$ for texture coordinates. An advected texture relies on a field of texture coordinates $\mathbf{u}(\mathbf{x}, t)$, which is displaced following a vector field $\mathbf{v}(\mathbf{x}, t)$: for any frame t, the vector $\mathbf{u}$ defines the location within the texture image P_0 to be displayed at position $\mathbf{x}$, i.e. the *mapping*. For simplicity we assume that $\mathbf{x}$ and $\mathbf{u}$ coordinates are normalized on the interval $[0, 1]$ and that $\mathbf{u}(\mathbf{x}, 0) = \mathbf{x}$. Therefore in Equation 1 the composited pigment texture will be $P(\mathbf{x}, t) = P_0(\mathbf{u}(\mathbf{x}, t))$. We will see in the following that several such layers will be combined to obtain the final result. Our vector field is obtained from an optical flow extracted from the video (we rely on a classical gradient-based method available in Adobe After Effects™). The vector $\mathbf{v}$ indicates for each frame t where the pixel at position $\mathbf{x}$ came from within frame $t-1$: $\mathbf{v}(\mathbf{x}, t) = \mathbf{x}_{t-1} - \mathbf{x}_t$.

The purpose of advection is to "attach" the texture P_0 to the moving pixels of the video, which is theoretically done by displacing the texture coordinates according to the vector field:[2] $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x} + \mathbf{v}(\mathbf{x}, t), t - 1)$. However, this backward mapping is problematic wherever the optical flow is poor or ill-defined, as at disocclusion boundaries. We will discuss how these problematic cases are handled momentarily.

3.2 Controlling the distortion

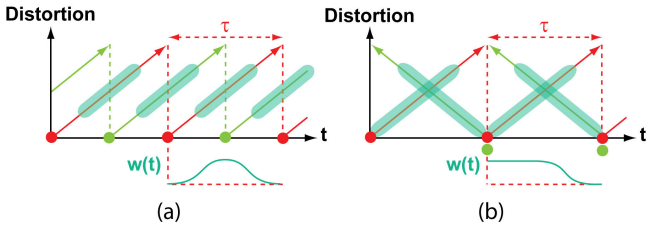
With this basic approach, the distortion of the advected texture increases with time. To combat this distortion, Max and Becker's [1995] and Neyret's [2003] approaches blend two or three phase-shifted texture layers, respectively (See Figure 2). In both schemes, the distortion is reset periodically (i.e., $\mathbf{u}(\mathbf{x}, t + \tau) = \mathbf{u}(\mathbf{x}, t)$), allowing the original texture to be used again. The regeneration period τ is chosen via a user defined delay [Max and Becker 1995] or a dynamic estimation of the distortion [Neyret 2003]. The fact that the regeneration occurs periodically guaranties that our system can handle videos of arbitrary length.

Like Max and Becker, we rely on two texture layers, but we combine one "forward" mapping $\mathbf{u}_f$, advected from the beginning to the end of the video sequence, with one "reverse" mapping $\mathbf{u}_r$, advected from the end to the beginning. The final advected pigment texture P' is a combination of these two fields, calculated on a per-pixel basis by taking into account the local distortions ω_f and ω_r of the forward and reverse mappings $\mathbf{u}_f$ and $\mathbf{u}_r$, respectively, within a given frame:

$$P'(\mathbf{x}, t) \;=\; \omega_f(\mathbf{x}, t) P_0\big(\mathbf{u}_f(\mathbf{x}, t)\big) \;+\; \omega_r(\mathbf{x}, t) P_0\big(\mathbf{u}_r(\mathbf{x}, t)\big)$$

We will show precisely how the distortion is measured and how ω_f and ω_r are computed later on. For now, to understand the intuition behind this approach, it suffices to note that texture distortion gradually increases in the direction of advection. Since $\mathbf{u}_f$ is advected forward, its distortion increases with time. However, since $\mathbf{u}_r$ is advected backwards through the video sequence, its distortion *decreases* with time. The combination of the two textures can therefore be used to create a less distorted texture. Moreover, a disocclusion in the forward sequence becomes an occlusion in the reverse, which is no longer a source of distortion; thus, a well-defined texture can always be used. (A similar observation was noted by Chuang et al. [2002] in their work on video matting.)

[2] In order to manage boundary conditions properly, we assume that the texture P_0 is periodic, and that $\mathbf{u}(\mathbf{x}, t) \approx \mathbf{u}(\mathbf{x}, t-1) + \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \cdot \mathbf{v}(\mathbf{x}, t)$ whenever $\mathbf{x} + \mathbf{v}(\mathbf{x}, t)$ seeks outside $[0, 1]^2$, in the spirit of [Max and Becker 1995].



(a) (b)

Figure 2: Approaches to control the distortion of the advected texture. (a) *Max and Becker scheme, which uses two phase-shifted, overlapping texture layers at any given time.* (b) *Our scheme, which also uses two texture layers at a time but advected in opposite time directions. In the two diagrams, the green solid areas represents, roughly, the relative contribution of each advected layer to the final texture's appearance, which corresponds to the weight $w(t)$. Note that in prior work* (a)*, in order to maintain temporal coherence, the advected layers do not begin contributing significantly until they are already somewhat distorted. In our method* (b)*, by contrast, textures contribute maximally where they are least distorted. Blending textures advected in opposite time directions also allows for less distortion everywhere, since distortion is decreasing in one texture just as it is increasing in the other. Finally, bidirectional advection handles disocclusion boundaries much better since the disocclusion leaves one of the two layers unaffected, with the unaffected layer contribution most to the texture's final appearance.*

The technical challenges of this approach are to quantify the visual distortions and to choose a clever way of combining the two advected fields. Three competing goals have to be taken into account: (1) minimizing the distortion; (2) avoiding temporal discontinuities; and (3) limiting visual artifacts such as contrast variation.

In the rest of this section, we detail our method: how we quantify the distortion (Section 3.3), and how we adjust the weights of the two advected fields (Section 3.4).

3.3 Distortion computation

We need a way to estimate the visual quality of a distorted texture at each pixel. Various methods exist to compute the distortion of a non-rigid shape. The general principle is to compute a deformation tensor and to choose an appropriate norm to compute the distortion.

All deformation tensors are based on the deformation gradient tensor F, corresponding to the Jacobian matrix of the shape coordinates (in our case the texture coordinates): $F_{ij}(\mathbf{x}, t) = \partial \mathbf{u}_i(\mathbf{x}, t) / \partial x_j$. As in previous work, we do not wish to consider translations and rotations to be distortions because they do not alter the visual properties of the texture. Instead, it is typical to rely on a strain tensor, which cancels antisymmetric components. However, unlike Neyret [2003], we wish to deal with large displacements, so we cannot use the infinitesimal strain tensor, which is the classical approximation. We therefore choose the Cauchy-Green tensor: $G = F^T F$ (one can verify that multiplying F by its transpose cancels the rotations). The eigenvectors of G give the principal directions of deformations, and the tensor's eigenvalues λ_i give the squares of the principal deformation values: an eigenvalue $\lambda_i > 1$ corresponds to a stretch, whereas an eigenvalue $\lambda_i < 1$ corresponds to a compression.

We want to derive an estimation of the visual quality of the *distortion* ξ as a scalar in $[0, 1]$ with 0 representing no distortion, and 1 representing distortion that is intolerable. We assume that compres-

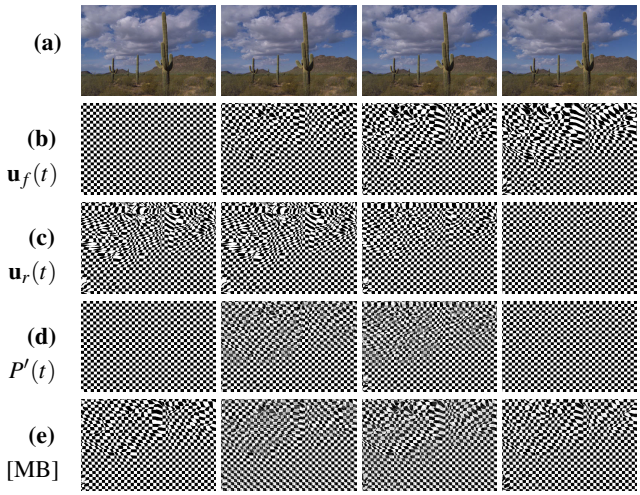


Figure 3: Analysis of our bidirectional advected texture. (a) *Original video sequence.* (b) *A checkerboard texture (for illustration purposes), advected forward using optical flow.* (c) *The same texture, advected in the reverse time direction using reverse optical flow.* (d) *The combined, bidirectional advected texture.* (e) *Advected texture using the previous approach of Max and Becker. Note how the bidirectional advected texture in* (d) *shows much less distortion in all frames than the previous approach in* (e).

sion and stretching are equally bad from a visual standpoint. We therefore define the *visual deformation* in the two principal directions σ_1 and σ_2 as: $\sigma_i(\mathbf{x},t) = \max\left(\sqrt{\lambda_i(\mathbf{x},t)}, 1/\sqrt{\lambda_i(\mathbf{x},t)}\right)$.

We define the visual distortion ξ as the quadratic mean of both deformations, normalized to $[0,1]$:

$$\xi(\mathbf{x},t) \quad = \quad \frac{\xi'(\mathbf{x},t) - \xi_{min}}{\xi_{max} - \xi_{min}}$$

where $\xi'(\mathbf{x},t) = \sqrt{\sigma_1^2(\mathbf{x},t) + \sigma_2^2(\mathbf{x},t)}$ is an unnormalized scalar measure of the visual distortion; $\xi_{min} = \sqrt{2}$ is the minimum value of ξ' (representing no distortion); and ξ_{max} is a maximum bound over which the distortion is considered too high, measured experimentally. In practice we use $\xi_{max} = 5$.

3.4 Adjusting weights

Given the distortions ξ_f and ξ_r of each advected mapping $\mathbf{u}_f$ and $\mathbf{u}_r$, we must now find the appropriate set of weights ω_f and ω_r at each pixel in order to minimize the final distortion.

Our weights must satisfy a number of properties: They should lie on the interval $[0,1]$; sum to 1 at every pixel; be inversely related to the texture distortion; gradually decrease to 0 at regeneration events in order to maintain overall temporal continuity; and vary smoothly, both in space and in time.

We choose the following definition for the weights, which satisfy all of these properties:

$$\omega_f(\mathbf{x},t) = \frac{\omega'_f(\mathbf{x},t)}{\omega'_f(\mathbf{x},t) + \omega'_r(\mathbf{x},t)} \qquad \omega_r(\mathbf{x},t) = \frac{\omega'_r(\mathbf{x},t)}{\omega'_f(\mathbf{x},t) + \omega'_r(\mathbf{x},t)}$$

with

$$\omega'_f(\mathbf{x},t) = g_f(\mathbf{x},t)\, h_f(t) \qquad \omega'_r(\mathbf{x},t) = g_r(\mathbf{x},t)\, h_r(t)$$

Here, g_f and g_r are measures of the distortions of the forward and reverse textures relative to the other:

$$g_f(\mathbf{x},t) = \frac{1 - (\xi_f - \xi_r)}{2} \qquad g_r(\mathbf{x},t) = \frac{1 - (\xi_r - \xi_f)}{2}$$

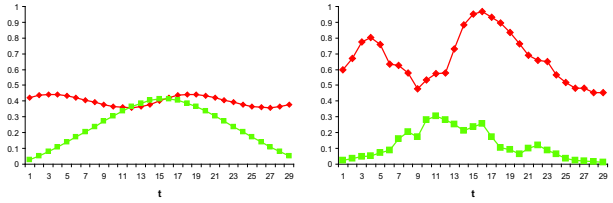


Figure 4: Comparison of distortion rates for our method (green squares) vs. Max and Becker (red diamonds). *Shown are resulting distortions for a theoretical case, in which distortion increases at a constant rate* (left)*; and a real case, measured at a disoccluded pixel (event occurs at* $t = 12$*)* (rigth). *Note how our method is able to cancel large distortions almost entirely.*

and h_f and h_r are the temporally decaying weighting functions shown in Figure 2:

$$h_f(t) = \cos^2\left(\frac{\pi}{2}\,\frac{t \bmod \tau}{\tau}\right) \qquad h_r(t) = \sin^2\left(\frac{\pi}{2}\,\frac{t \bmod \tau}{\tau}\right)$$

where τ is the delay between two regenerations.

Figure 3 shows the resulting advected texture for a test scene, and Figure 4 shows the resulting distortion values for two different pixels. If the distortion rate is regular, our blending behaves much like that of Max and Becker [1995]; however, when the distortion rate is high, for example at disocclusion boundaries, our blending results are much better.

3.5 Limiting contrast oscillation and tuning τ

The blending of the two advected layers produces a cycle between frames with one single texture ($\omega_f = 1$ or $\omega_r = 1$) and frames with two blended textures (for all other values of ω_f and ω_r). Contrast is reduced, especially as ω_f and ω_r approach the same value of $1/2$, because high frequencies are dimmed. To reduce the perception of the resulting oscillation of contrast, we include a second pair of advected layers with a regeneration cycle halfway out of phase. The average of these two advected textures gives a nearly constant amount of contrast. As two out-of-phase identical textures moving in the same direction may produce spatial correlation (i.e., ghosting), we use a visually similar but distinct texture image for this pair. Thus, our complete model relies on four layers.

In previous methods, the regeneration period τ is chosen as a trade-off between texture distortion and loss of motion perception (if texture is regenerated too often there is no longer advection). As pointed out by Neyret [2003], the ideal value of τ is generally not the same in each region of the image, depending on the local velocity. Thus, Neyret proposes an adaptive regeneration scheme, which consists of advecting several texture layers with increasing regeneration periods, and locally interpolating between layers to minimize the distortion. In this way, fast regeneration occurs only in regions of high distortion. Our method already has some spatial adaptation. Still, the same idea is also applicable in our case since velocity varies in the optical flow. Being able to choose between various periods also helps minimize the distortion due to disocclusion. In practice, we found that two pairs of layers were sufficient in most cases. One fast layer ($\tau = 15$ frames) allows for good correction of disocclusions, while a slower layer ($\tau = 30$ or 60 frames) provides good motion perception in slow-motion areas.

4 Shape abstraction

One of the distinctive characteristics of watercolor is the medium's ability to suggest detail with abstract washes of color. In this section, we examine how raw video footage can be abstracted into

shapes that are more characteristic of watercolor renderings, while remaining temporally coherent.

To this end, we generalize the work of Bousseau et al. [2006], who use mathematical morphology filters to create more regular image regions. While relatively little known, such filters have long been described as having the ability to "simplify image data, preserving their essential shape characteristics and eliminating irrelevancies" [Haralick et al. 1987]. In the following we briefly present the generalized versions of the morphological operators for continuous-tone images, before extending them to temporally coherent operators adapted to video stylization. For further details on morphological filtering, see the overview of Serra and Vincent [1992] or the work of Haralick et al. [1987].

4.1 Morphological operators

Figure 5: Mathematical morphology operators, generalized for continuous-tone images. (a) *Original image.* (b) *Erosion.* (c) *Dilation.* (d) *Opening (erosion followed by dilation).* (e) *Closing (dilation followed by erosion).* (f) *Closing followed by opening: our chosen morphological filter.*

Let I be an image and B a *structuring element*, that is an array that contains the relative coordinates of a pixel's neighborhood. The *morphological dilation* δ of I by B at a pixel $\mathbf{x}$ and its dual, the *morphological erosion* ε, are defined by

$$\delta_B(I)(\mathbf{x}) = \max_{b\in B}\{I(\mathbf{x}-b)\} \quad \varepsilon_B(I)(\mathbf{x}) = \min_{b\in B}\{I(\mathbf{x}+b)\}$$

The dilation spreads the light features of the image whereas the erosion spreads the dark features. (See Figure 5.)

The *morphological opening* is then defined as a sequence of one erosion followed by one dilation, $\delta_B \circ \varepsilon_B(I)$, and the *morphological closing* as one dilation followed by one erosion $\varepsilon_B \circ \delta_B(I)$. Opening removes light features of the image, whereas closing removes dark features.

An effective filter for removing both dark and light features, used by Bousseau et al. [2006], is the sequence of one closing followed by one opening. In this case, the size of the morphological structuring element defines the minimum size of the features preserved by the filtering, and the shape of the structuring element defines the shape of the filtered objects in the resulting image. For simplicity, the operators are applied on the three color channels separately. Although this independent filtering of the channels produces color ghosting on dilation and erosion (see Figure 5(b,c)), it becomes unnoticeable when these dual operators are applied in sequence (see Figure 5(d,e,f)).

4.2 Spatiotemporal morphological operators

Applying morphological filtering on each image separately produces a great deal of flickering, as many features appear and disappear at every frame (See Figure 6.) Moreover, as each feature is at least as large as the 2D structuring element, the features' appearances and disappearances produce noticeable "popping" artifacts. To reduce temporal artifacts, we extend the structuring element to the temporal domain. The resulting 3D structuring element can be thought of as a stack of neighborhoods in k successive frames. The 3D element reduces flickering as it guarantees that each new feature remains visible during all k frames. However, care must be taken to correctly orient the filter. Indeed, moving objects correspond to spatiotemporal regions that are not necessarily perpendicular to the time axis. As in previous work [Ozkan et al. 1993; Kokaram 1998], we compensate for this motion by translating the neighborhood at each pixel according to the optical flow for each frame. The resulting filtering produces uniform temporal regions, resulting in a higher temporal coherence.

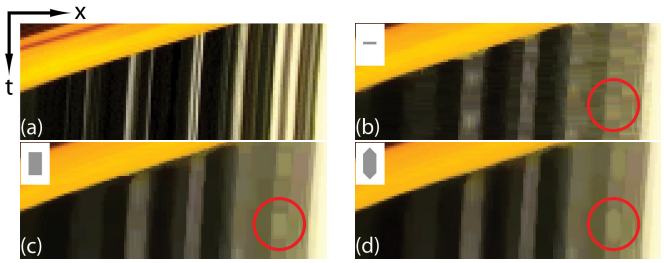


Figure 6: A visualization of the effects of various filters on the spatiotemporal volume (filter represented in gray). *These figures show a portion of a single horizontal scanline of video as a function of time, which appears on the vertical axis.* (a) *Original video.* (b) *The results of a 2D morphological filter (note the vertical noise, which corresponds to flickering in the filtered sequence).* (c) *The results of a constant-width spatiotemporal filter (note the sudden onset of features, corresponding to popping in the filtered sequence).* (d) *The results of our tapered filter (note how features appear and disappear gradually rather than suddenly).*

Unlike previous methods in video denoising, which used small filter kernels (usually $3\times3\times3$ pixels), we would like to use kernels with much larger spatiotemporal support (generally, $7\times7\times9$ pixels) to abstract away significant details. In order to reduce the popping produced by the abrupt appearance or disappearance of these details, we design the structuring element in such a way that visibility events occur gradually. As such, we taper the structuring element at its extremities (see Figure 7). The features in the resulting video appear and disappear gradually, as shown in Figure 6(d). Indeed, their shapes in the spatiotemporal volume, visualized in this figure, mirror precisely the shape of the tapered 3D structuring element that we use.

Figure 7: A visualization of how our tapered morphological filters follow the optical flow from frame to frame.

Finally, proper spatiotemporal filtering assumes that every pixel has a correct motion trajectory throughout the temporal support of each filter. In practice, this assumption fails for several reasons: First, a pixel can become occluded by another object in the video. Second, optical flow is usually just an approximate estimation of the motion flow, especially in disocclusion areas where no real correspondence

can be found between two successive frames. A common approach to increase the robustness for spatiotemporal filtering in such cases is to make the filter adaptive [Ozkan et al. 1993]: the idea is to ignore the contribution of outliers in the filtered value. However, a direct application of this method to morphological operators would create holes in the structuring element, which would have a direct impact on the shapes in the resulting image. Instead, we simply truncate the structuring element in the temporal domain as soon as an "outlier" is detected. A pixel is flagged as an outlier when its estimated motion accuracy is below a defined threshold. Similarly to Chuang et al. [2002], we estimate the accuracy of our optical flow assuming a constant intensity along motion trajectories. This is done by comparing the mean color on a 3×3 neighborhood of a pixel between two frames. In practice we use the L_2 norm of the distance in RGB.

5 Results and discussion

Following Bousseau et al. [2006], we produce the final watercolor rendering as the composite of the abstracted video with the advected pigmentation texture using Equation (1) (see Figure 8). We also add a similar edge darkening pass on the abstracted video. The spatiotemporal filter we have described can either be used alone or after first applying a bilateral filtering that simplifies colors, as in Winnemoeller et al. [2006].

We have implemented our method as an Adobe After Effects plug-in. Our performance statistics are dependent on the whole system, and our algorithms have not been optimized. Currently, advection takes 5 seconds per frame and morphological filtering about 30. The advection computation could be sped up considerably using graphics hardware, as the current bottlenecks are mainly texture access and bilinear interpolation. Morphological filtering operations are notoriously expensive. However, some of the clever implementation strategies devised for 2D processing [Weiss 2006] may be generalizable to 3D, which could greatly improve the performance.

Our results are best viewed in the accompanying videos.[3] The "cactus" sequence shows how our new advection scheme handles strong motion boundaries. The "waves" sequence gives an example of complex motions that are well depicted by the advection, without introducing distortions.

In all of the results, the quality of the texture advection is limited by the quality of the optical flow. Errors in the optical flow lead to "swimming" artifacts in the pigment texture. Such errors are especially visible in relatively unstructured areas of the scene, and near occlusion boundaries. Other kind of motion representations, such as motion layers [Wang and Adelson 1994] could correct these artifacts in some situations, but we believe that the vector field representation can handle motions that other representations cannot (e.g., zoom or large parallax on the same object). We have also applied our advection scheme on a computer-generated sequence, which shows how the advection can accurately depict complex motions and occlusions giving a correct motion flow, as shown in the accompanying video material.

As far as abstraction is concerned, our filtering is stable over time and offers large color regions that appear and disappear gradually over the animation. All the examples presented in our video have been abstracted with a $7 \times 7 \times 9$ structuring element. Increasing the spatial extent of the structuring element generally requires increasing its temporal extent as well, in order to allow time for the width of the structuring element to increase and decrease gradually.

[3] Videos are available on the project webpage
http://artis.imag.fr/Publications/2007/BNTS07

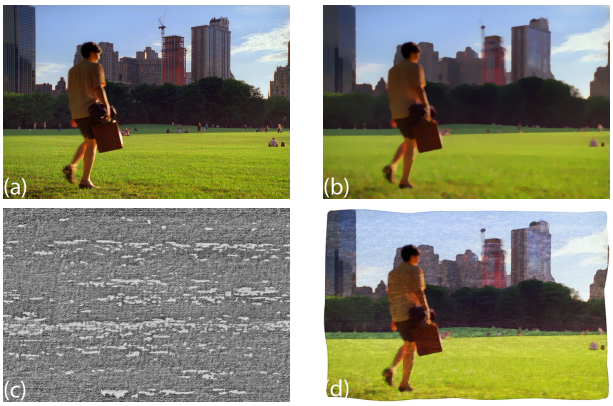


Figure 8: Watercolor compositing. *The final watercolor rendering* (d) *is obtained from an input video* (a) *as the composite of the abstracted video* (b) *with the advected pigmentation texture* (c)*, using Equation (1).*

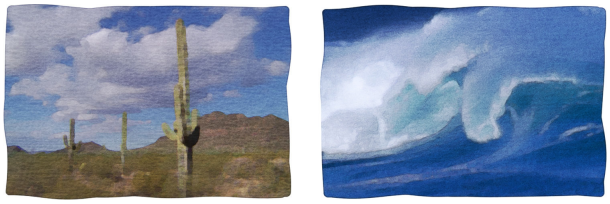


Figure 9: Final results. *Note the absence of distortions, even in the presence of motion boudaries* (left) *or complex motions* (right).

6 Conclusions

We have presented an approach to video watercolorization that provides good temporal coherence given good optical flow. By combining texture advection and mathematical morphology operations, and by extending them to handle the types of distortions commonly found in video sequences, we are able to produce computer-generated watercolor animations that are convincing for most scenes.

We see a number of areas for future work. First, we would like to investigate whether motion estimation methods based on a post-optimization of the optical flow, such as Particle Video [Sand and Teller 2006], could be used to obtain a better vector field from which we could perform our texture advection. It might also be worthwhile to explore other domains in which high-quality optical flow fields may be more readily available, such as watercolorizations of 3D pans and zooms across real-world scenes derived using structure-from-motion estimations [Horn 1986].

We would also like to look at speeding up our computations, perhaps through a GPU implementation of our texture advection approach, or adapting techniques for optimizing morphological filtering to 3D [Weiss 2006].

In addition, we would like to extend this work to other styles of illustration beyond watercolor, such as charcoal or pastel. In our experiments so far, we have found that the more "structured" the appearance of the medium's effects, the more objectionable any distortions due to incorrect optical flow appear. However, these artifacts might be ameliorated by decreasing the frame rate, and/or alternating between different versions of texture, as in, for example, much of Bill Plympton's animation.[4]

Finally, we are interested in seeing how our extensions could now be re-adapted to improve other domains. Our new advection method is applicable to scientific flow visualization, thus overcoming some

[4] http://www.plymptoons.com/

limitations of previous methods. Similarly, our new abstraction method can be used as a pre-processing step for any type of video stylization, improving existing stroke-based rendering techniques.

Acknowledgments

All the videos used in this work are from the Artbeats Digital Film Library. Thanks to Laurence Boissieux for the computer-generated sequence, to Mira Dontcheva for her help on the submission video, and to David Simons, Dan Wilk, and Bruce Bullis for their help on Adobe After Effects™. Thanks also to the anonymous reviewers for their constructive comments.

References

ALP, B., HAAVISTO, P., JARSKE, T., OISTAMO, K., AND NEUVO, Y. A. 1990. Median-based algorithms for image sequence processing. In *SPIE Vol. 1360, Visual Communications and Image Processing*, 122–134.

BOUSSEAU, A., KAPLAN, M., THOLLOT, J., AND SILLION, F. X. 2006. Interactive watercolor rendering with temporal coherence and abstraction. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 141 – 149.

CHUANG, Y.-Y., AGARWALA, A., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2002. Video matting of complex scenes. *ACM Transactions on Graphics (Proc. SIGGRAPH 2005) 21*, 3 (July), 243–248.

COLLOMOSSE, J. P., ROWNTREE, D., AND HALL, P. M. 2005. Stroke surfaces: Temporally coherent artistic animations from video. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (Sept.), 540–549.

CUNZI, M., THOLLOT, J., PARIS, S., DEBUNNE, G., GASCUEL, J.-D., AND DURAND, F. 2003. Dynamic canvas for immersive non-photorealistic walkthroughs. In *Graphics Interface*, 121–130.

CURTIS, C. J., ANDERSON, S. E., SEIMS, J. E., FLEISCHER, K. W., AND SALESIN, D. H. 1997. Computer-generated watercolor. In *SIGGRAPH 97*, 421–430.

FANG, H. 2006. Rototexture: Automated tools for texturing raw video. *IEEE Transactions on Visualization and Computer Graphics 12*, 6, 1580–1589.

HARALICK, R. M., STERNBERG, S. R., AND ZHUANG, X. 1987. Image analysis using mathematical morphology. *IEEE Trans. Pattern Anal. Mach. Intell. 9*, 4, 532–550.

HAYS, J., AND ESSA, I. 2004. Image and video based painterly animation. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 113–120.

HERTZMANN, A., AND PERLIN, K. 2000. Painterly rendering for video and interaction. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 7–12.

HORN, B. K. P. 1986. *Robot Vision*. MIT Press. ISBN 0-262-08159-8.

JOBARD, B., ERLEBACHER, G., AND HUSSAINI, M. Y. 2001. Lagrangian-eulerian advection for unsteady flow visualization. In *VIS '01: Conference on Visualization '01*, 53–60.

JOHAN, H., HASHIMOTA, R., AND NISHITA, T. 2005. Creating watercolor style images taking into account painting techniques. *Journal of the Society for Art and Science 3*, 4, 207–215.

KLEIN, A. W., SLOAN, P.-P. J., FINKELSTEIN, A., AND COHEN, M. F. 2002. Stylized video cubes. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)*, 15–22.

KOKARAM, A. C. 1998. *Motion Picture Restoration: Digital Algorithms for Artefact Suppression in Degraded Motion Picture Film and Video*. Springer-Verlag. ISBN 3-540-76040-7.

LAVEAU, N., AND BERNARD, C. 2005. Structuring elements following the optical flow. In *Mathematical Morphology: 40 Years On, Proc. of the 7th International Symposium on Mathematical Morphology*, Springer-Verlag, Ed., 43–52.

LEI, E., AND CHANG, C.-F. 2004. Real-time rendering of watercolor effects for virtual environments. In *IEEE 2004 Pacific-Rim Conference on Multimedia*, 474–481.

LITWINOWICZ, P. C. 1997. Processing images and video for an impressionist effect. In *SIGGRAPH 97*, 407–414.

LUFT, T., AND DEUSSEN, O. 2006. Real-time watercolor illustrations of plants using a blurred depth test. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 11–20.

LUM, E. B., AND MA, K.-L. 2001. Non-photorealistic rendering using watercolor inspired textures and illumination. In *Pacific Graphics*, 322–331.

MAX, N., AND BECKER, B. 1995. Flow visualization using moving textures. In *Proc. of the ICASW/LaRC Symposium on Visualizing Time-Varying Data*, 77–87.

NEYRET, F. 2003. Advected textures. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)*, 147 – 153.

OZKAN, M. K., SEZAN, M. I., AND TEKALP, A. M. 1993. Adaptive motion-compensated filtering of noisy image sequences. *IEEE transactions on circuits and systems for video technology 3*, 4, 277–290.

SAND, P., AND TELLER, S. 2006. Particle video: Long-range motion estimation using point trajectories. In *CVPR*, 2195 – 2202.

SERRA, J., AND VINCENT, L. 1992. An overview of morphological filtering. *Circuits Syst. Signal Process. 11*, 1, 47–108.

SIMS, K. 1992. Choreographed image flow. *The Journal of Visualization and Computer Animation 3*, 1, 31–43.

SMALL, D. 1991. Modeling watercolor by simulating diffusion, pigment, and paper fibers. In *SPIE*, vol. 1460, 140–146.

STAM, J. 1999. Stable fluids. In *SIGGRAPH 99*, 121–128.

VAN LAERHOVEN, T., LIESENBORGS, J., AND VAN REETH, F. 2004. Real-time watercolor painting on a distributed paper model. In *Computer Graphics International*, 640–643.

WANG, J. Y. A., AND ADELSON, E. H. 1994. Representing Moving Images with Layers. *The IEEE Transaction on Image Processing Special Issue: Image Sequence Compression 3*, 5, 625–638.

WANG, J., XU, Y., SHUM, H.-Y., AND COHEN, M. 2004. Video tooning. *ACM Transactions on Graphics (proc. of SIGGRAPH 2004) 23*, 3, 574 – 583.

WEISS, B. 2006. Fast median and bilateral filtering. *ACM Transactions on Graphics (proc. of SIGGRAPH 2006) 25*, 3, 519–526.

WINNEMOLLER, H., OLSEN, S. C., AND GOOCH, B. 2006. Real-time video abstraction. *ACM Transactions on Graphics (proc. of SIGGRAPH 2006) 25*, 3, 1221 – 1226.