



HAL
open science

Temporally Coherent Video Stylization

Pierre Bénard, Joëlle Thollot, John Collomosse

► **To cite this version:**

Pierre Bénard, Joëlle Thollot, John Collomosse. Temporally Coherent Video Stylization. Rosin, Paul and Collomosse, John. Image and Video-based Artistic Stylisation, 42, Springer, pp.257-284, 2012, Computational Imaging and Vision, 978-1-4471-4519-6. 10.1007/978-1-4471-4519-6_13. hal-00762944

HAL Id: hal-00762944

<https://inria.hal.science/hal-00762944>

Submitted on 10 Dec 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Chapter 13

Temporally Coherent Video Stylization

Pierre B enard, Jo elle Thollot and John Collomosse

This is the author's version of the work. It is posted here by permission of Springer for your personal use.

The final version is available on the editor's site:

http://dx.doi.org/10.1007/978-1-4471-4519-6_13

Abstract The transformation of video clips into stylized animations remains an active research topic in Computer Graphics. A key challenge is to reproduce the look of traditional artistic styles whilst minimizing distracting flickering and sliding artifacts; i.e. with temporal coherence. This Chapter surveys the spectrum of available video stylization techniques, focusing on algorithms encouraging the temporally coherent placement of rendering marks, and discusses the trade-offs necessary to achieve coherence. We begin with flow-based adaptations of stroke based rendering (SBR) and texture advection capable of painting video. We then chart the development of the field, and its fusion with Computer Vision, to deliver coherent mid-level scene representations. These representations enable the rotoscoping of rendering marks on to temporally coherent video regions, enhancing the diversity and temporal coherence of stylization. In discussing coherence, we formalize the problem of temporal coherence in terms of three defined criteria, and compare and contrast video stylization using these.

Pierre B enard

University of Toronto, 40 St George Street, Toronto, ON M5S 2E4, Canada

e-mail: Pierre.Benard@laposte.net

John Collomosse

Centre for Vision Speech and Signal Processing – University of Surrey, Guildford, Surrey, GU2 7XH, United Kingdom

e-mail: J.Collomosse@surrey.ac.uk

Jo elle Thollot

Grenoble University – LJK, INRIA, 655, avenue de l'Europe, 38334 Saint Ismier, France

e-mail: Joelle.Thollot@inria.fr

13.1 Introduction

Artistic Rendering (AR) arguably evolved from semi-automated stroke based rendering (SBR) systems of the early nineties. SBR is discussed in detail within Chapter 1. In brief, it is the process of compositing primitives (e.g. brush strokes) on to a virtual canvas to create a rendering. Following Willats and Durand [58], we refer to these rendering primitives as “marks”. Digital paint systems such as Haeberli’s ‘*paint by numbers*’ [24] were among the first to propose such a framework, seeking to partially automate the image stylization process. Marks adopt some attributes (e.g. color, and orientation) from a reference image, whilst user interaction governed other attributes such as scale and compositing order. Soon after, fully automatic algorithms emerged harnessing low-level image processing (e.g. edge filters [26, 40] and moments [50]) in lieu of user interaction. These advances in automation brought with them new algorithms designed *specifically* for video.

This Chapter maps the landscape of video stylization algorithms in approximate chronological order of development. We begin by briefly identifying some non-linear filtering methods, that when applied independently to video frames, can produce a coherent rendering output. Chapter 5 explores these techniques in greater detail. We then survey optical flow based methods [40], that like filters, are very general in the class of video footage that may be processed, but are limited predominantly to painterly styles. These approaches move marks such as strokes or texture fields over time according to a per-frame motion estimate. We then survey early approaches to stylization driven by coherent video segmentation [13, 15, 55]. These approach artistic stylization by treating coherent placement of strokes as a rotoscoping problem, and extend to cartoon-like styles. Finally we survey the recent, interactive techniques that extend these early works to sophisticated rotoscoping and artistic rendering tools.

13.1.1 Temporal Coherence

Barbara Meier developed an early object-space technique for creating painterly animations of 3D scenes [43], and whilst not a video stylization approach *per se* it was the first to consider the important issue of *temporal coherence* in SBR.

Meier’s approach to painting was to initialize (seed) marks (in her case, brush strokes) as particles over the 3D surfaces of objects. The particles were projected to 2D during rendering, and brush strokes generated on the image-plane. The motivation to anchor strokes to move with the object arises from Meier’s early observations on temporal coherence, and are echoed by many subsequent authors. Painting the scene geometry independently for each frame results in a distracting flicker. Yet, fixing stroke positions in 2D while allowing their attributes (e.g. color) to vary with the underlying video content gives the impression of motion behind frosted glass—the so called, *shower door effect*. Meier’s proposal was therefore to fix the strokes to the

surface of the 3D object; *minimizing flicker* whilst *maximizing the correspondence* between stroke motion and the motion of the underlying object.

Satisfying both criteria for coherence is tractable for 3D rendering where geometry is available, but is non-trivial when painting 2D video. As such, temporal coherence remains a key challenge in video stylization research. Understanding the motion of objects in an unconstrained monocular video feed requires a *robust* and *general* model of scene structure; a long-term goal that continues to elude the Computer Vision community. Moreover, there is no single best model for all video. The model selected to represent the scene's dynamics and visual structure impacts both *the classes of video content that can be processed*, and *the gamut of artistic styles that may be rendered*.

13.1.2 Problem Statement: Coherent Stylization

In order to better describe the problem of temporal coherence in this Chapter, we build upon Meier's discussion to propose a definition of temporal coherence. Temporal coherence video requires the concurrent fulfillment of three goals: *spatial quality*, *motion coherence* and *temporal continuity*.

1. Spatial quality describes the visual quality of the stylization at each frame. This is a key ingredient in generating computer animations that appear similar to traditional hand-drawn animations. Several properties of the marks must be preserved to produce a convincing appearance. In particular the size and distribution of marks should be independent of the underlying geometry of the scene. As a typical example, the size of the marks should not increase during a zoom, but their spatial density should not change neither. Deformation of the marks should be avoided. Marks should not compress around occlusions for instance.

2. Motion coherence is the correlation between the apparent motion flow of the 3D scene and the motion of the marks. A low correlation produces sliding artifacts and gives the impression that the scene is observed through a semi-transparent layer of marks; Meier's *shower door effect* [43].

3. Temporal continuity minimizes abrupt changes of the marks from frame to frame. Perceptual studies [47, 60] have shown that human observers are very sensitive to sudden temporal variations such as popping and flickering. The visibility and attributes of the marks should vary smoothly to ensure temporal continuity and fluid animations.

Unfortunately these goals are inherently contradictory and naïve solutions often neglect one or more criteria. For example, the texture advection technique of Bousseau et al. (Section 13.3.2.1) can be used to apply the marks over the scene with high motion coherence and temporal continuity, but deformations destroy the spatial quality of the stylization. Keeping the marks static from frame to frame ensures a good spatial quality and temporal continuity but produces a strong shower door effect since the motion of the marks has no correlation with the motion of the scene. Finally, processing each frame independently, similarly to hand-drawn ani-

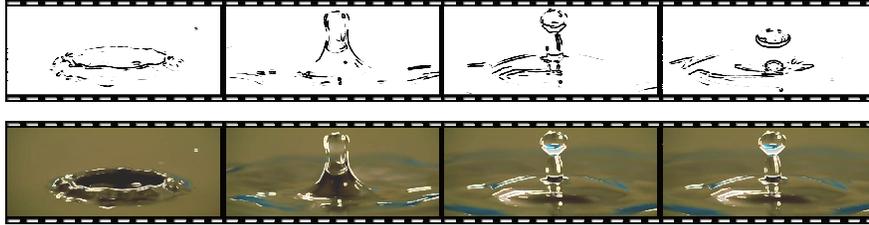


Fig. 13.1 Frames of a water drop video where luminance edges are depicted with lines of varying thickness, either drawn in black (top), or drawn in white over the original image (bottom), both with a disk footprint. From [52]  2011 Blackwell Publishing. Included here by permission.

mation, leads to pronounced flickering and popping since the position of the marks varies randomly from frame to frame.

13.2 Temporally Local Filtering

Arguably the most straightforward method to stylize a video is to apply image processing filters independently at each frame. Depending on the type of filter, the resulting video will be more or less coherent in time. Typically filters that incorporate hard thresholds will be less coherent than more continuous filters.

Winnem oller et al. [59] iteratively apply a bilateral filter followed by soft quantization to produce cartoon animations from videos in real-time. Image edges are extracted from the smoothed images with a difference of Gaussian filter. The soft quantization, less sensitive to noise, produces results with higher temporal coherence than traditional hard quantization.

As discussed in Chapter 5, Winnem oller’s stylization pipeline has been adapted to incorporate various alternative filters: e.g. Kuwahara based filters [35], combination of Shock filter with diffusion [36], and generalized geodesic distance transform [18] for soft clustering. These filtering approaches are fast to compute as they can be implemented on the GPU. However they are restricted in the gamut of artistic styles that may be produced.

Similarly, Vergne et al. [52] use 2D local differential geometry to extract luminance edges of a video. The feature lines are also implicitly defined, which prevents the use of an explicit parameterization needed for arc-length based effects or for mapping a brush stroke texture. However, they propose to formulate the mark rendering process as a spatially-varying convolution that mimics the contact of a brush of a given footprint with the feature line. It allows them to simulate some styles, like thickness variations that remain fully coherent over time (see Fig. 13.1).

This Chapter focuses primarily upon video stylization methods that encourage the *temporally coherent placement of marks*. This is achieved by rendering using information propagated from adjacent frames; i.e. frames are not rendered with temporal independence. The remainder of the Chapter covers these techniques, and the

reader is referred to Chapter 5 for more detailed coverage of the filtering techniques outlined in this section.

13.3 Optical Flow based Stylization

To make progress beyond independent filtering of frames, temporal correspondence may be established on a per pixel basis using motion estimation. Optical flow algorithms (e.g. [4] and related methods) can be applied to produce such an estimate.

Using this information, local filtering approaches can be extended by defining $2D + t$ filters that smooth the effect of the filter along time as described in Section 13.2. Moreover temporal continuity can be enforced using the optical flow to guide the evolution of the marks along the video. The difficulty is then to ensure the quality of the spatial properties of the marks. Various approaches have been proposed to solve this problem. We classify them in two categories: mark-based (Section 13.3.1) and texture-based (Section 13.3.2) approaches.

In this section we adopt the notation $\mathcal{I}_t(x, y)$ to denote the RGB video frame at time t , and similarly $I_t(x, y)$ for the greyscale frame. Edge orientation $\Theta_t(x, y)$ and edge strength $|\nabla I_t(x, y)|$ field are so denoted, and computed:

$$|\nabla I_t(x, y)| = \left(\frac{\delta I_t}{\delta x}^2 + \frac{\delta I_t}{\delta y}^2 \right)^{\frac{1}{2}} \quad (13.1)$$

$$\Theta_t(x, y) = \text{atan} \left(\frac{\delta I_t}{\delta y} / \frac{\delta I_t}{\delta x} \right) \quad (13.2)$$

13.3.1 Mark-based methods

Peter Litwinowicz proposed the first purpose designed algorithm for video stylization in 1997. The essence of the approach is to place marks upon the first video frame, and push (i.e. translate) them over time to match the estimated motion of objects in the video. Marks are moved according to the per-pixel motion estimate derived from the optical flow. As such, Litwinowicz adopts a weak motion model, applicable to very general input footage. However what optical flow offers in generality, it lacks in robustness. Pixels arising from the same object might be estimated with entirely different motion vectors. Whilst many optical flow algorithms exist, and some enforce local spatial coherence, in practice it is often the case that parts of objects are estimated with incorrect or inconsistent motion. This is especially true for objects with flat texture or weak intensity edges, as these visual cues often drive optical flow estimation algorithms. This can result in the *swimming* of painterly texture, due to motion mismatch between marks and underlying video content. The

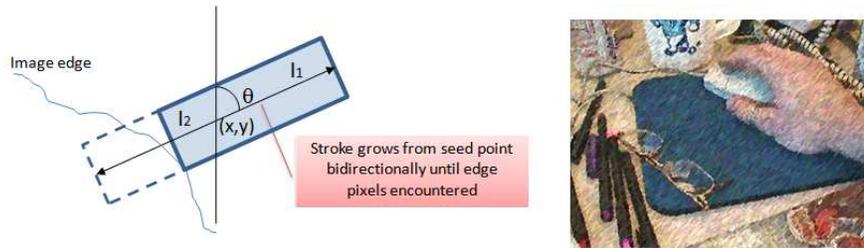


Fig. 13.2 Left: Stroke attributes in Litwinowicz’ approach to video stylization [40]. Strokes grow until a strong edge is encountered, thus preserving detail. Middle: Source Image. Right: Resulting painting. From  2002 ACM, Inc. Included here by permission.

sequential processing of frames can also cause motion estimation errors to accumulate, contributing to swimming artifacts.

In terms of our criteria for temporal coherence (Section 13.1.2), since the marks (e.g. brush strokes) are usually small with respect to object size, their motion remains very close to the original motion field of the depicted scene, providing good *motion coherence*. *Spatial quality* is also preserved by drawing the marks as 2D sprites and by ensuring an adequate density of marks. However as discussed, *temporal continuity* criteria is often violated due to error accumulation and propagation.

Nevertheless, Litwinowicz’ algorithm results in aesthetically pleasing renderings in many situations and especially when dealing with highly textured or anisotropic phenomena—where many more recent region-based methods struggle. Fluids, smoke, cloud, and similar phenomena are ideally suited to the generality of the optical flow fields, and the fields may also be manually embellished to produce attractive swirls reminiscent of a Van Gogh. Interactive software implementing this technique won an Oscar for visual effects in the motion picture “What Dreams May Come” (1999), for the creation of painterly landscapes of flowers, sea and sky [23].

13.3.1.1 Impressionist Painterly Rendering

Litwinowicz’ [40] algorithm uses a multitude of short rectangular brush strokes as marks, to create an impressionist video effect. A sequence of strokes are created using the first frame of video. Pixels within \mathcal{S}_1 are sub-sampled in a regular grid (typically every second pixel) yielding a set of N strokes $\mathcal{S}_1 = \{s_1^1, \dots, s_1^N\}$ where each stroke is represented by a tuple $s^i = (x, y, \theta, \mathbf{c})$ encoding each stroke’s seed location (x, y) , orientation θ , color $\mathbf{c} = \mathcal{S}_1(x, y)$ and length l_1, l_2 . Figure 13.2 illustrates the stroke geometry with respect to these parameters. Each stroke is grown iteratively from its seed point (x, y) until a maximum length is reached, or the stroke encounters a strong edge in $|\nabla I_1(\cdot)|$. Strokes do not interact, and a stroke may be grown ‘over’ another stroke on the canvas. To prevent the appearance of sampling artifacts due to such overlap, the rendering order of strokes in sequence \mathcal{S}_1 is randomized—in this first video frame.

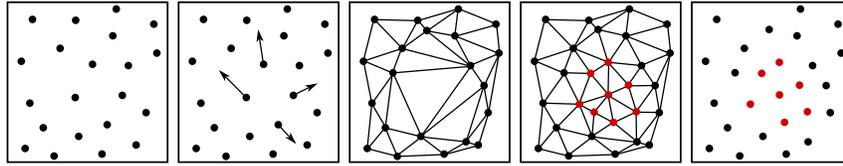


Fig. 13.3 Litwinowicz' mark (stroke) density control algorithm in five steps, from left to right: (1) Initial stroke positions. (2) Here, four strokes move under optical flow. (3) Delaunay triangulation of the stroke points. (4) Red points show new vertices introduced to regularize the density. (5) The updated list of strokes after culling points that violate the closeness test.

The orientation of each stroke is determined in one of two ways, depending on whether $|\nabla I_1(x,y)|$ exceeds a pre-defined threshold. In such cases, the stroke is local to a strong edge and so it is possible to sample a reliable edge orientation $\theta = |\Theta_1(x,y)|$. Otherwise the edge orientation is deemed to be noisy due to a weakly present intensity gradient, and so must be interpolated from nearby strokes that have reliable orientations. The interpolation is performed using a thin-plate spline in Litwinowicz' paper, but in practice any function that smoothly interpolates irregularly spaced samples may be used. In later work by Hays and Essa, for example, radial basis functions are used [25] to fulfil a similar purpose (Section 13.3.1.5).

13.3.1.2 Stroke Propagation

Given an optical flow vector field $\mathcal{O}(\cdot)$ mapping pixel locations in \mathcal{I}_{t-1} to \mathcal{I}_t , all strokes within \mathcal{I}_{t-1} are updated to yield set \mathcal{S}_t where $s_{t-1}^i(x,y) + \mathcal{O}(x,y) \leftarrow s_t^i(x,y)$. If stroke seed points are shifted outside of the canvas boundaries then those strokes are omitted from \mathcal{S}_t . Other stroke attributes within the s_t^i tuple remain constant to inhibit flicker. Note that the rendering order of strokes is randomized only in the first frame, and remains fixed for subsequent frames. This also mitigates against flicker.

The translation process may cause strokes to bunch together, or to become sparsely distributed leaving 'holes' in the painted canvas. A mechanism is therefore introduced to measure and regulate stroke density on the canvas (Fig. 13.3). Stroke density is first measured using a Delaunay triangulation of stroke seed points. Using the connected neighborhood of the triangulation it is straightforward to evaluate, for each stroke, the distance to its nearest stroke. Strokes are sorted by this distance. By examining the head and tail of this sorted list one may identify strokes within the most sparsely and densely covered area of the canvas.

Strokes may be culled from \mathcal{S}_t to thin out areas of the canvas with dense stroke coverage. This can be achieved by deleting strokes present in the tail of the list. Strokes may also be inserted into \mathcal{S}_t . To do so, new strokes are created from the current frame using the process outlined in Section 13.3.1.1. These newly created strokes must be distributed throughout the sequence \mathcal{S}_t to disguise their appearance.

A large block of newly created strokes appearing simultaneously becomes visual salient and causes flicker.

13.3.1.3 Dynamic Distributions

Many subsequent works address the issue of redistributing marks over time, providing various trade-offs between spatial quality and temporal continuity [25, 27, 51]. The general aim of such a ‘Dynamic distribution’ is to maintain a uniform spacing between marks (commonly harnessing the Poisson disk distribution for this purpose) while avoiding sudden appearance or disappearance of marks.

Extending the Poisson disk tiling method of Lagae et al. [37], Kopf et al. [34] propose a set of recursive Wang tiles which allows to generate 2D point distributions with blue noise property in real-time and at arbitrary scale. This approach relies on precomputed tiles, handling 2D rigid motions (zooming and panning inside stills). The subdivision mechanism ensures the continuity of the distribution during the zoom, while the recursivity of the scheme enables infinite zoom.

Vanderhaeghe et al. [51] propose a hybrid technique which finds a more balanced trade-off. They compute the distribution in 2D—ensuring blue noise property—but move the points according to the 3D motion of the scene by following the optical flow. At each frame, the distribution is updated to maintain a Poisson-disk criterion. The temporal continuity is enhanced further by (1) fading appearing and disappearing points over subsequent frames; and (2) allowing points in overly dense regions to slide to close under-sampled regions.

To further reduce flickering artifacts, Lin et al. [39] propose to create a damped system between marks adjacent in space and time, and to minimize the energy of this system. They also try to minimize marks insertions and deletions using two passes. Disoccluded regions emerging during the forward pass are not rendered immediately, but deferred until they reach a sufficient size. Then, they are painted and the gaps are completed by backward propagation. Lin’s damped spring model is discussed in greater detail within Section 13.4.2.3.

The data structures required to manage the attributes and rendering of each individual stroke makes mark-based methods complex to implement and not very well-suited to real-time rendering engines. Nevertheless, Lu et al. [42] proposed a GPU implementation with a simplified stochastic stroke density estimation which runs at interactive framerates but offers fewer guarantees on the point distribution.

To create animated mosaics, Smith et al. [48] and Dalal et al. [19] also rely on the motion flow of the input animation to advect groups of tiles. They propose two policies to spatially localize tiles insertions and deletions at either groups boundaries or groups center. This approach allows coherent group movement and minimizes the flickering of tiles.

13.3.1.4 Frame Differencing for Interactive Painting

The use of general, low-level motion estimation techniques (e.g. optical flow) for video echoes the reliance upon low-level filtering operators by image stylization, circa the 1990s.

Other low-level approaches for painterly video stylization suggested contemporaneously include Hertzmann and Perlin’s frame-differencing approach [28]. In their algorithm, the absolute RGB difference between successive video frames was used as a trigger to repaint (or “paint over”) regions of the canvas that changed significantly; i.e. due to object motion. A binary mask $M(\cdot)$ was generated using a pixel difference thresholded at an empirically derived value T :

$$M(x, y) = |\mathcal{I}_{t-1}(x, y) - \mathcal{I}_t(x, y)| > T \quad (13.3)$$

Strokes seeded at non-zero locations of $M(\cdot)$ are repainted at time t . Flicker is greatly reduced, as only moving areas of the video feed are repainted. Furthermore the computational simplicity of the differencing operation made practical real-time interactive video painting, to create an interactive painterly video experience. This contrasted to optical flow based approaches, which were challenging to compute in real-time due to the limitations in computational power at the time their work was carried out.

A further novelty of Hertzmann and Perlin’s interactive painting system was the use of curved brush strokes to stylize video. This work built upon Hertzmann’s earlier multi-resolution curved stroke painting algorithm for image stylization (discussed in more detail within Chapter 1). Previously Litwinowicz’ approach [40] and similar optical flow based methods [49] had used only short rectangular strokes.

13.3.1.5 Multi-scale Video Stylization with Curved Strokes

Hays and Essa developed a video stylization system fusing the benefits of optical flow, after Litwinowicz [40], with the benefits of coarse-to-fine rendering with curved brush strokes, after Hertzmann [26]. Although experiments exploring this fusion of ideas were briefly reported in [28], this was the first time such a system had been described in detail.

The system of Hays and Essa shares a number of commonalities with Litwinowicz’ original pipeline. A set of strokes \mathcal{C} is maintained as before, and propagated forward in time using optical flow. Strokes are also classified as strong, or not, based on local edge strength and interpolation applied to derive stroke orientations from the strong strokes. However the key to the improved temporal coherence of the approach is the way in which stroke attributes (such as color and orientation) evolve over time. Rather than remaining fixed, or being sampled directly from the video frame, attributes are blended based on their historic values. A particular stroke may have color \mathbf{c}_{t-1} at frame $t - 1$, and might sample a color \mathbf{c}_t from the canvas at frame t . The final color of the stroke \mathbf{c}'_t would be a weighted blend of these two colors:

$$\mathbf{c}'_t = \alpha \mathbf{c}_t + (1 - \alpha) \mathbf{c}_{t-1} \quad (13.4)$$

Or more generally, all stroke attributes would follow a similar blended update, enforcing a smoothed variation in stroke color, orientation, opacity and any other appearance attributes:

$$\mathbf{s}'_t \leftarrow \alpha \mathbf{s}_t + (1 - \alpha) \mathbf{s}'_{t-1} \quad (13.5)$$

Uniquely, Hays and Essa also propose opacity as an additional mark attribute. When adding or removing strokes to preserve stroke density over time, strokes do not immediately appear or disappear. Rather they are faded in, or out, over a period of several frames. This ‘fade-out’ greatly enhances temporal coherence and suppresses the ‘popping’ artifacts that can occur with [40].

Rendering in Hays and Essa’s system follows Hertzmann’s curved brush stroke pipeline, as described in Chapter 1. To decide where to add strokes, areas of the canvas containing no paint are identified and strokes generated at the coarsest level. Strokes are also added at successfully finer layers, local to edges present at the spatial scale of that layer.

Strokes are deleted if they are moved, by the optical flow process, too far from strong edges existing at a particular spatial scale of the pyramid. This prevents the accumulation of fine-scale strokes that tend to clutter the painting.

13.3.2 Texture-based methods

Texture-based approaches are mostly used for continuous textures (canvas, watercolor) or highly structured patterns (hatching). By embedding multiple marks, textures facilitate and accelerate rendering compared to mark-based methods. Textures are generally applied over the entire frame. The challenge is then to deform the texture so that it follows the scene motion while preserving the spatial quality of the original pattern.

13.3.2.1 Bi-directional Flow

A criticism of early optical flow techniques is their tendency to accumulate error and propagate it forward in time, causing instability on longer sequences.

Bousseau et al. [11] apply non-rigid deformations to animate a texture according to the optical flow of a video [4]. This approach extends texture advection methods used in vector field visualization [44] by advecting the texture forward and backward in time to follow the motion field. This *bi-directional* advection allows the method to deal with occlusions where the optical flow is ill-defined in the forward direction but well defined in the backward direction.

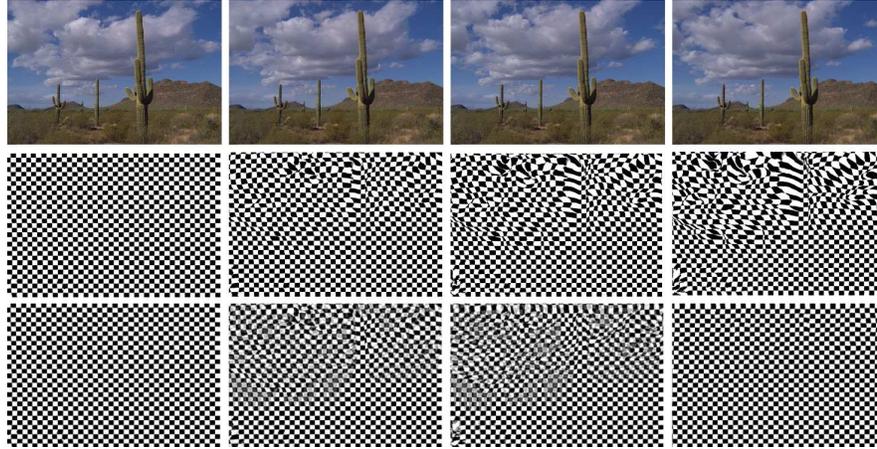


Fig. 13.4 A checkerboard texture (second row) advected along the optical flow of a video (first row) will rapidly be deformed and lose its spatial properties. Bousseau et al. “Video Watercolorization” propagates one instance of the texture forward and the other one backward in time according to the flow field, and alpha-blends them to minimize the distortion (third row). From [11] ©2007 ACM, Inc. Included here by permission.

Rather than placing individual strokes, Bousseau et al. create a watercolor effect by multiplying each video frame \mathcal{I}_t with a global greyscale texture \mathcal{G}_t of identical size to the frame. Pixels in the image and texture field are multiplied place for place to create the stylized frame \mathcal{I}'_t :

$$\mathcal{I}'_t = \mathcal{I}_t(1 - (1 - \mathcal{I}_t)(\mathcal{G}_t - 0.5)) \quad (13.6)$$

where pixels values in both \mathcal{I}_t and \mathcal{G}_t are assumed normalized. This affects a form of alpha-blending of the texture.

The texture in \mathcal{G}_t is a weighted combination of two similarly sized textures, one propagated or “advected” forward in time using the forward flow field, and the other advected back in time. Call these watercolor textures F_t and B_t ; their weights are respectively $\omega_f(t)$ and $\omega_b(t)$:

$$\mathcal{G}_t = \omega_f(t)F_t + \omega_b(t)B_t \quad (13.7)$$

Both fields F_t and B_t are generated from some user supplied continuous texture function. Bousseau et al. do not specify a particular texture function but suggest the pixel intensities should have more or less homogeneous spatial distribution, and the texture should exhibit a reasonably a flat frequency distribution.

The texture field is warped under the respective (forward or backward) flow field to render each frame. Over time, the texture will become distorted due to the motion vectors significantly compressing or stretching the texture. This is detected via a set of heuristics, and new textures initialized for advection periodically or as needed. These detection heuristics are outlined in more detail within their paper. Bousseau

et al. propose an advanced blending scheme that periodically regenerates the texture to cancel distortions and favor at each pixel the advected texture with the least distortion. Suppose the textures are initialized periodically every τ frames. The weights $\omega_f(t)$ and $\omega_b(t)$ for combining the texture fields F_t and B_t are given as:

$$\begin{aligned}\omega_f(t) &= \cos^2\left(\frac{\pi t \bmod \tau}{2\tau}\right) \\ \omega_b(t) &= \sin^2\left(\frac{\pi t \bmod \tau}{2\tau}\right)\end{aligned}\tag{13.8}$$

Although the process of advecting a global single texture forward in time is in common use within scientific visualization domain, Bousseau et al. were the first to introduce this approach for video stylization. Flow fields in video exhibit more frequent discontinuities than are typical in scientific visualization, due to object occlusions. The use of bi-directional advection, rather than simply forward advection, was principally motivated by the desire to suppress temporal incoherence caused by such discontinuities. However the bidirectional advection requires the entire animation to be known in advance, which prevents the use of this method for real-time applications. To overcome this limitation, Kass and Pesare [32] propose to filter a white or band-pass noise. Their recursive filter produces a coherent noise with stationary statistics within a frame (high flatness) and high correlations between frames (high motion coherence). This approach is fast enough for real-time application, but is restricted to isotropic procedural noise and it needs depth information to handle occlusions and disocclusions properly.

13.3.2.2 Coherent Shape abstraction

In order to produce an aesthetically pleasing watercolor effect, Bousseau et al. applied video processing in addition to texture advection, to abstract away some of the visual detail in the scene. This was achieved using morphological operators to remove small-scale details in the frames. Bousseau et al. observed that a binary opening operation (an erosion, following by a dilation) could remove lighter colored objects in the image. The reverse sequence of operations—a binary closure—can remove darker objects. Depending on the scale of the structuring element using in the morphological operations, different scales of object may be abstracted; i.e. removed, or their shapes simplified.

In the Computer Vision literature, the use of morphological scale-space filtering has been well known to produce these kinds of image simplifications. For example, the 1D and 2D *image sieves* developed by Bangham et al. in the late nineties comprise similar morphological operations. Such filtering is particularly effective at image simplification, as angular features such as corners are not ‘rounded off’ as might result using a linear low-pass filter such as successive Gaussian blurring. Indeed, sieves were applied to color imagery several years earlier in 2003 precisely for the purposes of image stylization [3]. Bousseau *et al* were however the first to

extend the use of such morphological filters to coherently stylize video, filtering in 3D (space-time) rather than on a per frame basis [10].

13.4 Video Segmentation for Stylization

In an effort to improve temporal coherence and explore a wider gamut of styles, researchers in the early-mid 2000s began to apply segmentation algorithms to the video stylization problem.

Segmentation is the process of dividing an image into a set of regions sharing some homogeneity property. The implicit assumption in performing such segmentation is that regions should correspond to objects within the scene. However, in practice the homogeneity criteria used in the segmentation are typically defined at a much lower level (e.g. color or texture). For video segmentation, it is desirable to segment each frame not only with accuracy with respect to such criteria, but also with temporal coherence. The boundaries of regions should remain stable (exhibit minimal change) over time, and spurious regions should not appear and disappear.

The stable mid-level representation provided by video segmentation offers two main advantages over low-level flow-based video stylization:

First, rendering parameters may be applied consistently across objects; a common phenomenon in real artwork. Furthermore, users intervention may be incorporated to selectively stylize particular objects [15, 30].

Second, the motion of rendering marks may be fixed to the reference frame of each region as it moves over time. This ensures motion coherence—one of our key criteria for temporal coherence (Section 13.1.2). Rotoscoping and artistic stylization are therefore closely related. Given a coherent video segmentation, one might rotoscope any texture onto the regions for artistic effect, from flat-shaded cartoons to the complex brush stroke patterns of an oil painting. Many image stylization approaches may be applied to video, by considering each region as a stable reference frame upon which to apply the effect [1, 15, 57]. Furthermore, the boundaries of the regions may also be stylized [7, 31, 33].

In essence, a coherent video segmentation enables the coherent parameterization of lines (boundaries) or regions one may wish to stylize. Such a parameterization allows the coherent mapping of textures (or placement of marks) allowing for precise control of a broad range of styles.

13.4.1 Coherent Video Segmentation

Video segmentation is a long-standing research topic in Computer Vision, and a number of robust approaches to coherent video segmentation now exist. However, in the early 2000s, research focused firmly upon image segmentation. Variants of the mean-shift algorithm [17] were very popular. Chapter 7 covers the application



Fig. 13.5 Video Tooning: (a) Space-time mean-shift is used to over-segment the video volume. (b) Volume fragments are grouped by the user, and pre-prepared texture applied to create a cartoon rotoscoped effect. (c) The interiors and bounding contours of region groups may also be rendered. From [55]  2004 ACM, Inc. Included here by permission.

of the EDISON variant of mean-shift to a variety of image stylization tasks, spurred by the early work of DeCarlo and Santella [20]. Around 2004 two complementary approaches were simultaneously developed to extend mean-shift to video for the purpose of video stylization. The *Video Tooning* approach of Wang et al.'s [55] adopts an 3D extension of mean-shift to the space-time video cube (x, y, t) . Collomosse et al.'s *Stroke Surfaces* approach [13, 15] adopts a 2D plus time (2D+t) approach, creating correspondences between regions in independently segmented, temporally adjacent frames.

13.4.1.1 Video Tooning

Mean-shift is an unsupervised clustering algorithm [21]. It is most often applied to image segmentation by considering each pixel as a point in a 5D space (r, g, b, x, y) encoding color and location. The essence of the algorithm is to identify local modes in this feature space, by shifting a window (kernel) toward more densely populated regions of the space. The resulting modes become regions in the video, with pixels local to each mode being labelled to that region.

Extension of this algorithm to a space-time video cube may be trivially performed by adding a sixth dimension to pixel features, encoding time (r, g, b, x, y, t) . Due to differences in the spatial and temporal resolution of video, and the isotropy of typical Mean-shift kernels, this can result in spurious regions manifesting local to movement in the footage. One solution is to add further dimensions to the space, encoding the motion vector (i.e. optical flow) of each pixel however this has the disadvantage of increasing the dimensionality of the feature space, requiring longer videos (i.e. more samples) to cluster effectively. Wang et al.'s contribution was to compensate for the artifacts in a 6D clustering by using an anisotropic kernel during the mean-shift. The scale of the kernel is determined on a per pixel basis by analyzing local variation in color [54].

Figure 13.5 provides a representative video segmentation, demonstrating the tendency of space-time mean-shift to over-segment object. In their Video Tooning system, Wang et al. invite users to group space-time volumes into objects using an



Fig. 13.6 Left: Visualization of the Stroke Surfaces space-time representation [15]. ©2005 IEEE. Included here by permission. Right: Two rendering styles generated by rendering the same stroke surface representation [15].

interactive tool. The resulting regions within each frame may then be rotoscoped with any texture. We discuss a general approach to performing such rotoscoping in Section 13.4.2. In the example of Fig. 13.5, a manually created child’s drawing is rotoscoped on to each region, and composited upon a drawn background.

13.4.1.2 Stroke Surfaces

Due to memory constraints, a space-time video segmentation is practical only for shorter sequences. Furthermore, such methods are prone to over-segmentation especially of small fast moving video objects, resulting in the representation of objects as many disparate sub-volumes. These can require considerable manual intervention to group. An alternative is to segment regions independently within each video frame, and associate those regions over time (a $2D + t$ approach). Independently segmenting frames often yields different region topologies between frames, and many small noisy regions. However the larger video objects one typically wishes to rotoscope exhibit greater stability.

The approach of Collomosse et al. is to associate regions at time t with regions at time $t - 1$ and $t + 1$ using a set of heuristics, e.g. region color, shape, centroid location. These associations form a graph for each region over time, which may be post-processed to remove short cycles and so prune sporadically splitting/merging regions. Only the temporally stable regions remain, forming sub-volumes through the space-time video volume.

At this stage, the space-time volume representation is similar to that of Wang et al. and amenable to rotoscoping (Section 13.4.2). However the temporal coherence of the segmented objects is further enhanced in Collomosse et al.’s pipeline through the formation and manipulation of *stroke surfaces*.

A stroke surface is a partitioning surface separating exactly two sub-volumes. Stroke surfaces are fitted via an optimization process adapted from the Active Contour literature; full details are available in [15]. As a single stroke surface describes the space-time interface between two video objects, the coherence of the corresponding region boundary may be smoothed by smoothing the geometry of the

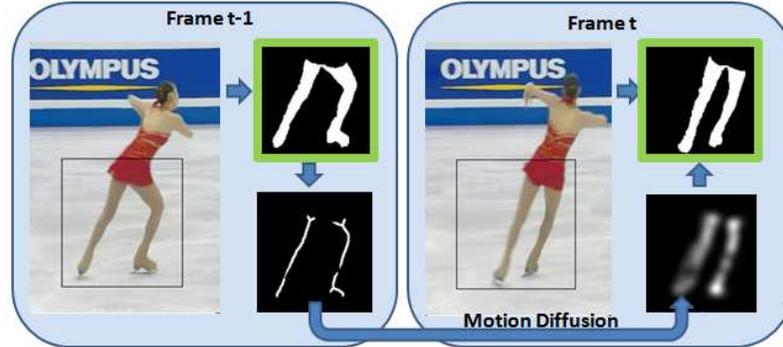


Fig. 13.7 Outline of Wang et al.'s video segmentation algorithm. Multi-label graph cut is applied to each video frame, influenced by region labellings from prior frames. Each labelled region is coded as a binary mask; labels are propagated from prior frames by warping the skeletons of those masked regions via optical flow. The warped skeleton is diffused according to motion estimation confidence and the diffused masks of all regions form a probability density function over labels that is used as a prior in the multi-label graph cut of the next frame.

stroke surface in the temporal direction. Temporally slicing the surface produces a series of smoothed regions which may be used, as in Wang et al., for rotoscoping.

13.4.1.3 Region Tracking

Both of the above video adaptations of mean-shift above require space-time processing, either for initial segmentation [55] or region association pruning [15]. For online processing (e.g. for streaming video, or to avoid memory overhead) it may be desirable to segment video progressively (on a per-frame basis) based on information propagated forward from prior frames. A number of robust systems have emerged in recent years for tracking a binary matte through video [2], providing single object segmentation suitable for stylizing a single object. However a progressive segmentation algorithm capable of tracking multiple object labels through video is necessary to perform stylization of the entire scene.

One such approach, recently proposed by Wang et al., harnesses a multi-label extension of the popular graph-cut algorithm to robustly segment video [56]. Wang et al.'s solution is to perform a multi-label graph cut on each frame of video, using information both from the current frame and from prior information propagated forward from previous frames.

Given a segmentation of I_{t-1} , the region map is skeletonized to produce a set of pixels central to each region considered to be labelled with high confidence. These labelled pixels are warped to new positions in I_t , under a dense optical flow fields computed between the two frames. These labels are used to initialize the graph cut on the next frame, alongside models of color and texture that are incrementally learned over time from the labelled image regions.

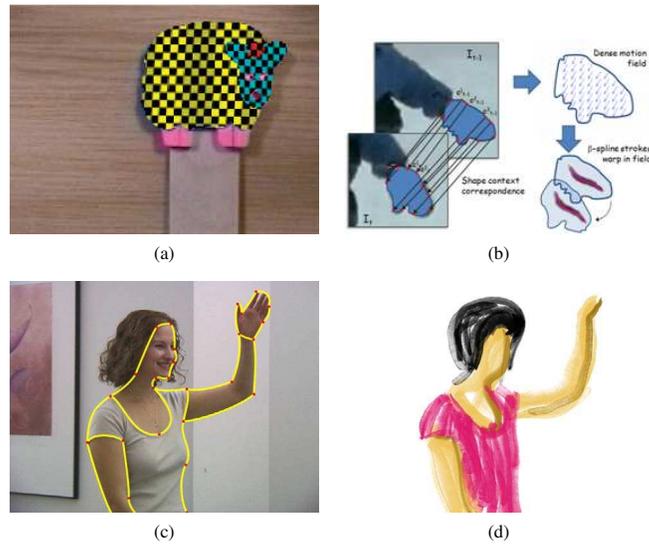


Fig. 13.8 Three region rotoscoping strategies: (a) In Stroke Surfaces [15] regions were treated as rigid bodies, mapped using affine transformations. (c) Dynamic programming was used by Agarwala et al. [1] to assign points within regions to moving control points on the bounding contour, tracked from (b) an initial hand-drawn user contour. (d) Wang et al. [57] perform affine registration of regions using shape contexts and then interpolate a dense motion field using Poisson filling. From [15] ©2005 IEEE, [57] ©2011 Elsevier, and [1] ©2004 ACM, Inc. Included here by permission.

As discussed in Section 13.3.1, dense optical flow fields are often poorly estimated and so this propagation strategy can fail in the longer term. To compensate, Wang et al. perform a diffusion of each pixel in I_{t-1} over multiple pixels in I_t ; each pixel obtains a probability of belonging to a particular region label. In practice this is achieved using a Gaussian distribution centered upon the optical flow-derived location of the point in I_t . The standard deviation (i.e. spread) of the Gaussian is modulated to reflect the confidence in the optical flow estimate, which can in turn be estimated from the diversity of motion vector directions local to the point. Figure 13.7 outlines the process.

13.4.2 Rotoscoping Regions

Once a coherent video segmentation has been produced, marks (such as brush strokes) may be fixed to each region. In all cases it is necessary to first establish correspondence between the boundary of a region across adjacent frames, usually encoded via the control points of a contour. The dense motion field inside the region is then deduced from the motion vectors established between these control points.

There are a number methods in the literature to model this dense motion, for example treating the region as:

1. a rigid body, e.g. moving under an affine motion model deduced from the control points of the region boundary [15] (Fig. 13.8(a)).
2. a deforming body, with marks adopting motion of the closest control point on the region boundary [1] (Fig. 13.8(b)).
3. a deforming body, with marks moving under a motion model that minimizes discontinuities within the motion field within region, whilst moving with the control points of the region boundary. [57] (Fig. 13.8(c)).

Although many shape correspondence techniques exist, a robust solution to inter-frame boundary matching is commonly to use Shape Contexts [5]. Suppose using Shape Contexts, or otherwise, we obtain a set of n control points $C' = \{\mathbf{c}'_1, \mathbf{c}'_2, \dots, \mathbf{c}'_n\}$ at time t , and the corresponding points $C = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_n\}$ at the previous frame $t - 1$. Pixels within the region at time $t - 1$ are written $P_{t-1} \{\mathbf{p}_{1..n}\}$ and we wish to determine their locations $P_t = \{\mathbf{p}'_{1..n}\}$ at time t . We wish to obtain the dense motion field V_{t-1} responsible for this shift:

$$V_{t-1}(\mathbf{p}_i) = \mathbf{p}'_i - \mathbf{p}_i \quad (13.9)$$

In this chapter we cover one rigid (1) and deforming (3) solution.

13.4.2.1 Rigid motion

If operating under the rigid model we can simply write the i^{th} control point location $\mathbf{c}'_i = \mathbf{A}\mathbf{c}_i$ in homogeneous form, expanded as:

$$\begin{bmatrix} c'_{i,x} \\ c'_{i,y} \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_{i,x} \\ c_{i,y} \\ 1 \end{bmatrix} \quad (13.10)$$

The 3×3 affine transformation matrix \mathbf{A} is deduced between C and C' (contain the respective control points in homogeneous form) as a least squares solution:

$$\mathbf{A} = \mathbf{C}'\mathbf{C}^T (\mathbf{C}\mathbf{C}^T)^{-1} \quad (13.11)$$

The dense vector field (Eq. (13.9)) is created by applying the resulting transformation to all pixels P_{t-1} within the region at $t - 1$, i.e. $\mathbf{p}'_i = \mathbf{A}\mathbf{p}_i$.

13.4.2.2 Smooth deformation

A smooth dense motion field may be extrapolated from control points, for example using a least squares fitting scheme that minimizes the Laplacian of the motion

vectors. This is achieved by solving an approximation to Poisson’s equation; commonly referred to in Graphics as *Poisson in-filling* after Pérez and Blake’s initial application of the technique to texture infilling in 2003 [46].

Recall we wish to deduce a dense motion field $\mathcal{V}_{t-1}(\mathbf{p}_i)$ defining the motion of for all pixels P_{t-1} . However we know only irregularly and sparsely placed points in this field $\mathcal{V}_{t-1}(\mathbf{c}_i) = \mathbf{c}'_i - \mathbf{c}_i$. Call this known, sparse field V and the dense field \mathcal{V} , dropping the time subscript for brevity.

We seek the dense field \mathcal{V}_P over all pixel values $\Omega \in \mathfrak{R}^2$, such that $\mathcal{V}(c) = V(c), \forall c \in C_t$ and minimizing:

$$\operatorname{argmin}_{\mathcal{V}} \Sigma \Sigma_P (\nabla \mathcal{V} - V)^2 \text{ s.t. } \mathcal{V}|_{\delta P} = V|_{\delta P} \quad (13.12)$$

i.e. $\Delta \mathcal{V} = 0$ over P , s.t. $\mathcal{V}|_{\delta P} = V|_{\delta P}$. This is Poisson’s equation and is practically solvable for our discrete field as follows.

The desired 2D motion vector field $\mathcal{V}(P)$ is first split into its component scalar fields $\mathcal{V}_x(P)$ and $\mathcal{V}_y(P)$. For each scalar field, e.g. that of the x component, $\mathcal{V}_x(p_i) = v_i$, where $i = [1, n]$ we form the following $n \times n$ linear system using ‘known’ pixels $V(p) = v_i$ (i.e. at the control points, below denoted $k_i = v_i$) and unknown pixels (i.e. everywhere else):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots \\ 0 & -1 & \vdots & -1 & 4 & -1 & \dots & -1 & 0 \\ \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_i \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ k_i \\ \vdots \\ 0 \end{bmatrix} \quad (13.13)$$

The system can be solved efficiently by a sparse linear solver such as LAPACK, yielding values for all v_i and so the dense scalar field—in our example for the x component of $\mathcal{V}(P)$. Repeating this for both x and y components yields the dense motion field $\mathcal{V}(P)$ that may be used to move strokes or other rendering marks within the region as Fig. 13.8 illustrates.

A variety of other smoothness constraints have been explored for region deformation, including thin-plate splines [39] and weighted combinations of control point motion vectors [1].

13.4.2.3 Spring-based dampening

Once strokes, or similar marks, have been propagated to their new positions under the chosen deformation model, their position may be further refined. Inaccuracy in the region segmentation, and occlusions, can cause sporadic jumps in stroke posi-

tion. Lin et al. showed that this can be successfully mitigated by dampening stroke motion using a simple string model [39].

Strokes are connected to their space-time neighbors (i.e. other strokes present within a small space-time range) using springs. Using the notation $S_{i,t}$ to denote the putative position of stroke i at time t —and $S_{i,t}^0$ to denote the initial (i.e. post-deformation) position of the strokes—the energy of the spring system $E(S)$ is given as a weighted sum:

$$E_{system}(S) = E_1(S) + 2.8E_2(S) + 1.1E_3(S) \quad (13.14)$$

In their formulation the first term $E_1(\cdot)$ indicates spatial deviation from the initial position:

$$E_1(S) = \sum_{\forall S_{i,t}} |S_{i,t} - S_{i,t}^0|^2 \quad (13.15)$$

The second term enforces temporal smoothness in position:

$$E_2(S) = \sum_{\forall S_{i,t}} (S_{i,t-1} + 2S_{i,t} + S_{i,t+1})^2 \quad (13.16)$$

and the third term enforces proximity to neighboring strokes, which are denoted by the set $\mathcal{N}_{i,t}$ for a given stroke $S_{i,t}$:

$$E_3(S) = \sum_{\forall S_{i,t}} \sum_{\forall p \times q \in \mathcal{N}_{i,t}} \delta(p, q)^2 \quad (13.17)$$

here $\delta(\cdot)$ is a function evaluating smaller for strokes of similar size and spatial position. The system is minimized via an iterative Levenburg-Marquadt optimization over all strokes in the video.

13.4.3 Rotoscoping Boundaries for Stylized Lines

Parameterized lines allow the use of texture mapping to produce dots and dashes or to mimic paint brushes, pencil, ink and other traditional media. The correspondence of region boundaries outlined in Section 13.4.2 enables such a parameterization to be established in a temporally coherent fashion. This opens the door to a wide variety of artistic line stylization techniques.

There are two simple policies for texturing a path. The first approach, that we call the *stretching policy* (Fig. 13.9(a)), stretches or compresses the texture so that it fits along the path a fixed number of times. As the length of the path changes, the texture deforms to match the new length. The second approach, called the *tiling policy* (Fig. 13.9(b)), establishes a fixed pixel length for the texture, and tiles the path with as many instances of the texture as can fit. Texture tiles appear or disappear as the length of the path varies.

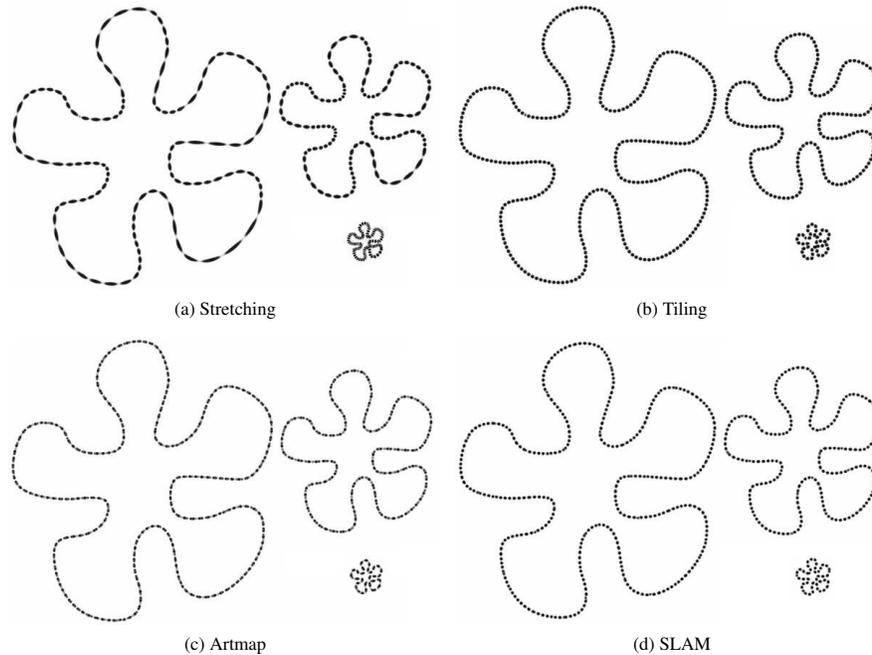


Fig. 13.9 Four strokes texture mapping policies. (a) Stretching ensures coherence during motion but deforms the texture. Conversely (b) tiling perfectly preserves the pattern, but produces sliding when the path is animated. (c) Artmap [33] avoids both problems at the price of fading artifacts because the hand-drawn texture pyramid is usually too sparse. (d) Self-Similar Line Artmap [7] solves this problem. From [6] ©2012 Blackwell Publishing. Included here by permission.

The two policies are appropriate in different cases. The tiling policy is necessary for textures that should not appear to stretch, such as dotted and dashed lines. Because the tiling policy does not stretch the texture, it is also usually preferred for still images. Under animation, however, the texture appears to slide on or off the ends of the path similarly to the shower door effect (Fig. 13.9(b)). In contrast, the stretching policy produces high motion coherence under animation, but the stroke texture loses its character if the path is stretched or shrunk too far (Fig. 13.9(a)). Kalnins et al. [31] combine these two policies with 1D texture synthesis using Markov random field to reduce repetitions.

The *artmap* method [33] (Fig. 13.9(c)) is an alternative to the simple stretching and tiling policies. This method uses texture pyramid, where each texture has a particular target length in pixels. At each frame, the texture with the target length closest to the current path length is selected and drawn. This mechanism ensures that the brush texture never appears stretched by more than a constant factor (often $2\times$).

Nevertheless, stretching artifacts still appear when the length of the path extends beyond the length of the largest texture in the artmap. Fading or popping artifacts can also occur during transitions between levels of the texture pyramid. Finally, a major

drawback of the artmap method resides in the manual construction of the texture pyramid. Artists need to draw each level of the pyramid, tacking care of the coherence across levels. As a result, current artmap implementations such as “Styles” in Google SketchUp® use as few as four textures, which accentuates artifacts during transitions.

To reduce transition artifacts and automate the creation process, Bénard et al. [7] propose *Self-Similar Line Artmap (SLAM)*, an example-based artmap synthesis approach that generates an arbitrarily dense artmap based on a single exemplar. Their synthesis not only guarantees that each artmap level blends seamlessly into the next, but also provides continuous infinite zoom by constructing a self-similar texture pyramid where the last level of the pyramid is included in the first level. The synthesis takes a few minutes as a pre-process and provides good results for many brush textures, although fine details might be lost for very complex patterns.

13.4.4 Painterly Rotoscoping Environments

Early papers recognizing the links between coherent stylization and rotoscoping [15, 55] used the techniques of Section 13.4.2 to directly fix marks to regions, enabling those marks to match-move video content.

However recent systems adopt a more indirect approach, harnessing the region correspondence to transform data fields over time that drive the placement of strokes. A popular choice of field is the intensity gradient (equivalently, the edge orientation) field, which typically plays an important role in the placement of individual marks such as strokes [26]. Ensuring coherence in the transformed orientation field ensures coherence in the final rendering.

The advantage in deferring stroke placement until the rendering of individual frames (rather than placing strokes on the first frame, and subsequently moving them) is that more frame-specific information can be taken into account during stylization. This can lead to a broader range of styles as Kagaya et al. demonstrated in their multi-style video painting system [30]. It can also lead to greater control, as when framed in an interactive setting, users can manipulate the fields and parameters used to create particular effects on particular objects. For example, O’Donovan and Hertzmann’s AniPaint system enable the rotoscoping of regular curved strokes onto regions, along with “guide strokes” that influence the orientation of further strokes placed on the frame [45]. Kagaya et al.’s multi-style rendering system [30] enables the interactive specification of tensor fields at key frames that are used to form elegant brush strokes in a manner reminiscent of the Line Integral Convolution methods used to create a painterly effect in the filtering approaches of Chapter 5. These key framed fields are smoothly interpolated over time using a space-time extension of the heat diffusion process outlined in Chapter 6, which like the smooth deformation method of Section 13.4.2.2, is based upon a Laplacian smoothing constraint.

The incorporation of user interaction into the video stylization pipeline also offers the possibility of correcting the initial automated video segmentations provided

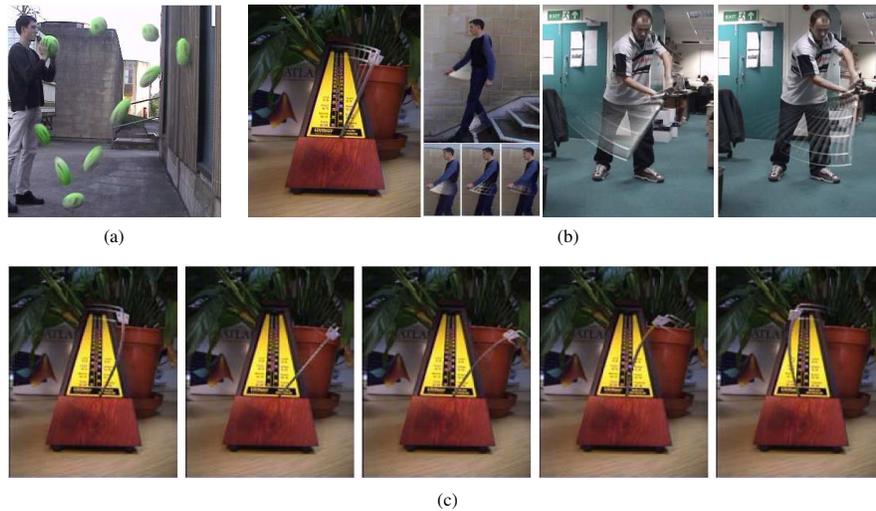


Fig. 13.10 Motion stylization in Video using [14]. (a) Squash and Stretch effect with flight and collision emphasis. (b) Augmentation of motion using blur and speed-lines. (c) Deformation emphasizing inertia. From [14].

by Computer Vision algorithms. These corrections remain inevitable as the general video segmentation problem is far from solved.

13.5 Segmentation for Motion Stylization

Video analysis at the region level enables not only consistent rendering within objects, but also the analysis of object motion. This motion may then be stylized using a variety of motion emphasis cues borrowed from classical animation, such as mark-making (speed-lines, ghosting or ‘skinning’ lines), deformation and distortion as a function of motion, and alteration of timing. In his influential paper, Lasseter [38] describes many such techniques, introducing them to the Computer Graphics community. However his paper presents no algorithmic solutions to the synthesis of motion cues. Hsu et al. [29] also identify depiction of motion as important, though the former discusses the effect of such cues on perception. Both studies focus only the placement of speed-lines through user-interactive processes.

Automated methods to generate speed-lines in video require camera motion compensation, as the camera typically pans to keep moving objects within frame. This can be approximated by estimating inter-frame homographies. Trailing edges of object may be corresponded over time to yield a set of trails, which having been warped to compensate for the camera motion-induced homography, may be smoothed and visualized as speed-lines. A collection of heuristics derived from animation practice were presented in [14] and optimized against to obtain well placed speed-lines. The

trailing edges of objects may also be rendered to produce ghosting or ‘skinning’ effects, which when densely packed can additionally serve as motion blur. The visual nature of speed-lines has also led to their application in motion summarization via reverse story-boarding [22] visualized using a mosaic constructed from the video frames. Cheney et al. [12] presented the earliest work exploring automated deformation of objects to emphasize motion. This work introduced the ‘squash and stretch’ effect, scaling 3D objects along their trajectories and applying the inverse scale upon surface impact. A similar effect was applied to 2D video in [14], where a nonuniform scaling of the object was performed within a curvilinear basis set established using a cubic spline fitted to the object’s trajectory. Other distortions were explored within this basis set including per-pixel warping of the object according to velocity and acceleration; giving rise to the visual effect of emphasizing drag or inertia. Figure 13.10 illustrates the gamut of effects available in this single frame-work.

A layered approach to deformation was described by Liu et al. [41]. Video frames were segmented into distinctly moving layers, using unsupervised clustering of motion vectors (followed by optional manual correction). The layers were then distorted according to an optical flow estimate of pixels within each layer. Texture infilling algorithms were applied to fill holes, and a priority ordering assigned to layers to resolve conflicts when warped layers overlapped post-deformation. Animators frequently manipulate the timing and trajectory of object motion to emphasize an action; for example, a slight move backward prior to a sprint forwards. This effect is referred to as *anticipation* or *snap* in the animation. The automated introduction of snap into video objects was described in [16], learning an articulated model of the moving object e.g. a walking person, by observing the rigidity and inter-occlusion of moving parts in that object. The joint angles parameterizations were manipulated to exhibit a small opposing motion in proportion to the scale of each movement made. A more general motion filtering model based of region deformation, rather than articulated joints, was described in [53].

13.6 Discussion and Conclusion

This chapter illustrates the large amount of work addressing the problem of temporally coherence video stylization, and highlights a number of limitations that represent interesting directions for future research. The requirements implied by temporal coherence are both contradictory and ill-defined, which in our sense is one of the challenges of this field. In order to facilitate the concurrent analysis of existing methods in this Chapter, we proposed a formulation of the temporal coherence problem in terms of three goals: spatial quality, motion coherence and temporal continuity. Table 13.1 summarizes the relative trade-offs against these criteria exhibited by the main families of technique surveyed in this Chapter. In addition, we compare against three additional criteria which should be considered when selecting appro-

Table 13.1 Summary of the trade-offs made by the different families of methods surveyed in this Chapter.

		Spatial quality	Coherent motion	Temporal continuity	Style variety	Complexity	Footage diversity
Naïve	Static marks	++	--	++	++	++	++
	Advection	--	++	++	++	-	++
	Random marks	++	--	--	++	++	++
Filtering	Chapter 5	++	++	+	-	++	++
Opt. Flow	Mark-based	++	++	-	+	-	+
	Texture-based	+	++	+	+	-	+
Segmentation	3D (x,y,z)	++	++	+	+	--	--
	2D+t	++	++	+	+	--	-

appropriate methods. These are the overall complexity of implementation, the variety of styles that may be simulated, and the diversity of footage that may be processed.

Of these criteria it is arguably hardest to precisely define the goal of *spatial quality*. To go further, we are convinced that human perception should play a greater role in evaluating video stylization work (see Chapter 15 for an in-depth survey of evaluation approaches in NPR). Our spatial quality criterion relates to the perception of each frame as being somehow ‘hand-made’. Temporal continuity involves *visual attention* which encapsulates explain human visual sensitivity to flicker and ‘popping’. Motion coherence could benefit from studies on *motion transparency* to describe more precisely sliding effects. Some studies have been done in the context of 3D scenes animations [8, 9] and could be use as a starting point for evaluating stylized videos. Beyond the evaluation of temporal coherence, these connections could also help drive coherent video stylization algorithms. Quantitative measurements could be deduced from perceptual evaluations, paving the way to the formulation of temporal coherence as a numerical optimization problem. Such a formulation would give users precise control on the different goals of temporal coherence.

The coherent stylization of video footage remains an open challenge. This is primarily because the coherent movement of marks with video content requires the accurate estimation of video content motion. This is currently an unsolved problem in Computer Vision, and is likely to remain so in the near-term. Consequently the more successful solutions, and arguably the more aesthetically compelling output, has resulted from semi-automated solutions that require user interaction. Such system enable both the correction the stylization process, but more fully embrace the user interaction to enable intuitive and flexible control over the stylization process.

Currently video stylization algorithms are caught in a compromise between robustness and generality of style. Low-level motion estimation based on flow can produce a reasonable motion estimation over most general video, but this is frequently noisy because each pixel may potentially be estimated with a different motion vector. Although modern flow estimation algorithms seek to preserve spatial coherence in the motion vector estimates, in practical video it is common to see inconsistent motion estimation within a single object. On the other hand, Mid-level motion estimation based on video segmentation can ensure consistency within

objects by virtue of their operation—delimited the boundaries of objects through coherent region identification. However such methods trade this robustness for the inability to deal with objects than cannot be easily delineated such as hair, water, smoke, and so on. The treatment of stylization as a rotoscoping problem in mid-level framework is attractive, as it allows easy generalization of image-based techniques to video, and the creation of aggressively stylized output such as cartoons. This leads to greater style diversity with these techniques, versus flow-based techniques (and non-linear filtering techniques covered in Chapter 5) that have so far been limited to painterly effects. An open challenge in the field is to somehow combine the benefits of these complementary approaches, perhaps by finding a way to fuse both into a common framework to reflect the mix of object types within typical video footage. Regarding region-based video stylization, and rotoscoping more generally, the region deformation models currently considered in the literature (e.g., those of Section 13.4.2) are quite basic. Region boundaries may deform naturally, or due to scene occlusion, yet there is no satisfactory method for discriminating between, and reacting to, these different causes of shape change.

Despite these shortcomings, stylized video is featuring increasingly within the creative industries within movies (e.g., “Waking Life”, “Sin City”, “A Scanner Darkly”), TV productions, advertisements and games. The field should strive to work more closely with the end-users of these techniques. If user interaction and creativity will remain within the stylization work-flow for some time, then collaboration with Creatives and with Human Factors researchers may prove at least as fruitful a research direction as raw algorithmic development, and would prove valuable in evaluating the temporal coherence of algorithms developed.

References

1. Agarwala, A., Hertzmann, A., Salesin, D.H., Seitz, S.M.: Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics* **23**, 584–591 (2004)
2. Bai, X., Wang, J., Simons, D., Sapiro, G.: Video SnapCut: Robust Video Object Cutout Using Localized Classifiers. *ACM Trans. Graph.* **28**(3), 1–11 (2009)
3. Bangham, J.A., Gibson, S.E., Harvey, R.: The art of scale-space. In: *Proc. BMVC*, pp. 569–578 (2003)
4. Beauchemin, S.S., Barron, J.L.: The computation of optical flow. *ACM Computing Surveys* **27**(3), 433–466 (1995)
5. Belongie, S., Malik, J., Puzicha, J.: Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(4), 509–522 (2002)
6. B enard, P., Bousseau, A., Thollot, J.: Temporal Coherence for Stylized Animation. *Computer Graphics Forum* **30**(8), 2367–2386 (2012)
7. B enard, P., Cole, F., Golovinskiy, A., Finkelstein, A.: Self-similar texture for coherent line stylization. In: *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*, p. 91. ACM press, Annecy, France (2010)
8. B enard, P., Lagae, A., Vangorp, P., Lefebvre, S., Drettakis, G., Thollot, J.: A Dynamic Noise Primitive for Coherent Stylization. *Computer Graphics Forum* **29**(4), 1497–1506 (2010)

9. Bénard, P., Thollot, J., Sillion, F.: Quality Assessment of Fractalized NPR Textures: a Perceptual Objective Metric. In: Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization, pp. 117–120. ACM press, Chania, Greece (2009)
10. Bousseau, A., Kaplan, M., Thollot, J., Sillion, F.X.: Interactive watercolor rendering with temporal coherence and abstraction. In: Proc. NPAR, pp. 141–149 (2006)
11. Bousseau, A., Neyret, F., Thollot, J., Salesin, D.: Video watercolorization using bidirectional texture advection. *ACM Trans. Graph.* **26**(3), 104:1–7 (2007)
12. Cheney, S., Pingel, M., Iverson, R., Szymanski, M.: Simulating cartoon style animation. *Proc. NPAR* pp. 133–138 (2002)
13. Collomosse, J., Rowntree, D., Hall, P.M.: Stroke surfaces: A spatio-temporal framework for temporally coherent nonphotorealistic animations. Tech. Rep. CSBU-2003-01, University of Bath, UK (2003). URL <http://opus.bath.ac.uk/16858/>
14. Collomosse, J., Rowntree, D., Hall, P.M.: Video analysis for cartoon-style special effects. In: Proc. BMVC, pp. 749–758 (2003)
15. Collomosse, J., Rowntree, D., Hall, P.M.: Stroke surfaces: Temporally coherent non-photorealistic animations from video. *IEEE Trans. Vis. Comput. Graphics* **11**(5), 540–549 (2005)
16. Collomosse, J.P., Hall, P.M.: Video motion analysis for the synthesis of dynamic cues and Futurist art. *Graphical Models* **5**(68) (2006)
17. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.* **24**(5), 603–619 (2002)
18. Criminisi, A., Sharp, T., Rother, C., Pérez, P.: Geodesic image and video editing. *ACM Transactions on Graphics* **29**(5), 134:1–134:15 (2010)
19. Dalal, K., Klein, A.W., Liu, Y., Smith, K.: A spectral approach to NPR packing. In: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering, pp. 71–78. ACM (2006)
20. DeCarlo, D., Santella, A.: Stylization and abstraction of photographs. In: Proc. SIGGRAPH, pp. 769–776 (2002)
21. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory* **21**, 32–40 (1975)
22. Goldman, D.B., Curless, B., Salesin, D., Seitz, S.M.: Schematic storyboarding for video visualization and editing. *ACM Trans. Graph.* **25**(3), 862–871 (2006)
23. Green, S., Salesin, D., Schofield, S., Hertzmann, A., Litwinowicz, P., Gooch, A., Curtis, C., Gooch, B.: Non-photorealistic rendering. In: SIGGRAPH courses (1999)
24. Haeblerli, P.: Paint by numbers: Abstract image representations. In: Proc. SIGGRAPH, pp. 207–214 (1990)
25. Hays, J., Essa, I.: Image and video based painterly animation. In: Proc. NPAR, pp. 113–120 (2004)
26. Hertzmann, A.: Painterly rendering with curved brush strokes of multiple sizes. In: Proc. SIGGRAPH, pp. 453–460 (1998)
27. Hertzmann, A.: Paint by relaxation. In: Computer Graphics International, pp. 47–54. IEEE Computer Society, Hong-Kong (2001)
28. Hertzmann, A., Perlin, K.: Painterly rendering for video and interaction. In: Proc. NPAR, pp. 7–12 (2000)
29. Hsu, S.C., Lee, I.H.H., Wiseman, N.E.: Skeletal strokes. In: Proc. UIST, pp. 197–206 (1993). DOI 10.1145/168642.168662
30. Kagaya, M., Brendel, W., Deng, Q., Kesterson, T., Todorovic, S., Neill, P.J., Zhang, E.: Video painting with space-time-varying style parameters. *IEEE Trans. Vis. Comput. Graphics* **17**(1), 74–87 (2011)
31. Kalnins, R.D., Markosian, L., Meier, B.J., Kowalski, M.A., Lee, J.C., Davidson, P.L., Webb, M., Hughes, J.F., Finkelstein, A.: WYSIWYG NPR: drawing strokes directly on 3D models. In: Proceedings of SIGGRAPH 2002, vol. 21, p. 755. ACM Press, San Antonio, USA (2002)
32. Kass, M., Pesare, D.: Coherent noise for Non-Photorealistic Rendering. *ACM Transactions on Graphics* **30**, 30:1–30:6 (2011)

33. Klein, A.W., Li, W., Kazhdan, M.M., Corr a, W.T., Finkelstein, A., Funkhouser, T.A.: Non-photorealistic virtual environments. In: *Proceedings of SIGGRAPH 2000*, pp. 527–534. ACM Press, New Orleans, USA (2000)
34. Kopf, J., Cohen-Or, D., Deussen, O., Lischinski, D.: Recursive Wang tiles for real-time blue noise. *ACM Trans. Graph.* **25**(3), 509–518 (2006)
35. Kyprianidis, J.E.: Image and video abstraction by multi-scale anisotropic Kuwahara filtering. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering*, pp. 55–64. ACM, New York, NY, USA (2011)
36. Kyprianidis, J.E., Kang, H.: Image and video abstraction by coherence-enhancing filtering. *Computer Graphics Forum* **30**(2), 593–602 (2011)
37. Lagae, A., Dutr , P.: A procedural object distribution function. *ACM Transactions on Graphics* **24**(4), 1442–1461 (2005)
38. Lasseter, J.: Principles of traditional animation applied to 3D computer animation. In: *Proc. SIGGRAPH*, vol. 21, pp. 35–44 (1987)
39. Lin, L., Zeng, K., Lv, H., Wang, Y., Xu, Y., Zhu, S.C.: Painterly animation using video semantics and feature correspondence. In: *Proc. NPAR*, pp. 73–80 (2010)
40. Litwinowicz, P.: Processing images and video for an Impressionist effect. In: *Proceedings of SIGGRAPH 97*, pp. 407–414. ACM Press, Los Angeles, USA (1997)
41. Liu, C., Torralba, A., Freeman, W., Durand, F., Adelson, E.H.: Motion magnification. *ACM Trans. Graph.* **24**(3), 519–526 (2005)
42. Lu, J., Sander, P.V., Finkelstein, A.: Interactive painterly stylization of images, videos and 3D animations. In: *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, vol. 26, pp. 127–134. ACM press, Washington, USA (2010)
43. Meier, B.J.: Painterly rendering for animation. In: *Proc. SIGGRAPH*, pp. 477–484 (1996). DOI 10.1145/237170.237288. URL dl.acm.org/citation.cfm?id=237288
44. Neyret, F.: Advected Textures. In: *Proceedings of Eurographics / SIGGRAPH Symposium on Computer Animation*, pp. 147–153. Eurographics Association, San Diego, USA (2003)
45. O’Donovan, P., Hertzmann, A.: AniPaint: Interactive painterly animation from video. *IEEE Trans. Vis. Comput. Graphics* **18**(3), 475–487 (2012)
46. Perez, P., Gangnet, A., Blake, A.: Poisson Image Editing. In: *Proc. ACM SIGGRAPH*, pp. 313–318 (2003)
47. Schwarz, M., Stamminger, M.: On predicting visual popping in dynamic scenes. In: *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization*, p. 93. ACM Press, Chania, Greece (2009)
48. Smith, K., Liu, Y., Klein, A.: Animosacs. In: *Proc. SCA*, pp. 201–208 (2005)
49. Szir nyi, T., T th, Z., Figueiredo, M., Zerubia, J., Jain, A.: Optimization of paintbrush rendering of images by dynamic MCMC methods. In: *Proc. EMMCVPR*, pp. 201–215 (2001)
50. Treavett, S.M.F., Chen, M.: Statistical techniques for the automated synthesis of non-photorealistic images. In: *Proc. EGUK*, pp. 201–210 (1997)
51. Vanderhaeghe, D., Barla, P., Thollot, J., Sillion, F.: Dynamic point distribution for stroke-based rendering. In: *Proceedings of the 18th Eurographics Symposium on Rendering 2007*, pp. 139–146. Eurographics Association, Grenoble, France (2007)
52. Vergne, R., Vanderhaeghe, D., Chen, J., Barla, P., Granier, X., Schlick, C.: Implicit Brushes for stylized Line-based rendering. *Computer Graphics Forum* **30**, 513–522 (2011)
53. Wang, J., Drucker, S.M., Agrawala, M., Cohen, M.F.: The cartoon animation filter. *ACM Trans. Graph.* **25**(3), 1169–1173 (2006)
54. Wang, J., Thiesson, B., Xu, Y., Cohen, M.F.: Image and video segmentation by anisotropic kernel mean shift. In: *Proc. ECCV*, pp. 238–249 (2004). DOI 10.1007/978-3-540-24671-8_19
55. Wang, J., Xu, Y., Shum, H.Y., Cohen, M.F.: Video tooning. *ACM Trans. Graph.* **23**(3), 574 (2004)
56. Wang, T., Collomosse, J.: Progressive Motion Diffusion of Labeling Priors for Coherent Video Segmentation. *IEEE Transactions on Multimedia* **14**(2), 389–400 (2012)

57. Wang, T., Collomosse, J., Hu, R., Slatter, D., Greig, D., Cheatle, P.: Stylized ambient displays of digital media collections. *Computers & Graphics* **35**(1), 54 – 66 (2011). DOI 10.1016/j.cag.2010.11.004
58. Willats, J., Durand, F.: Defining pictorial style: Lessons from linguistics and computer graphics. *Axiomathes* **15**, 319–351 (2005)
59. Winnemöller, H., Olsen, S., Gooch, B.: Real-time video abstraction. In: Proc. SIGGRAPH, pp. 1221–1226 (2006)
60. Yantis, S., Jonides, J.: Abrupt visual onsets and selective attention: evidence from visual search. *Journal of Experimental Psychology: Human Perception and Performance* **10**(5), 601–621 (1984)