Stroke Surfaces: Temporally Coherent Artistic Animations from Video

John P. Collomosse, David Rowntree, and Peter M. Hall

Abstract—The contribution of this paper is a novel framework for synthesizing nonphotorealistic animations from real video sequences. We demonstrate that, through automated mid-level analysis of the video sequence as a spatiotemporal volume—a block of frames with time as the third dimension—we are able to generate animations in a wide variety of artistic styles, exhibiting a uniquely high degree of temporal coherence. In addition to rotoscoping, matting, and novel temporal effects unique to our method, we demonstrate the extension of static nonphotorealistic rendering (NPR) styles to video, including painterly, sketchy, and cartoon shading. We demonstrate how this novel coherent shading framework may be combined with our earlier motion emphasis work to produce a comprehensive "Video Paintbox" capable of rendering complete cartoon-styled animations from video clips.

Index Terms—Artistic rendering, video-based NPR, stroke surfaces, video paintbox, rotoscoping.

1 Introduction

Processing images into artwork is a significant and demanding area within nonphotorealistic rendering (NPR). In this paper, we describe "Stroke Surfaces"—a novel framework for the creation of nonphotorealistic animations from video clips.

The production of temporally coherent animations from video is a longstanding problem in NPR. Static image-based NPR techniques do not generalize easily to video, and painting video frames individually results in a distracting flickering (swimming) within the video sequence. This paper contributes a video-based NPR solution that can produce animations free from such artifacts. This enables the animator to reintroduce some form of controlled incoherence; for example, object outlines can pulsate rhythmically or can appear roughly drawn. The animator has control over the painting style; object interiors and object outlines are treated independently, so the animator can choose a line-drawing as output, a painting, or a traditional flat-shaded cartoon with solid outlines. Different styles can be applied to distinct objects; specified objects need not be painted at all but left in a photorealistic state. Rotoscoping and matting effects are also possible—in fact, rotoscoping, matting, and painting are unified under our rendering framework.

Our work initially focused upon the problem of rendering video in cartoon-like styles; however, we are now capable of synthesizing a wider gamut of artistic styles, including sketchy, oil and watercolor paint, and cartoon flat and gradient shaded animations within a single framework (see Figs. 6, 8, and 9). The framework we present here also complements work recently published by the authors in [1], [2] which addresses the associated problem of emphasizing motion within video (for example, using cartoon-style streak-lines or squash and stretch deformation). In Section 5, we briefly describe how this motion emphasis work may be integrated with our Stroke Surfaces framework to produce a comprehensive "Video Paintbox" capable of rendering cartoon-styled animations from video.

Throughout this work, our rationale has been to design a framework that offers animators a high degree of expression over the character of animations; both by providing control over visual elements, but, importantly, also by providing a degree of automation that frees them from laborious work (such as tracing around objects every few frames). The enabling factor for such automation is the development and application of Computer Vision methods to NPR and, as such, our work falls within the increasingly active convergence area between computer graphics and computer vision.

1.1 Overview of the Rendering Framework

Internally, our framework is comprised of three principal components: An *intermediate representation* (IR) provides an abstract description of the input video clip. This is parsed from the video by a *computer vision* front end which is responsible for video analysis. This representation, together with any rendering parameters, forms the input to the *computer graphics* back end, which is responsible for all decisions relating to synthesis, subject to animator control.

We have designed a data structure specifically with coherent and flexible rendering in mind. To form our data structure, we begin by independently segmenting frames into connected homogeneous regions using standard Computer Vision techniques [3] and use heuristics to create semantic associations between regions in adjacent frames. Regions are thus connected over time to produce a collection of conceptually high-level spatiotemporal "video objects." These objects carve subvolumes through the video volume delimited by continuous isosurface patches, which we term "Stroke Surfaces." The video is encoded by a set of such boundaries and a counterpart database containing various properties of the enclosed video objects. The

- *J.P. Collomosse and P.M. Hall are with the Department of Computer Science, University of Bath, Claverton Down, Bath, BA2 7AY, UK. E-mail: {jpc, pmh}@cs.bath.ac.uk.*
- *D. Rowntree is with Nanomation Ltd., 6 Windmill St., London, W1T 2JB, UK. E-mail: david@nanomation.co.uk.*

Manuscript received 5 July 2004; revised 24 Nov. 2004; accepted 7 Jan. 2005; published online 10 May 2005.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0066-0704.

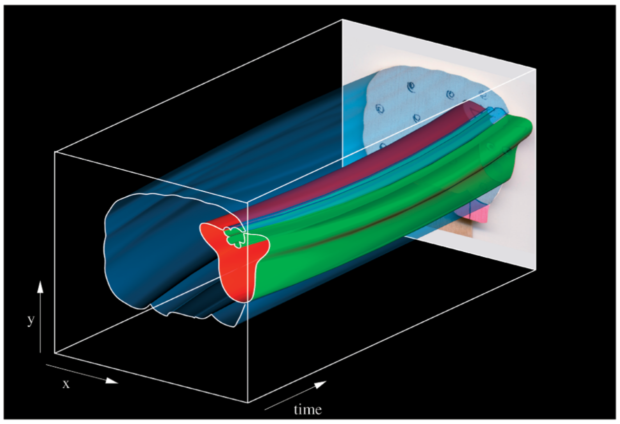

Fig. 1. A visualization of the Stroke Surfaces generated from the *SHEEP* sequence, which, when intersected by the a time plane, generate the region boundaries of a temporally coherent segmentation. The boundary of single video object, for example, the sheep's body, may be described by multiple Stroke Surfaces.

surfaces and database, respectively, form the two halves of the IR, which is passed to the back end for rendering. To render a frame at time t the back end intersects the Stroke Surfaces with the plane $z = t$ to generate a series of splines corresponding to region boundaries in that frame. By manipulating the IR (for example, temporally smoothing the Stroke Surfaces), the back end is able to create temporally coherent animations in a range of artistic styles, under the high-level direction of the animator.

We give detailed descriptions of the front end, IR, and the back end in Sections 2, 3, and 4, respectively.

1.2 Related Work

Our research is aligned with image-based NPR, the branch of NPR which aims to synthesize artwork from real-world images. Research in this area is strong and diverse, encompassing artistic styles such as pen-and-ink, water color, and impasto oil paint which have been emulated [4]. Automatic NPR methods are of the greatest relevance to us since automation is a prerequisite to efficiently painting the numerous frames in any animation (though early rotoscoped animations such as Snow White [Disney, 1937] achieved this through exhaustive manual effort). It is unfortunate that contemporary automatic methods (for example, the painterly methods of [5], [6]) do not generalize easily to video. Some labor can be saved using modern in-betweening methods; rotoscopers working on the film "Waking Life" [Fox Searchlight, 2001] drew around objects every 10 or 20 frames to create cartoon-like sequences. A recent approach [7] tracks, rather than interpolates, key-framed contours over the course of a video. Computer-aided systems have also been developed for producing line-art cartoons [8].

The majority of video-based NPR techniques address the problem of automatically producing painterly animations from video. Litwinowicz [9] was the first to address the problem in a fully automatic manner. Brush strokes painted upon the first frame are translated from frame to frame in accordance with optical flow motion vectors estimated between frames. Translating strokes in this manner causes them to bunch in some places and spread in others so that the density distribution of paint strokes over time is automatically controlled. A similar painterly technique driven by optical flow was proposed by Kovacs and Sziranyi [10]. Hertzmann and Perlin [11] use differences between consecutive frames of video, repainting only those areas which have changed above a global threshold. While these methods can produce impressive painterly video, they do not fully control flicker. Errors present in the estimated motion field quickly accumulate and propagate to subsequent frames, resulting in increasing temporal incoherence which then requires exhaustive manual correction [12]. Others have applied snakes [13] to assist drawing on a per-frame basis [14], with an aim to semi-automatically producing cartoons from video. Unfortunately this approach is not robust to occlusion and instabilities inherent in the per-frame snake relaxation process can cause animations to scintillate. If one were processing video for interaction or real-time animation, then a frame-by-frame approach would be justified (an example is Hertzmann's "Living Painting" [11]). However, our motivation is to process video offline for postproduction and, as such, a global analysis over all frames seems a more promising avenue for producing our temporally coherent animations.

Our approach treats the video sequence as a spatiotemporal array of color—a voxel volume formed by stacking sequential frames over time. This representation is popular in the Computer Vision community (see, for example, [15], [16]) and has been used by the Graphics community: Klein et al. [17] use it to support their work in *video cubism*, which allows users to interactively create NPR effects, but which does not support video painting. Our NPR framework makes use of the spatiotemporal representation of video not only to control temporal coherence, but also to synthesize a wide gamut of artistic effects that would not be possible otherwise. In this regard, our work has a common philosophy with recent work by Wang et al. [18]. Like us, Wang et al. identify spatiotemporal subvolumes that correspond to semantic regions and also fit spatiotemporal surfaces through the video volume. Yet, the two pieces of work contrast in several important aspects:

1. We automatically process the video volume into subvolumes separated by their interfacing surfaces and construct a graph structure to specify the connectivity between these subvolumes; the user is allowed to modify this connectivity as a postprocess by single mouse clicks in distinct frames. Interaction occurs iteratively as a means of refining the quality of animated output. This graph representation is robust to the occlusions and clutter that arise in real video. Wang et al. place their interaction prior to video processing—users draw around specific regions in key frames less than a second apart, which is more labor intensive and processes only the video foreground.
2. We segment the video into regions on a frame-by-frame basis and then associate regions across time (a so called $2D + t$ approach), whereas Wang et al. segment the volume directly using a extension of the mean-shift [3] algorithm to $3D$. We favor $2D + t$ because it helps overcome the problem of volume oversegmentation. Consider a small, fast moving object; imaged regions of that object may not overlap in adjacent frames. In this situation, a $3D$ voxel-based

segmentation would oversegment the object; our $2D + t$ associative approach is able to link the two regions together, thus treating the object as a single video volume.

3. We use a piecewise bicubic representation for our surfaces, whereas Wang et al. use polyhedral meshes fitted by marching cubes. The advantage of parametric surfaces is that they allow specific control over continuity conditions and, thus, over temporal coherence in the video. Also, in contrast to [18], our surfaces represent the interfaces between subvolumes, rather than the boundary of each distinct subvolume. We avoided the latter since it led to duplication of surface information, causing boundaries to no longer appear coincident following surface manipulations (for example, when performing smoothing operations to enhance temporal coherence, or performing warps for animation effects—Section 4.1).
4. We fit a set of basis vectors to semantic regions, which move coherently over time; this is not a feature of [18]. These basis sets serve to widen the gamut of artistic styles from flat shading to include stroke-based painterly rendering approaches such as those used by by Litwinowicz [9]. Furthermore, they allow us to unify matting, rotoscoping, and painting in a common framework.

2 Front End: Segmenting the Video Volume

The front end is responsible for the smooth segmentation of the video volume into subvolumes in a voxel representation which describe the trajectories of features. There are three stages to this process. We begin by independently segmenting frames into connected homogeneous regions. The second stage associates segmented regions in each frame with regions in adjacent frames to generate video objects. The third stage removes spurious associations and temporally smooths the video objects.

2.1 Frame Segmentation

In common with earlier segmentation-based NPR work ([6]), we segment on the basis of homogeneous color. This was chosen for initial simplicity only and does not constitute a fundamental limitation of our approach. Even so, the choice of segmentation algorithm influences the success of the later region-matching step because segmentations of adjacent frames must yield similar class maps to facilitate association.

Robustness—defined as the ability to produce near-identical results given distinct but near identical images—is an important property of the segmentation algorithm we use. Although most published 2D segmentation algorithms are accompanied by an evaluation of their performance versus a ground truth segmentation, to the best of our knowledge, a comparative study of robustness, as we have defined it, is not present in the literature. Consequently, we empirically investigated the robustness of contemporary 2D segmentation algorithms. Briefly, each segmentation algorithm was applied to a series of video frames to produce a series of segmented class maps. Robustness of the algorithm was a measured as a function of error between distance transforms of consecutive class maps. Full experimental details, including a more formal definition of robustness, can be found elsewhere [19]. Here, we state the conclusion of that experiment: that EDISON [20] proved the most robust algorithm under our definition.

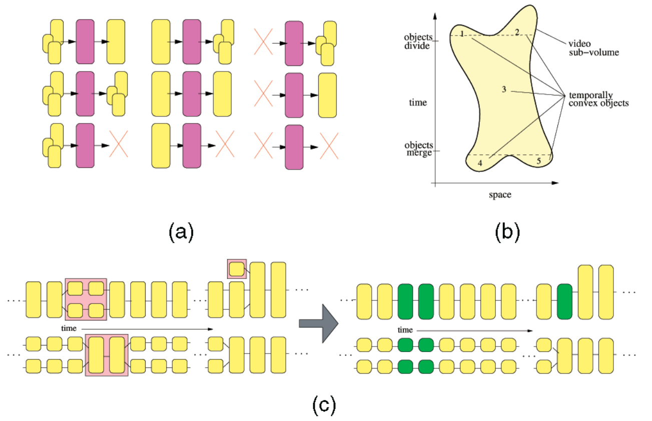

Fig. 2. (a) Example of a single video-subvolume split into five temporally convex objects. (b) Two examples of an object association graph before and after the filtering step of Section 2.3. Sporadic associations (red) are removed and object boundaries interpolated (green) from neighbors.

2.2 Region Matching

Each frame is now partitioned into a set of regions which must be associated over time. The problem of creating correspondence between sets of regions, each within multiple frames, is combinatorial in nature and an optimal solution cannot be found through exhaustive search for any practical video. We propose a heuristic solution which we have found to perform well (that is, it results in a locally optimal solution exhibiting acceptable temporal coherence) and has quadratic complexity in the number of regions per frame.

Consider a single region $r \in R_t$, where R_t denotes the set of segmented regions in frame t. We wish to find the set of regions in adjacent frames with which r is associated. We compute this by searching sets R_{t-1} and R_{t+1} independently —examining potential mappings from r to R_{t-1} and then from r to R_{t+1}. Thus, r may be associated with zero, one, or more regions in adjacent frames. The suitability of match between two regions in adjacent frames $r \in R_t$ and $\rho \in R_{t\pm1}$ is evaluated by an objective function $E(r, \rho)$. We describe this function momentarily (Section 2.2.1), but first complete description of the association algorithm.

For the purposes of illustration, let us consider the creation of associations between r and regions in the set R_{t+1}. A matching set is initialized to empty and the area (in pixels) of r is computed as an "area-count." A potential set of associating regions in R_{t+1} is identified whose centroids fall within a distance Δ of the centroid of r. A score $E(.)$ for potential region each is computed. The regions are sorted into a list in descending order of their score. Next, a cumulative sum of their area counts is computed, working from the start of the list and storing the cumulative sum with each region. The area-count of r is subtracted from each term. The matching set extends down this list until either the score or area measure falls below a threshold. For a given R_i, we form matching sets with regions in adjacent frames both in the future and in the past. The resulting subvolumes are broken into, possibly many, temporally convex *video objects*. A property of the temporally convex representation is that many separate video objects can merge to produce one object and a single video object division can split into many objects. We therefore generate a graph structure, with video objects as nodes, specifying how those objects split and merge over time.

2.2.1 Heuristics for Association

The quality of match between two regions R_i and R_j is measured by the following heuristic function:

$$E(R_i, R_j) = \begin{cases} 0 & \text{if } \delta(R_i, R_j; \Delta) > 1 \\ e(R_i, R_j) & \text{otherwise,} \end{cases} \tag{1}$$

$$\begin{aligned} e(R_i, R_j) = w_1\sigma(R_i, R_j) + w_2\alpha(R_i, r_j) - \\ w_3\delta(R_i, R_j; \Delta) - w_4\gamma(R_i, r_j). \end{aligned} \tag{2}$$

The function $\delta(.)$ is the spatial distance between the region centroids as a fraction of some threshold distance Δ. The purpose of this threshold is to prevent regions that are far apart from being considered as potentially matching; $e(.)$ is not computed unless the regions are sufficiently close. We have found $\Delta = 30$ pixels to be a useful threshold. Constants $w_{[1..4]}$ weight the influence of each of four heuristic functions; the functions are all bounded in $[0, 1]$. These parameters may be modified by the user to tune the association process, though, typically, less than an order of magnitude of variation is required. We have found $w_1 = 0.8, w_2 = 0.6, w_3 = 0.6, w_4 = 0.4$ to be suitable values for the processing of all videos presented in this paper. The function $\gamma(.)$ is the Euclidean distance between the mean colors of the two regions in *CIELAB* space (normalized by division by $\sqrt{3}$). $\alpha(.)$ is a ratio of the two regions' areas in pixels. $\sigma(.)$ is a shape similarity measure, computed between the two regions. Regions are first normalized to be of equal area, $\sigma(.)$ is then computed by taking Fourier descriptors of the angular description function [21] of each region's boundary. Shape similarity is inversely proportional to Euclidean distance between the magnitude vectors of the Fourier descriptors for both regions (disregarding phase). The shape descriptors are therefore invariant to shape translation, rotation, and uniform scaling.

2.3 Filtering and Smoothing

Our video description thus far is comprised of a graph in which spatiotemporal objects are nodes and edges connect matched objects. This graph is noisy in the sense that some objects are short-lived, so we filter out objects that exist for less than six frames (about $\frac{1}{4}$ second). The "holes" left by cutting such objects are filled by extrapolating from immediate neighbors. A serendipitous effect of this process is that poorly segmented areas of the video tend to merge into one large coherent object.

We now describe a temporal smoothing process performed by fitting continuous surfaces to voxel objects. Note that these surfaces are discarded following the smoothing process, and are distinct from the Stroke Surfaces later fitted to form the IR.

The boundary of any object is described by a spatiotemporal surface. Disregarding "end caps"—surfaces for which time is constant—we fit a piecewise-smooth surface to object boundaries using bicubic Catmull-Rom patches, which are interpolating splines. Fitting is performed via a generalization of active contours to surfaces (after [13], [22]):

$$E = \int_0^1 \int_0^1 (E_{int}[\underline{Q}(s,t)] + E_{ext}[\underline{Q}(s,t)])\mathrm{d}s\mathrm{d}t, \tag{3}$$

the internal energy is

$$\begin{aligned} E_{int} = \alpha\left|\frac{\partial\underline{Q}(s,t)}{\partial s}\right|^2 + \beta\left|\frac{\partial\underline{Q}(s,t)}{\partial t}\right|^2 + \\ \gamma\left|\frac{\partial^2\underline{Q}(s,t)}{\partial s^2}\right|^2 + \delta\left|\frac{\partial^2\underline{Q}(s,t)}{\partial t^2}\right|^2 \end{aligned} \tag{4}$$

and the external energy is

$$E_{ext} = \eta f(\underline{Q}(s,t)). \tag{5}$$

Function $f(.)$ is the Euclidean distance of the surface point $\underline{Q}(s,t)$ to the closest voxel of the object, hence constant η (which we preset as unity) controls the influence of the data in the fit. We preset control over spatiotemporal gradients $\alpha = 0.5$, $\beta = 0.25$, and spatial curvature $\gamma = 0.5$. Constant ζ dictates the penalties associated with high curvatures in the temporal dimension, which correlate strongly with temporal incoherence in the segmentation. We have chosen to make the value of this temporal constraint available to the user, thus allowing variation in the degree of temporal smoothing.

Surfaces are fitted within the volume using an adaptation of Williams' algorithm to locate surface points [23], which, in the final iterations of the fit, relaxes penalties due to high magnitudes in the second derivative. We inhibit this relaxation in the temporal dimension to improve temporal coherence. Once the continuous, bounding surfaces have been smoothly fitted, the volume is requantized to return to a voxel representation.

3 Forming the Intermediate Representation

The results of the video segmentation process (Section 2) are a set of segmented, smooth, video objects in a voxel representation, and an object association graph describing how video objects are connected over time. In our IR, spatiotemporal objects are represented in terms of *stroke surfaces* that mark the boundary between exactly two neighboring objects. Each connected component in the boundary of an object pair has its own Stroke Surface. Stroke surfaces may contain internal "holes" if the boundary between neighboring object has holes. Each Stroke Surface has a *winged edge structure* [24], meaning it contains pointers referencing a counterpart database, which holds information about the objects which it separates. Stroke Surfaces are preferred over storing each object in terms of its own bounding surface because boundary information is not duplicated and the IR is more compact and more manipulable.

But, there are disadvantages, too: It is harder to vary the topology of the scene, for example, the adjacency of the objects, while maintaining the containment of objects in this representation (this motivated the temporal smoothing step in the previous section). Stroke Surfaces are not fitted to objects thate have no spatial adjacency but are only temporally adjacent—this information is already encoded in the association graph. Now, given adjacent objects A and

B, we first identify each connected component in their boundary and fit a Stroke Surface to each component independently, using a two-stage algorithm. First, a two-dimensional surface is fitted over the boundary using an active-surface approach of the kind already described in Section 2.3. Second, any holes in the connected component are accounted for by fitting a surface patch over each hole. The Stroke Surface is then comprised of those points in the bounding surface that are not in any of the "hole" surfaces.

The boundary between objects is defined as follows: Consider two abutting objects, A and B, say. The distance transform for all pixels in A with respect to object B is computed, the distance transform for all pixels in B with respect to object A. Thus, a scalar field is generated that spans all voxels in both A and B; the minimum of the scalar fields is the boundary used in the fitting process.

3.1 The Counterpart Database

We maintain a counterpart database that stores information about objects. The database is indexed by the pointers held in the Stroke Surfaces' winged edge structure. Its purpose is to store information that is useful to the NPR process in the back end, for example, a record of the object's color at each temporal instant—other specifics are best left to Section 4. Some data, though, is common to all NPR applications; this generic data is comprised of the following fields:

1. **Born:** The frame number in which the object starts.
2. **Died:** The frame number in which the object ends.
3. **Parent object list:** A list of objects which have either split or merged at time B to form the current object. Objects can be assigned an empty list of parents: No object in frame one, for example, has a parent; neither do objects that are revealed by removal of a previously occluding object.
4. **Child object list:** A list of objects which the current object will become. If this list is empty, the object simply disappears (possibly due to occlusion or reaching the end of the video).

These attributes encode the graph structure that links objects (see Section 2.2). The Stroke Surfaces and counterpart database together comprise the basic intermediate representation (IR) that is passed to the back-end for rendering.

3.2 User Correction

Processing video into the IR is strongly dominated by automation, but Segmentation is imperfect and user correction is necessary for some sequences. We bias the parameters to the EDISON algorithm (used in the frame segmentation step) slightly toward oversegmentation because oversegmentation is much more easily resolved via merging objects than undersegmentation (which introduces the complicated problem of specifying a spatiotemporal surface along which to split video objects). In practice, therefore, the volume may be oversegmented. We therefore provide an interactive facility for the user to correct the system by linking objects as required. This operation takes place directly after the region association and filtering process. These links may take two forms:

1. *Semantic link:* The animator links two objects by creating edges in the object association graph. Objects remain as two distinct volumes in our representation, but the graph is modified so that any smoothing of graphical attributes (such as region color) occurs over all linked objects (see Fig. 3b). This type of link is used to encode additional, semantic knowledge of the scene (for example, the continuity of the yellow bear's fur in Fig. 3b).
2. *Physical link:* The animator physically links two adjacent video objects by merging their voxel volumes. The objects to be merged are deleted from the representation and replaced by a single object which is the union of the linked objects (see Fig. 3a). This type of link is often used to correct over-segmentation due to artifacts such as shadow or noise and is preferable in this respect to "semantic" linking since merged volumes will undergo coarse smoothing as a single object.

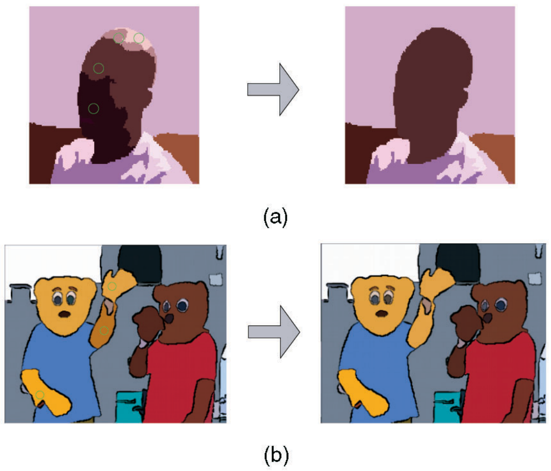
Fig. 3. Users may create manual links between objects to tune the segmentation or composition. (a) The user creates a physical link between oversegmented objects, a single object replaces four. (b) The user creates a semantic link between two objects. The objects remain distinct, but edges are created in the object association graph; during rendering, attributes are blended between regions to maintain coherence.

4 Back End: Rendering the IR

We now describe the back end of the framework, responsible for rendering the IR. Given our spatiotemporal video representation, any frame is on a plane of constant time that intersects the volume. The Stroke Surfaces can therefore be rendered as lines and the interiors of objects as areas. This immediately gives us the opportunity to render video into novel nonphotorealistic styles that were not possible hitherto. For example, we can create line renderings or paint regions in a flat color (as we alluded to earlier, attributes such as color can be added to the database in an style-specific field). Many other nonphotorealistic effects are possible, as we now describe.

4.1 Surface Manipulation

The representation of video as a set of spatiotemporal Stroke Surfaces simplifies manipulation of the image sequence in both spatial and temporal domains and enables us to synthesize novel temporal effects which would otherwise be difficult to produce on a per frame basis. To

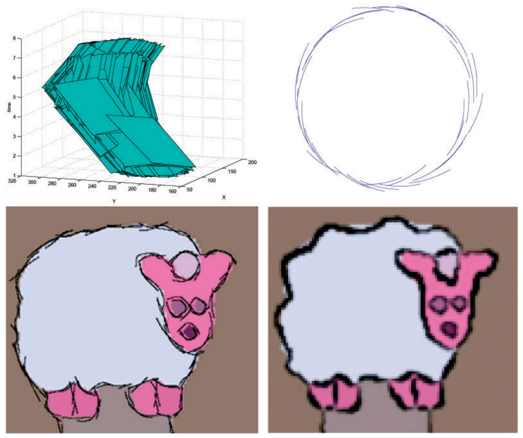
Fig. 4. Examples of controlled incoherence that may be inserted into an animation through Stroke Surface manipulation. Top left: Visualization of the XYT volume in which the surfaces describing an object have been shattered into small pieces and "jittered" by random affine transformations. This produces a broken boundary that yields a sketched effect when rendered (top right). Bottom: A coherent sketchy effect creates by shattering and jttering surfaces in the *SHEEP* sequence (left) and controlled wobbles created by perturbing the same surfaces (right).

this end, we have devised a method for manipulating Stroke Surfaces to produce small scale deformations.

Consider a single Stroke Surface $\underline{q}(s,t)$ embedded in the video volume, where (s,t) form an arc-length parameterization of that surface. We subsample the fields $s \in [0,1], t \in [0,1]$ at regular intervals to obtain a grid of well-distributed points on manifold $\underline{q}(s,t)$. By creating a simple 3D triangular mesh using these regular samples as vertices, we are able to create a piecewise planar approximation to $\underline{q}(s,t)$, which we write as $\underline{p}(s,t)$. The difference between these two vector fields $\underline{e}(s,t) = \underline{q}(s,t) - \underline{p}(s,t)$ forms the basis for deforming the Stroke Surface. By holding $\underline{p}(.)$ constant and manipulating $\underline{e}(.)$ in frequency space, we are able to produce a range of novel temporal effects. For example, since adjacent objects are represented in terms of their interfacing surfaces, residual temporal incoherences in those boundaries may be dampened by smoothing the surfaces. Thus, applying a low-pass (Gaussian) filter in the temporal dimension allows us to further enhance temporal coherence. There is no danger of introducing "holes" into the video volume as with the coarse smoothing step (Section 2.3)—if volume is lost from one video object, it is gained by surrounding video objects. Similarly, by introducing novel frequency components, we can produce coherent "wobbles" in the boundaries of animated objects. Fig. 4 contains stills from such animations, reminiscent of commercial cartoons, for example, "Roobarb and Custard" [Grange Calveley, 1972].

4.2 Rendering Region Interiors

As described earlier, a straightforward strategy for rendering a region interior is to uniformly fill it with a single, mean color computed over that region's footprint in the original video. The cartoon-like "flat shading" effect that results goes some way to satisfying the second (shading) subgoal of our original motivation—the automated generation of cartoons from video. However, as video objects divide and merge over the course of the clip, the mean color of their imaged regions can change significantly from frame to frame (perhaps due to shadows). This can cause unnatural, rapid color changes and flickering in the video.

Thus, the ability to obtain temporally smooth segmentation (Stroke Surface) boundaries in an animation is insufficient on its own. One must also apply shading attributes to the bounded regions in a temporally smooth manner. The flickering of region color we demonstrate is symptomatic of the more general problem of assigning the graphical attributes stored in the IR database to regions in a coherent way. We draw upon our spatiotemporal representation to mitigate against this incoherence.

Recall that objects are associated via a graph structure; pointers to each video object's child and parent objects are stored in the database component of the IR. We analyze this graph to obtain a binary voxel map describing the union of all video objects within the subgraph containing the video object corresponding to the region being rendered. By averaging graphical database attributes, such as color, over the volume, we can create a smooth transition of those attributes over time (even if objects appear disjoint in the current frame but connect at some other instant in the past or future). Such coherence could not be obtained using the per frame sequential analysis performed by current video-driven NPR methods [9], [10], [11].

The highly abstracted nature of a flat shaded video can be unappealing for certain applications; artists often make use of shadows and shading cues to add a sense of lighting and depth to a scene. Although comprehensive detection and handling of shadows remains an area for future work, we have attempted to reintroduce shading cues in two ways.

First, we fit a linear shading gradient to each object on a per frame basis. The gradient at time t over an object may be described as a triple $\underline{G_t} = [g_0, g_1, \theta]$, where g_0 and g_1 are the start and end shading intensities, respectively, and θ specifies the direction of shading over the region (as an angle). An optimal $\underline{G_t}$ is sought by minimizing the error between the luminance channel of the region texture and a synthesized intensity gradient generated by a putative $\underline{G_t}$. These gradient triples are computed by the front end and stored as a field in the IR database. When rendering, we can augment our flat shaded regions by rendering the gradient triple, smoothing the parameters in a similar manner to color to promote temporal coherence (Fig. 8, top middle).

Second, we compute the interframe (planar projective) homographies for each video object's texture as it moves from frame to frame. The sequences of homographies are stored in the database and smoothed over time. In this way, we attach a rigid reference frame to the object that transforms smoothly over time. We transform region texture at each time instant to fall within a common reference frame. By detecting and thresholding edges within these texture samples, we are able to form connected, binary spatiotemporal components for strong edges within the interior of the object. Stroke surfaces are fitted to these components, as in Section 3, and thus represent interior edge cues in our IR; in this case, both pointers in the winged edge structure reference the same object. This property also proves useful to differentiate rendering style between interior and exterior edges.

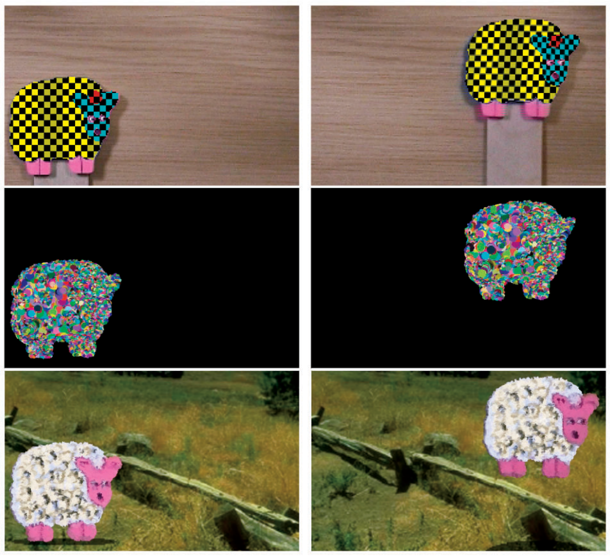

Fig. 5. Coherent painterly rendering: Painting frames are attached to objects (top) and strokes move with them (middle). The painted sheep has been matted onto a static photography (bottom).

We are also able to apply morphological operators to interior regions, prior to rendering. Fig. 8 (bottom left) demonstrates a water color effect, combined with a sketchy outline, in which we have applied an erosion operator to the region prior to rendering; this gives the aesthetically pleasing impression of brush strokes stopping "just short of the edges."

4.3 Reference Frames and Rotoscoping

Modern NPR techniques (for example, stroke-based renderers [2], [5], [9] or adaptive textures [25]) may be considered a form of modern day rotoscoping as both share a common objective: to produce stylized animations from video in which motion is coherent with the motion of the underlying source image sequence.

Recall the local motion estimates for video objects stored in the IR. These estimates model interframe motion as a homography and were applied to recover internal edge cues. However, the same motion estimates may be used to to implement automated rotoscoping in our framework. Animators draw a (potentially animated) design upon a single key-frame of the original footage. The interframe homography stored in the IR database is then used to automatically transform the design from frame to frame—the design remains static within the reference frame to which it is attached; it appears to be rigidly attached to the video object (see Fig. 5).

Rather than have the animator draw a design on the rigid reference frame, we may use that frame as the basis on a canvas on which to deposit paint strokes. These strokes may adapt their color according to the video texture over which they are painted. The visual attributes of these strokes, such as color, scale, and orientation, can also be smoothed over time. Thus, we can produce flicker-free painterly effects in video by considering painting and rotoscoping as essentially the same task. We have found that affine transforms are aesthetically better suited for carrying painting strokes from frame to frame and we simply compute the closest affine transform (in a least squares sense) to each homography. In doing so, we allow strokes to translate, rotate, and scale with the region on which they are painted; the latter prevents holes from appearing between strokes as objects approach the camera.

As a special case of rotoscoping, we may set the interframe homography estimates for a region to imply no region motion. By supplying these regions with video as a texture, rather than hand-drawn animations, we are able to replace shaded video objects with alternatively sourced video footage. This facilitates video matting within our framework, as we demonstrate in Fig. 5 (bottom) by substituting a novel background into the *SHEEP* sequence. We can also use the source video footage itself as a texture and thus reintroduce photorealistic objects from the original video back into the nonphotorealistic animation (producing the "mixed media" effects in Fig. 8, top right).

4.4 Rendering Region Outlines

Having concluded our explanation of interior region rendering, we now describe the edge rendering process of the back end. Recall the surface intersection operation by which we determine the Stroke Surfaces to be rendered a particular frame. The splines which result from this intersection form trajectories along which we paint long, flowing strokes, which are stylized according to a user-selected procedural NPR brush model. This produces attractive strokes which move with temporal coherence through the video sequence; a consequence of the smooth spatiotemporal nature of the Stroke Surfaces. We have used an implementation of Strassman's hairy brush model [26] to render our splines, producing thick painterly strokes to depict the exterior (holding) lines of objects (see Fig. 8, bottom right).

Because temporal incoherence has been brought under control, the corollary is that we may reintroduce incoherences in a controlled way. We have discussed (Section 4.1) how we may introduce undulations in the surfaces, which causes the lines to "wobble." However, we can produce other effects specific to line rendering. For example, Stroke Surfaces can be "shattered" into small shards, each of which exhibits temporal coherence over a short time. Shards are subjected to small, random affine transformations and rendered using fine brush strokes to produce a sketchy effect (Fig. 4).

5 Results and Comparison

Fig. 8 contains examples of stills taken from animations rendered by our rendering framework. In Fig. 9, we demonstrate the results of integrating our proposed framework with our earlier cartoon motion emphasis framework [1], [2]. This earlier framework shares a similar architecture to that described in this paper. Object descriptions were extracted from the source video by a Computer Vision front end and passed to a graphics back end which applied various animation motion cues to the original footage, squash and stretch object deformation and streak line placement being but two examples. In our integrated Video Paintbox, the front end of the motion emphasis framework operates upon the original footage as before. However, the footage passed to the back end is substituted for the nonphotorealistic animations

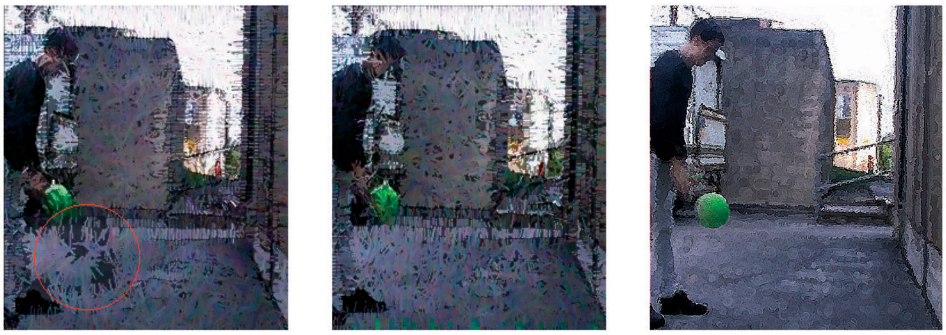

Fig. 6. Identical frames four seconds into the *BOUNCE* sequence rendered in a oil-painted style with our technique (right) and OF (left, middle). The errors that have built up in the cumulative optical flow estimate at this stage cause large-scale distortion and flickering in the image. We have omitted the stroke density regulation stage (OF-) in the left-hand image; this improves temporal coherence. However, large holes are created in the canvas (red circle) and, elsewhere, tight bunching of strokes causes the scene to tend back toward photorealism.

stylized by our Stroke Surface framework. This enables the synthesis of cartoon-styled animations which not only appear "cartoon shaded," but also exhibit many of the motion cues used by traditional animators.

The majority of artistic styles that may be synthesized by the complete Video Paintbox have no counterpart in the published NPR literature; most existing techniques address only the single, specific problem of producing temporally coherent painterly animations. This improved diversity of style is one of our principal contributions; however, we also wish to demonstrate the contribution made due to the improvements in temporal coherence. To this end, we compared our animation system with Litwinowicz's video painting algorithm based on optical flow [9]. This approach tends to produce animations exhibiting poor temporal coherence; a manifestation of the motion estimation errors which accumulate due to the per frame sequential nature of the algorithm. This is especially noticeable in "flat" regions since optical flow tends to show movement only at edges and textures. In addition, the translations performed over time causes strokes to "bunch" together, leaving "holes" in the canvas which must be filled with new strokes. The addition of strokes is driven by a stochastic process in [9] and thus also contributes to temporal incoherence. We measured the temporal coherence of painted video objectively in two ways: stroke flicker and motion coherence.

Considering flicker, the location and visual attributes of strokes should not vary rapidly, so, if $\underline{s}_i$ is a vector of stroke parameters (location, color, etc.) in frame i, then $f = ||\underline{s}_t - \underline{s}_{t-1}| - |\underline{s}_{t+1} - \underline{s}_t||$ is a measure of flicker; the higher the score, the greater the flicker. We averaged f over many frames and many strokes and used it to compare our method with two versions of Litwinowicz's algorithm, one in which the "bunching/holes" problem was not resolved (OF-) and one in which it was (OF). Numeric results are shown in Fig. 10 (right) for three test videos (see accompanying material), in which we have normalized scores so that OF is unity. We offer about an order of magnitude improvement over [9].

	OF	OF-	Our method
SPHERES	0.62	0.45	0.05
BOUNCE	0.79	0.65	0.11
SHEEP	0.78	0.61	0.13

Fig. 7. Table summarizing the "level of visual flicker" of paint strokes present in an animation produced by each of three algorithms (horizontal) over each of three source video sequences (vertical). Flicker level ranges from zero (no flicker) to unity (sever flicker).

Considering motion coherence, stroke movement should cohere with the motion of objects to which they are attached. We rotated, scaled, and translated images with and without texture, thus providing a ground-truth vector field over a variety of conditions. We then computed an optical flow field using a standard method [27] and a vector field using our method. In all cases, our method reproduced the ground truth vector field very closely, but the optical flow method produced very different results. This was especially clear in regions of flat texture; the local motion estimates of optical flow measured little motion in these regions. By contrast, our motion estimate was computed over the entire segmented object, thus distributing error globally over all pixels. Further details of all these experiments available in [19].

6 Conclusions and Discussion

We have presented a novel NPR framework for synthesizing nonphotorealistic animations from video sequences. The spatiotemporal approach adopted by our framework enables us to smoothly vary attributes, such as region or stroke color over time, and to create improved motion estimates of objects in the video. By manipulating video as distinct regions tracked over time, rather than individual voxels, we are able to produce robust motion estimates for objects and synthesize both region-based (e.g., flat-shaded cartoon) and stroke-based (e.g., traditional painterly) NPR styles. For the latter, brush stroke motion is guaranteed to be consistent over entire regions—contradictory visual cues do not arise where stroke motion differs within a given object. We have also provided a single framework for matting and rotoscoping and allow many novel effects that are unique to our NPR application. We have successfully integrated our framework with earlier motion emphasis work [1], [2], enabling complete cartoon style animations to be generated from video with a high degree of automation. Further work might investigate application of other stroke-based NPR algorithms to our rotoscoping framework and experiment with alternative brush models or alpha blending techniques to stylize boundaries.

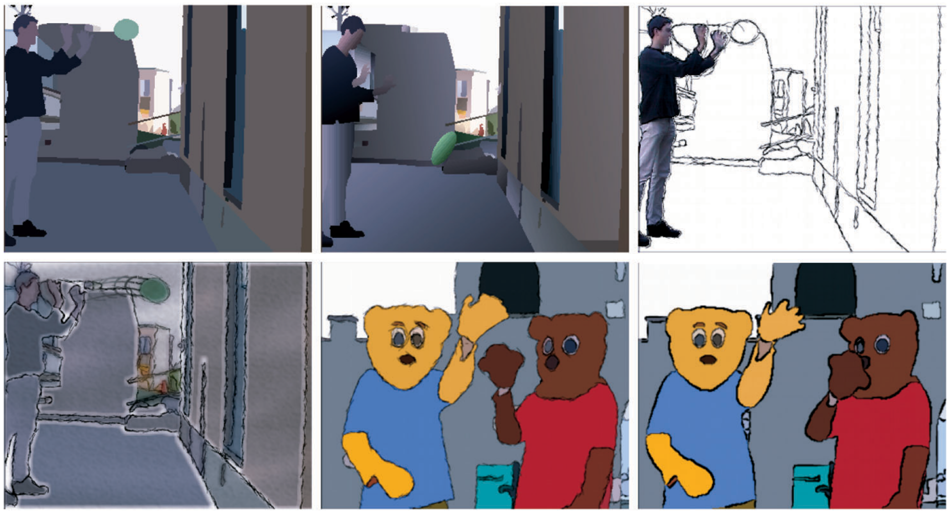

Fig. 8. Some of the available artistic effects. Top, left to right: Cartoon flat and gradient shaded animations from the *BOUNCE* sequence. Mixed reality effect where original footage has been selectively matted in to a sketchy line animation. Bottom, left to right: Water color wash effect and cartoon flat shaded bears with sketchy and thick stroke outlines. See Fig. 9 (left) for stills of the source footage.

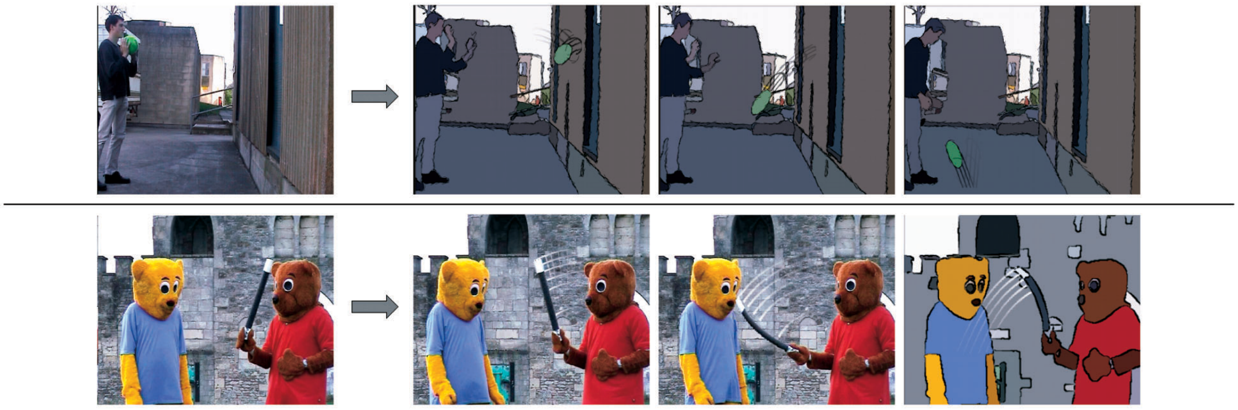

Fig. 9. The Stroke Surface framework may be integrated with our motion emphasis work [1], [2] to produce complete cartoon-styled animations. Videos of these two sequences have been included in the supplementary material.

We did not set out to produce a fully automated system—not only do we desire interaction for creative reasons (setting high-level parameters, etc.), but also for the correction of the Vision algorithms in the front end. The creative nature of animation demands the user be kept "in the loop," a point well-illustrated in Fig. 9, which demonstrates not only the strengths of our automated process (for example, flicker-free shading and motion emphasis), but also the limitations. In Fig. 9 (top), it is unlikely that an animator would create such a "busy" background; commonly, such superfluous detail is abstracted away. This has been accomplished in Fig. 9 (bottom) by interactively merging video objects to hint at the presence of brickwork in the background. This is a manifestation of the "general segmentation problem," which precludes the possibility of segmenting any given video into semantically meaningfully parts. However, we have kept the burden of correction low (Section 3.2). Users need only click on video objects once, for example, to merge two oversegmented feature subvolumes, and those changes are propagated throughout the spatiotemporal video volume. In practical terms, user correction takes only a couple of minutes of user time in contrast to the hundreds of man hours required to correct the optical flow fields of contemporary video NPR techniques [12] or repeatedly key a rotoscoping process.

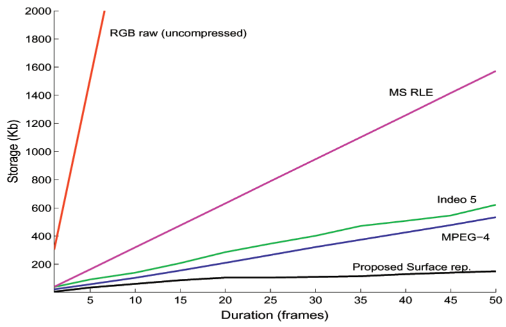

Fig. 10. Demonstrating the comparatively low storage requirements of the surface representation when transmitting cartoons. Our test comparison uses up to 50 frames of a typical gradient-shaded animation produced by our system.

An area for future investigation is the compact nature of our continuous Stroke Surface representation. Fig. 10 summarizes the details of a brief comparative investigation, contrasting the storage requirements of our video description with those of common video compression technologies. Approximately 150KB were required to store 50 frames of a typical video sequence. The compact nature of our description compares favorably with the other video compression algorithms tested when operating upon cartoon-shaded animation, although we note that the spatiotemporal nature of our representation prohibits real-time encoding of video. It is therefore conceivable that our video description could be transmitted—and after, rendered into a number of artistic styles. This creates a novel level of abstraction for video in which a service provider might determine the video content, while the client determines the style in which that content is rendered. The level of abstraction is analogous to that of XML/XSLT documents. Splitting the responsibilities of video content provision and visualization between client and server is a promising direction for developing our video representation.

Perhaps the most limiting assumption in our system is that video must be segmented into homogeneous regions in order to be parsed into the IR (and thus subsequently rendered). Certain classes of video (for example, crowd scenes or running water) do not readily lend themselves to segmentation and thus cause our method difficulty. Typically, such scenes are undersegmented as large feature subvolumes, causing an unappealing loss of detail in the animation. This is not surprising; the segmentation of such scenes is a difficult task even for a human observer. Thus, although we are able to produce large improvements in the temporal coherence of many animations, our method is less general than optical flow-based methods, which are able to operate on all classes of video, albeit with a lower degree of coherence.

A further criticism is that the homography, used to attach reference frames assumes both planar motion and rigid objects. However, many artistic styles (such as cartoon shading and sketchy rendering) do not use this aspect of the system. Ideally, the video representation would be extended to increase the generality of all artistic styles: three-dimensional descriptions of objects or curvilinear (rather than linear) bases for the reference frame are two possible directions for development.

A full description of our Video Paintbox can be found in [19]. See http://www.cs.bath.ac.uk/~vision/cartoon for a selection of rendered clips.

REFERENCES

[1] J.P. Collomosse, D. Rowntree, and P.M. Hall, "Cartoon-Style Rendering of Motion from Video," *Proc. First Int'l Conf. Vision, Video, and Graphics (VVG),* pp. 117-124, July 2003.

[2] J.P. Collomosse, D. Rowntree, and P.M. Hall, "Video Analysis for Cartoon-Like Special Effects," *Proc. 14th British Machine Vision Conf.,* vol. 2, pp. 749-758, Sept. 2003.

[3] D. Comanicu and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 603-619, May 2002.

[4] G. Gooch and A. Gooch, *Non-Photorealistic Rendering.* A.K. Peters, July 2001.

[5] A. Hertzmann, "Painterly Rendering with Curved Strokes of Multiple Sizes," *Proc. ACM SIGGRAPH,* pp. 453-460, 1998.

[6] D. DeCarlo and A. Santella, "Abstracted Painterly Renderings Using Eye-Tracking Data," *Proc. ACM SIGGRAPH,* pp. 769-776, 2002.

[7] A. Agarwala, A. Hertzmann, D. Salesin, and S. Seitz, "Keyframe-Based Tracking for Rotoscoping and Animation," *Proc. ACM SIGGRAPH,* pp. 584-591, 2004.

[8] J. Fekete, E. Bizouarn, T. Galas, and F. Taillefer, "Tictactoon: A Paperless System for Professional 2D Animation," *Proc. ACM SIGGRAPH,* pp. 79-90, 1995.

[9] P. Litwinowicz, "Processing Images and Video for an Impressionist Effect," *Proc. ACM SIGGRAPH,* pp. 407-414, 1997.

[10] L. Kovacs and T. Sziranyi, "Creating Video Animations Combining Stochastic Paintbrush Transformation and Motion Detection," *Proc. 16th Int'l Conf. Pattern Recognition,* vol. II, pp. 1090-1093, 2002.

[11] A. Hertzmann and K. Perlin, "Painterly Rendering for Video and Interaction," *Proc. ACM Non-Photorealistic Animation and Rendering (NPAR),* pp. 7-12, 2000.

[12] S. Green, D. Salesin, S. Schofield, A. Hertzmann, and P. Litwinowicz, "Non-Photorealistic Rendering," *ACM SIGGRAPH, NPR Course Notes,* 1999.

[13] M. Kass, A. Witkin, and D. Terzopoulos, "Active Contour Models," *Int'l J. Computer Vision,* vol. 1, no. 4, pp. 321-331, 1987.

[14] A. Agarwala, "Snaketoonz: A Semi-Automatic Approach to Creating Cel Animation from Video," *Proc. ACM Non-Photorealistic Animation and Rendering Conf. (NPAR),* pp. 139-147, June 2002.

[15] D. Terzopolous, A. Witkin, and M. Kass, "Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion," *Artificial Intelligence,* vol. 36, no. 1, pp. 91-123, 1988.

[16] D. DeMenthon, "Spatio-Temporal Segmentation of Video by Hierarchical Mean Shift Analysis," *Proc. Statistical Methods in Video Processing (SMVP) Workshop at ECCV,* 2002.

[17] A.W. Klein, P.J. Sloan, R.A. Colburn, A. Finkelstein, and M.F. Cohen, "Video Cubism," Technical Report MSR-TR-2001-45, Microsoft Research, May 2001.

[18] J. Wang, Y. Xu, H-Y. Shum, and M. Cohen, "Video Tooning," *Proc. ACM SIGGRAPH,* pp. 574-583, 2004.

[19] J.P. Collomosse, "Higher Level Techniques for the Artistic Rendering of Images and Video," PhD thesis, Univ. of Bath, U.K., May 2004.

[20] C. Christoudias, B. Georgescu, and P. Meer, "Synergism in Low Level Vision," *Proc. 16th Int'l Conf. Pattern Recognition,* vol. 4, pp. 150-155, Aug. 2002.

[21] R.L. Cosgriff, "Identification of Shape," Technical Report 820-11, ASTIA AD 254792, Ohio State Univ., 1960.

[22] G. Hamarneh and T. Gustavsson, "Deformable Spatio-Temporal Shape Models: Extending asm to 2d+time," *J. Image and Vision Computing,* vol. 22, no. 6, pp. 461-470, 2004.

[23] D. Williams and M. Shah, "A Fast Algorithm for Active Contours and Curvature Estimation," *Graphical Models and Image Processing,* vol. 55, no. 1, pp. 14-26, 1992.

[24] B. Baumgart, "A Polyhedral Representation for Computer Vision," *Proc. Nat'l Computer Conf.,* pp. 589-596, 1975.

[25] P. Hall, "Non-Photorealistic Rendering by Q-Mapping," *Computer Graphics Forum,* vol. 1, no. 18, pp. 27-39, 1999.

[26] S. Strassmann, "Hairy Brushes," *Proc. ACM SIGGRAPH,* vol. 20, pp. 225-232, 1986.

[27] T. Gautama and M.M. VanHulle, "A Phase-Based Approach to the Estimation of the Optical Flow Field Using Spatial Filtering," *IEEE Trans. Neural Networks,* vol. 5, no. 13, pp. 1127-1136, 2002.

John P. Collomosse received the PhD degree from the University of Bath in 2004; his thesis addresses the problem of artistic rendering from digital images and video. Now employed as a lecturer (assistant professor), he is still based in Bath and holds related research interests in augmented reality, NPR, and computer vision.

David Rowntree founded the London-based animation company "Nanomation" in 1999, where he works both as an animator and developer of novel Softimage shaders.

Peter M. Hall's first degree is in physics with astrophysics and his PhD degree is in scientific visualization. He is a lecturer (assistant professor) in the Department of Computer Science, University of Bath, appointed in 1999. His research interests are in computer vision and computer graphics. He in an executive member of the British Machine Vision Association and chairs the UK Vision, Video, and Graphics.