A Mid-Level Description of Video, with Application to Non-photorealistic Animation

J. P. Collomosse and P. M. Hall,
Department of Computer Science,
University of Bath,
Bath, BA2 7AY, UK.
jpc | pmh @cs.bath.ac.uk

Abstract

The contribution of this paper is a novel spatiotemporal description of real video sequences. Our description comprises a set of surfaces that separate objects in the video, and an accompanying database which describes objects in the video. We explain how to automatically process video into our description, and use it to solve a long-standing problem in Computer Graphics; that of automatically producing non-photorealistic (NPR) animations from video sequences. We show empirically that our description is highly compact relative to alternative coding schemes, and that our NPR animation technique out performs the current state-of-the-art in automatic video painting by about an order of magnitude, in terms of temporal coherence.

1 Introduction

In this paper we introduce a novel mid-level description for encoding content in real video sequences. Although our principal contribution is this video description technique, its development was motivated by an application in Computer Graphics — specifically, the problem of synthesising non-photorealistic (NPR) animations from video sequences. Processing video for use by the entertainment industry is just one aspect of contemporary Computer Vision, and forms part of a more general convergence trend between Computer Vision and Computer Graphics. Our video description not only solves a long standing problem in Computer Graphics but also encompasses a range of artistic styles and enables novel effects to be produced.

We consider video as a spatiotemporal volume, with time as the third dimension. The interpretation of video as a volume, rather than as time-sequence frames, was proposed as far back as the early eighties [15] and has been successfully applied to the fields of both Computer Vision (for example, CBIR [21]) and Computer Graphics (for example, interactive video manipulation [12, 18]). By treating video as a three-dimensional volume, frequency patterns in both the spatial and temporal dimensions can be analysed in a conceptually simple way and the trajectories of objects may be treated as volumes. We are particularly interested in segmenting and tracking objects within the volume, and wish to aggregate object properties (such as colour) over time.

Instead of the traditional $3D$ view of a video volume [8, 9], we adopt a "two-spatial dimension plus time" ($2D+t$) view. We reason a $2D+t$ approach is to be preferred over a $3D$ approach, because (a) small, fast-moving objects may form disconnected volumes and these are much easier to handle, and (b) we have access to a wide range of standard 2D segmentation algorithms. We segment each frame independently, and combine the results across frames to generate a spatiotemporal description of the whole video. Similar approaches to associating contours over time have been described in the medical vision literature [10, 11, 17], we differ by using temporal association using region-based properties rather than edges. Broadly, segmented regions in adjacent frames are matched using criteria such as colour and shape to create objects. Surfaces are fitted into the spatiotemporal volume that separate objects from one another. The connectivity of the object is represented by a graph, and a database stores information that is useful to our NPR application. The result is a compact description of the video; a detailed account of its production is given in Section 2.

Our video description was designed to allow the automated production of coherent non-photorealistic video. Video painting is currently based upon optical flow: paint strokes (short, coloured line segments) are painted in frame one, and moved to frame two by optical flow vectors, and so on. This has many problems, the foremost being a distracting, visually uncomfortable flickering. Our video description solves this problem, as explained in Section 3. In the same section we also show, empirically, that our description can be more compact than MPEG-4 for cartoon animations, and that it reduces flicker by an order of magnitude compared to state-of-the-art automatic methods for video painting. We conclude, in Section 5, that processing images into artwork benefits from novel mid-level computer vision — and that computer vision may benefit by considering applications beyond its usual domain.

2 A Description of Video

Our description treats video as a $2D+t$ volume. It comprises a set of spatiotemporal surfaces that separate objects; such objects are spatiotemporal sub-volumes which describe the trajectories of features over time. Each surface separates exactly two objects, so that many such surfaces make up the boundary of an object. Surfaces each have two pointers associated with them — these pointers index a database that describes the objects on either side of the surface, so comprising a *winged edge structure* [1].

There are five stages to the production of such a description. 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 objects. The third stage removes spurious association, and temporally smooths the objects. Fourth, we fit separating surface between objects, initiating the winged-edge structure. Finally we produce a database of object attributes to be indexed by the pointers of the separating surfaces.

2.1 Frame Segmentation

Our first aim is to independently segment each video frame into homogeneous regions. In common with many segmentation techniques [2, 9, 22] we segment on the basis of homogeneous colour. This was chosen for initial simplicity only, and is not a fundamental limit

or objection to 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 loosely as the ability to produce near-identical results given 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 algorithm robustness, as we have defined it, is not present in the literature. Consequently, we empirically investigated the robustness of contemporary 2D segmentation algorithms. Experimental details, including a more formal definition of robustness, can be found elsewhere [5]. Here, we state the conclusion of that experiment: that EDISON [4] proved the most robust algorithm under our definition.

2.2 Region matching

Frame segmentation partitions each frame into a set of regions. A single region in a given frame may be associated with zero, one, or more regions in adjacent frames. This is because a region may be "split" by a passing foreground object, or "merge" if the foreground object moves away, for example. We therefore associate a single region in a given frame with a set of regions in adjacent frames. Associations are constructed by considering individual region pairs, R_i in frame i and R_j in an adjacent frame j. The quality of match between these regions is measured by

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

$$e(R_i,R_j) = w_1\sigma(R_i,R_j) + w_2\alpha(R_i,r_j) - w_3\delta(R_i,R_j;\Delta) - w_4\gamma(R_i,r_j) \quad (2)$$

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 $w_{[1..4]}$ are parameters that 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 typical values for the videos we present in Section 3. The function $\gamma(.)$ is the the Euclidean distance between the mean colours of the two regions in *CIELAB* space (normalised 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 normalised to be of equal area, $\sigma(.)$ is then computed by taking Fourier descriptors of the angular description function [7] of each region's boundary. Shape similarity is inversely to 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.

The set of regions that match R_i is constructed by the following greedy algorithm. The matching set is initiated to empty, and the area (in pixels) of R_i is computed as an "area-count". A potential set of matching regions is identified, using the distance limit Δ, in the

adjacent frame, and a score for each in 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_i is subtracted from each term. The matching set extends down this list until either the score of 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. These sub-volumes are broken into, possibly many, temporally convex *objects*. A property of the temporally convex representation is that many separate objects can merge to produce one object, and a single object division can split into many objects. We therefore generate a graph structure, with objects as nodes, specifying how objects split and merge over time.

2.3 Filtering and Smoothing

The description thus far comprises 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 6 frames (about $\frac{1}{4}$ second). The "holes" left by cutting such objects are filled by extrapolating from immediate neighbours. A serendipitous effect of this process is that poorly segmented areas of the video tend to merge to form one large coherent object.

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 bi-cubic Catmull-Rom patches [3], which are interpolating splines. Fitting is performed via a generalisation of active contours [16] to surfaces:

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

the internal energy is

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

and the external energy is

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

Function $f(.)$ is the Euclidean distance of the 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, so allowing variation in the degree of temporal smoothing in the sequence.

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 re-quantised to return to a voxel representation. These surfaces are then discarded, since they were used only as a means to perform temporal smoothing. We continue by fitting *separating surfaces* and generating a database of object properties.

2.4 Separating surfaces

Spatiotemporal objects are represented in terms of *separating surfaces* that mark the boundary between exactly two neighbouring objects. Each connected component in the boundary of an object pair has its own separating surface. Separating surfaces may contain internal "holes", if the boundary between neighbouring object has holes. Each separating surface has a *winged edge structure* [1], meaning it contains pointers referencing a database, which holds information about the objects which it separates.

Separating surfaces are preferred over storing each object in terms of its own bounding surface because boundary information is not duplicated and the description 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 subsection). Separating surfaces are not fitted to objects that are 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 separating 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 subsection 2.3. Secondly any holes in the connected component are accounted for by fitting a surface patch over each hole. The separating surface then comprises those points in the bounding surface that are not in any of the "hole" surfaces.

The boundary between objects is defined in the following way. 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 using in the fitting process.

2.5 The Database

We maintain a database that stores information about objects. The database is indexed by the pointers held in the separating surfaces' winged edge structure. Its purpose is to store information that is useful to applications. For example, we store the object's colour averaged over the spatiotemporal object — other specifics are best left to Section 3. Some data, though, is common to all applications; generic data comprises 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 suddenly revealed by the 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 because the end of the video has been reached.

Clearly, these attribute encode the graph structure that links objects (see Section 2.2). The separating surfaces and this database together comprise the video description we have been working towards throughout this section. Of course, most applications will require additional information, and this is certainly the case in our application to NPR animation, as we demonstrate in Section 3.

2.6 User Correction

Processing the video description is strongly dominated by automation, but Segmentation is imperfect, and user correction is needed for some sequences. Extensive correction of a sequence is rarely required, and correction itself takes little time since a couple of mouse clicks that either associates spatiotemporal objects over or causes objects to merge (we bias segmentation toward over segmentation).

3 An Application to NPR Animation

Our video description technique was designed with a specific purpose in mind: temporally coherent (flicker-free) artistic animations from video. Processing two-dimensional content into artwork is an area of significant interest in the field of non photorealistic rendering (NPR). However, the problem of automatically processing video into NPR animations has been sparsely researched. The literature that does exist concentrates primarily on the problem of producing coherent painterly animations [14, 19, 20].

The majority of NPR techniques process images into artwork by compositing multiple virtual "brush strokes"; in their simplest form, blobs of colour. Attributes of strokes, such as their colour or orientation, are derived from local image data. This can assist in the retention of edge detail, in the artwork, but this paradigm for painting does not extend well to video. The current state-of-the-art is to paint strokes in frame one, and transform strokes to subsequent frames using optical flow [19, 20]. Unfortunately, this approach gives rise to a distracting "flicker", the frequency of which is very close to optimal for attracting human attention [5]. The result is that flickering competes with video content for visual attention, leading to headaches for audience, or else a significant degree of manual intervention by animators.

Using optical flow to produce video artwork is analogous to producing strokes, in that local signal information is used: images and video are currently processed into artwork using low-level computer vision methods. We argue that mid-level vision techniques should be used to process images and video into artwork; high-level techniques are currently too specific to be of general use, so mid-level offers a good balance between interesting vision and practical application. This philosophy is expressed within this work via our mid-level description of video.

Given our spatiotemporal video description, any frame is on a plane of constant time that intersects the volume. The separating surfaces can therefore be rendered as lines, and the interiors of objects as areas. This immediately gives us the opportunity to render video in artwork styles that were not possible hitherto. For example, we can create line

renderings, or paint regions in a flat colour (as we alluded earlier, attributes such as colour can be added to the database in an application-specific field). We can shrink volumes, giving the appearance that the colour in an area does not quite reach up to the region boundary. We can choose to render some regions by accessing voxels from the original volume, and therefore create "mixed media" renderings in which some parts of the video look painted while other parts look photorealistic. We can even key-in from an alternative video source, so that the subject of a video sequence can be given a completely new background, for example.

We can also move strokes coherently, removing the problem of flicker. To do this we compute the inter-frame (planar projective) homographies for a specified object, 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, and use that frame as the basis on a canvas on which to paint. These homographies are of use if we want to repaint an object, replacing one face with another for example. Of course, the replacement object can be animated, and in this way we support rotoscoping. It turns out that affine transforms are better suited for carrying paint strokes from frame to frame, and we compute the closest affine transform (in a least squares sense) to each homography. Thus we can produce flicker-free painterly effects in video by considering painting and rotoscoping as essentially the same task.

Because temporal incoherence has been brought under control, the corollary is that we can re-introduce incoherences in a controlled way, if we so choose. For example, the separating surfaces can be "shattered" into small shards, each of which exhibits temporal coherence over a short time. Each shard is subjected to a small, random affine transformation to produce a sketchy effect when intersected by the time plane during rendering. Alternatively, we may introduce undulations in the surfaces, which causes the lines to "wobble".

4 Some Comparative Results

Figure 2 summarises 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 favourably with the other video compression algorithms tested; 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 determines the video content, whilst the client may determine 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 content visualisation between the client and server is a promising direction for development of video description techniques.

We also compared our animation system with the state-of-the-art based video painting algorithm based on optical flow [20]. 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

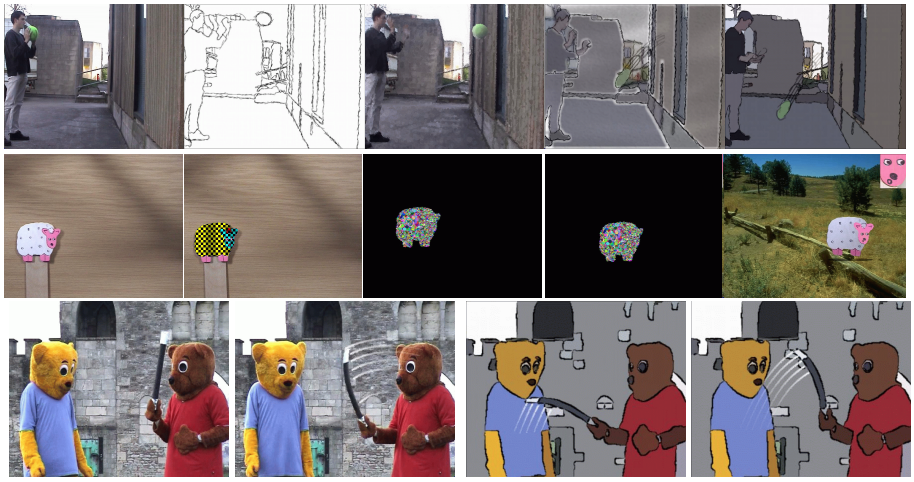
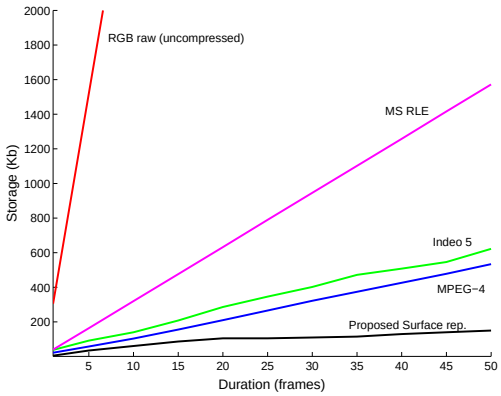


Figure 1: A gallery of some NPR effects supported by our video description (these animations are included in electronic material accompanying this paper). Top row: mixed media effects, showing sequential frames from different renderings of the same video sequence. Middle row: demonstrates placing a rigid reference-frame, stoke coherence, rotoscoping and matting using a simple example. Bottom row: frames from a single animation that integrates video painting with motion emphasis cues.

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 [20], and so 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, colour, 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 state-of-the-art, one in which the "bunching/holes" problem was not resolved (SoA-), and one in which it was (SoA). Numeric results are shown in Figure 2 (right) for three test videos (see accompanying material), in which we have normalised scores so that state-of-art is unity. It is clear that we offer about an order of magnitude improvement over state of the art.

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 [13], 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 are of all



FLICKER RATE

	SoA	SoA-	Ours
spheres	1	0.72	0.08
bounce	1	0.82	0.14
sheep	1	0.78	0.16

Figure 2: Left: Demonstrating the comparatively low storage requirements of the surface representation for up to 50 frames of a typical gradient-shaded cartoon animation. Right: Quantifying the lower rate of stroke flicker obtained via our method.

these experiments available in [5].

5 Conclusion

We have introduced a mid-level description of video, and used it to solve the long-standing problem of temporal coherence in video painting; indeed we not only solve flickering but provide a single framework for matting and rotoscoping, and allow many novel effects that are unique to our NPR application. We have been able to successfully integrate our animation technique with earlier motion emphasis work [6], enabling complete cartoon style animations to be generated from video with a high degree of automation.

Perhaps the most significant criticism of the work 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 description would be extended to increase generality of all artistic styles: three-dimensional descriptions of objects, or curvilinear (rather than linear) bases are two possible directions for development.

Despite these, and other limitations, the video description has proved very useful. and shown the utility of mid-level vision to NPR; especially if that mid-level vision is specifically designed for NPR. Equally, although our video description was motivated by a specific application it may find use elsewhere: considering problems raised by an application outside computer vision's normal domain can lead to interesting vision problems.

A full description of our NPR animation system can be found in [5]. A selection of rendered video clips accompanies this paper, and more may be found on-line at `http://www.cs.bath.ac.uk/~vision/cartoon`.

References

[1] B. Baumgart. A polyhedral representation for computer vision. In *National Computer Conference*, pages 589–596, 1975.

[2] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color and texture-based segmentation using EM and its application to content-based image retrieval. In *Proceedings of Intl. Conference on Computer Vision (ICCV)*, pages 675–682, 1998.

[3] Edwin E. Catmull and Raphael J. Rom. A class of local interpolating splines. *Computer Aided Geometric Design*, pages 317–326, 1974.

[4] C. Christoudias, B. Georgescu, and P. Meer. Synergism in low level vision. In *Conference of Pattern Recognition*, Quebec City, Canada, August 2001.

[5] J. P. Collomosse. *Higher Level Techniques for the Artistic Rendering of Images and Video*. PhD thesis, University of Bath, May 2004.

[6] J. P. Collomosse, D. Rowntree, and P. M. Hall. Video analysis for cartoon-like special effects. In *Proceedings BMVC*, volume 2, pages 749–758, Norwich, September 2003.

[7] R. L. Cosgriff. Identification of shape. Technical Report 820-11, ASTIA AD 254792, Ohio State Univ. Research Foundation, Columbus, Ohio USA, 1960.

[8] D. DeMenthon. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *Proc. Statistical Methods in Video Processing Workshop*, 2002.

[9] Y. Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[10] H. Fuchs, Z. M. Kedmen, and S. P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 10(20):693–702, 1977.

[11] S. Ganapathy and T. G. Dennehey. A new general triangulation method for planar contours. In *Proc. Computer Graphics (ACM SIGGRAPH)*, volume 16, pages 69–75, 1982.

[12] M. Gardner and S. G. Nikolov. Special film effects using spatio-temporal volume processing. In *Proc.* 1st *IEE Conference on Visual Media Production*, pages 145–154, March 2004.

[13] T. Gautama and M. Van Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Transactions on Neural Networks*, 5(13):1127–1136, 2002.

[14] A. Hertzmann and K. Perlin. Painterly rendering for video and interaction. In *Proceedings 1st ACM Symposium on Non-photorealistic Animation and Rendering*, pages 7–12, 2000.

[15] B. Jahne. *Spatio-temporal image processing*. Springer-Verlag, 1993.

[16] M. Kass, A. Witkin, and D. Terzopoulos. Active contour models. *Intl. Journal of Computer Vision*, 1(4):321–331, 1987.

[17] E. Keppel. Approximating complex surfaces by traingulation of contour lines. *IBM Journal of Research and Development*, 19:1–96, January 1975.

[18] A. W. Klein, P. J. Sloan, R. A. Colburn, A. Finkelstein, and M. F. Cohen. Video cubism. Technical report, Microsoft Research MSR-TR-2001-45, May 2001.

[19] L. Kovacs and T. Sziranyi. Creating video animations combining stochastic paintbrush transformation and motion detection. In *16th Intl. Conference on Pattern Recognition*, volume II, pages 1090–1093, 2002.

[20] P. Litwinowicz. Processing images and video for an impressionist effect. In *Proceedings Computer Graphics (ACM SIGGRAPH)*, pages 407–414, Los Angeles, USA, 1997.

[21] J.-Y. Pan and C. Faloutsos. Videocube: A novel tool for video mining and classification. In *Proc.* 5th *Intl. Conference on Asian Digital Libraries (ICADL)*, pages 11–14, July 2002.

[22] J.-P. Wang. Stochastic relaxation on partitions with connected components and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 20(6):619–636, 1994.

[23] D. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. *CVGIP: Image Understanding*, 55(1):14–26, 1992.