Video Analysis for Cartoon-like Special Effects

J. P. Collomosse[1], D. Rowntree[2] and P. M. Hall[1]
[1] Department of Computer Science, University of Bath, Bath, England.
[2] Nanomation Ltd, 6 Windmill Street, London, England.
{jpc | pmh} @ cs.bath.ac.uk

Abstract

In recent years the Vision community has shown interest in processing images and video for use by the entertainment industries. Typical applications include 3D reconstruction of models, and rendering graphics models into video. This paper broadly aligns with that trend, but differs in that we process video to emphasise motion in Cartoon-like styles, in which moving objects deform in defiance of physical laws, and leave trailing marks of one kind or another in their wake. We provide an introduction to the effects real animators use, and show how a judicious choice of standard processing techniques, supplemented by novel methods, can be used to achieve convincing results. We illustrate the robustness of our method using several video sequences, ranging in content from simple oscillatory to articulated motion, under both static and moving camera conditions.

1 Introduction

Processing video for use by the entertainment industries is just one aspect of contemporary Computer Vision. More generally the convergence of Computer Vision, Computer Graphics, and to a lesser extent Speech and Audio Scene Analysis, is driving a wide range of novel applications. To give just a few examples: there is interest in acquiring 3D models from video, either for scenery [1], or actors [2]; others work entirely in 2D, mosaicing images [3]; or generating novel views [4]. This area of work attracts as much interest from the Computer Graphics community, for example research in image-based rendering [5], or texture motion synthesis [6].

The vast majority of work in this convergence area uses photographic imagery as its foil. Our objective differs; we wish to process video into cartoon-like styles, and as such are aligned with the non-photorealistic rendering (NPR) community within Computer Graphics. NPR is an unfortunately broad term which ranges from the automatic, artistic rendering of 3D models [7] to the interactive processing of photographs for painterly effect [8]. As the NPR community tends increasingly toward automation, so they increasingly rely on ever more sophisticated methods from the Vision community.

The problem of non-photorealistic animation decomposes into two sub-goals: producing temporally coherent shading effects in the video, and emphasising motion in the image sequence. This paper is concerned with the latter problem which raises many Computer Vision issues, including camera motion compensation, tracking, occlusion handling, depth recovery, and trajectory analysis. The literature is sparse concerning the production

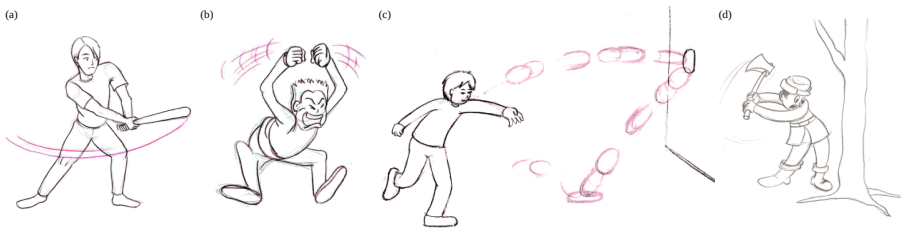

Figure 1: Examples of motion cues used in traditional animation: two examples of streak line augmentation cues (a,b), the latter with ghosting lines. Two examples of deformation cues; squash and stretch (c) and suggestion of inertia through deformation (d).

of non-photorealistic animations from video; the majority of techniques focus upon rendering video in painterly styles [9, 10]; aiming to mitigate against, rather than emphasise, motion for the purpose of coherence. To the best of our knowledge an attempt to visually emphasise motion within video is a novel contribution, and one that implies interesting new application areas for Computer Vision.

Animators have evolved various ways of emphasising the characteristics of a moving object (Figure 1). Streak lines are commonly used to emphasise motion, and typically follow the movement of the tip of the object through space. The artist can use additional 'ghosting' lines which indicate the trailing edge of the object as it moves along the streak lines. Ghosting lines are usually perpendicular to streak lines. Deformation is often used to emphasise motion, and a popular technique is squash and stretch in which a body is stretched tangential to its trajectory, while conserving area [11]. Other deformations can be used to emphasise an object's inertia; a golf club or pendulum may bend along the shaft to show the end is heavy and the accelerating force is having trouble moving it. The magnitude of deformation is a function of motion parameters such as tangential speed, and of the modelled rigidity of the object. In this paper we process real video to introduce these motion cues; examples are given in Figure 1 (c.f. Figures 4, 5).

In an influential paper [11], Lasseter introduces many of these techniques to the Computer Graphics community, but presents no algorithmic solutions. Recent work addresses one of these techniques by applying a squash and stretch effect to spheres and cylinders in object space prior to ray-tracing [12]. Strothotte *et al* [13], after Hsu *et al* [14], also identify depiction of motion as important, though the former are concerned primarily with the effect of motion cues on temporal perception. In both studies streak lines are generated via user-interactive processes.

2 Video analysis for motion emphasis

We begin by tracking objects (such as an arm, leg, bat or ball) whilst compensating for camera motion. Objects are assigned a depth ordering so that motion cues may later be inserted at the correct depth in the scene. The trajectories of objects are then analysed in order to determine the placement of motion cues, and to evaluate variables which are blended with user defined parameters to determine the final appearance of motion cues.

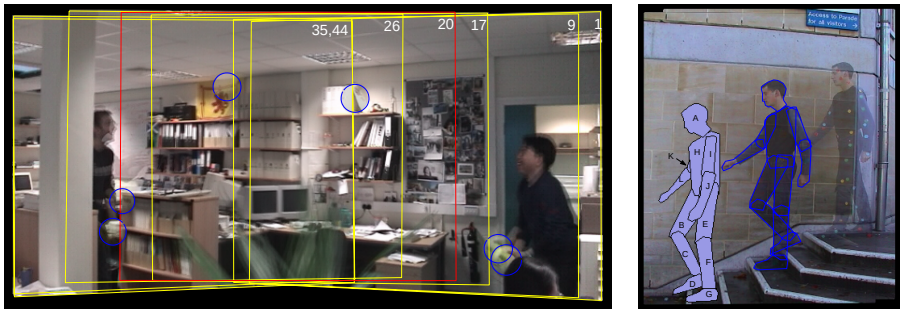

Figure 2: Left: The camera compensated *VOLLEY* sequence sampled at regular time intervals, camera viewport outlined in yellow, tracked feature outlined in blue. Right: *STAIRS* sequence. Markers are required to track this more complex subject (top) but are later removed automatically (middle). Recovery of relative depth ordering permits compositing of features in the correct order (bottom).

2.1 Preliminaries

We compensate for camera motion using a robust motion estimation technique proposed by Torr [15]. Harris interest points [16] are identified in adjacent video frames, and RANSAC [17] use to produce an initial estimate of the homography between frames. This estimate is then refined using a Levenburg-Marquadt iterative search [3]. Frames are projected via their homographies to produce a motion compensated sequence in which the tracking of features is subsequently performed.

We use a template based tracker based on colour, considering only the hue and saturation components of the HSV model to affect simple luminance invariance. Users identify objects (templates) to be tracked by drawing polygons in a single frame, which are "shrink wrapped" to the object's edge contour [18]. We assume contour motion is well approximated by a linear conformal affine transform (LCAT) in the image plane, which has 4 parameters (uniform scale s, orientation θ, and spatial position u,v); in homogeneous coordinates we have a product of matrices: $\mathbf{M}(s,\theta,u,v) = \mathbf{T}(u,v)\mathbf{R}(\theta)\mathbf{S}(s)$. Variation of these parameters is assumed to well approximate a second order motion equation over short time intervals. In a similar manner to camera motion correction, several well distributed interest points [16] are identified automatically within the object. The LCAT $\mathbf{M}_{tt'}$ from frame t to frame t' is initially estimated by RANSAC, and then refined by Levenburg-Marquadt search to minimise mean squared pixel error, $D[\mathbf{M}_{tt'}]$, between template and target region.

In cases where point correspondences for tracking can not be found (perhaps due to signal flatness, small feature area, or self-similarity), distinctively coloured markers may be physically attached to the subject in place of Harris interest points, and later removed automatically.

We handle occlusion in a novel way. The likelihood L_t of the feature being *visible* at any time t may be written as a function of detected pixel error $L_t = exp(-\lambda D[\mathbf{M}_{tt'}])$; λ is the reciprocal of the average time an object is unoccluded. At each frame t we pass the estimated LCAT $\mathbf{M}_{tt'}$, and the confidence in that estimate, L_t, through a Kalman filter to obtain our optimal estimate for the LCAT. The Kalman filter state describes the second order motion equation in the 4D parameter space of the LCAT, trained over the immediate

history of contour motion. We threshold L_t at 0.5 to decide whether an object is occluded in a given frame, and interpolate the LCAT from unoccluded neighbours in these cases.

Finally, we use a novel approach to determine a partial depth ordering for tracked features, based on their mutual occlusion over time. Additional assumptions are introduced at this stage: the physical ordering of tracked features cannot change over time, and a tracked feature can not be both in front and behind another tracked feature. The result is the assignment of an integer value to each feature corresponding to its relative depth from the camera.

The reader is referred to [19] for fuller accounts of both the tracking and depth recovery algorithms.

2.2 Correspondence trails, deformation bases, and occlusion buffers

We have observed that streak lines, ghosting, and squash and stretch are common elements of traditional animation. Animators tend to sketch elegant, long curved streaks which emphasise motion over a local but extended history. Ostensibly, streak lines can be produced on a per frame basis by attaching lines to an object's trailing edge, tangential to the direction of motion [14]. Unfortunately, such an approach is only suitable for visualising instantaneous motion, and only produces straight streak lines. Optical flow cannot be used to create streak lines, primarily because it gives point trajectories.

In this section we show how to construct *correspondence trails*, *deformation bases*, and *occlusion buffers*, all of which are used when rendering. It is in this and the rendering section that we provide the most novelty. We begin with an object, which has been tracked over the full course of a video, as already described. We analyse tracking data and show how to produce an *animated object* which is a re-presentation of the original, subject to the placement of augmentation cues (streak lines and ghosting), deformation cues (squash and stretch and other distortions), and suitable occlusion handling. We work entirely with two-dimensional data.

The trajectory of the object is characterised by the trajectory of its centroid, $\mu(t)$. Given a tracked object, this is easy to acquire. This trajectory is segmented into piecewise smooth sections, delimited by G^1 discontinuities. Because we have fixed each frame into a common basis (via the inter-frame homographies) we can estimate the observed velocity, $\dot{\mu}(t)$ and acceleration, $\ddot{\mu}(t)$, of the centroid. The centroid trajectory plays a central role in establishing deformation bases and correspondence trials, as we now explain — beginning with deformation bases.

A local time-window selects a section of the centroid trajectory, and this window moves with the object. The instantaneous spatial width of this window is proportional to instantaneous centroid velocity. At each instant, t we use the centroid trajectory to establish a curvilinear basis frame. First we compute an arc-length parameterisation of the trajectory: $\mu(r)$ in which $r = \int_t^{t'} \dot{\mu}(\mathbf{t})dt$, note $t' < t$ gives negative displacements. Next we develop an ordinate at an arc-length distance r from the instant; the unit vector $\mathbf{n}(r)$ perpendicular to $\dot{\mu}(\mathbf{t})$. Thus, at each instant t we can specify a point in the world-frame using two coordinates, r and s:

$$\mathbf{x}_t(r,s) = \mu(r) + s\mathbf{n}(r) \tag{1}$$

We can write this more compactly as $\mathbf{x}_t(\mathbf{r}) = \mathbf{C}(\mathbf{r})$, where $\mathbf{r} = [r,s]^T$, and maintain the

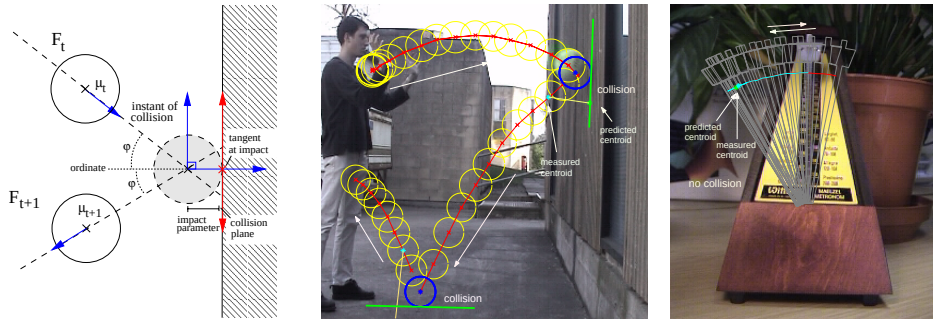

Figure 3: Collision geometry (left); bounces are detected as collisions (middle), whilst the simple harmonic motion of the metronome is not (right).

inverse function $\mathbf{r}_t(\mathbf{x}) = \mathbf{C}^{-1}(\mathbf{x})$ via a look-up table. This mapping, and its inverse, comprise one example of a *deformation basis*.

The above analysis holds for most points on the centroid trajectory. It breaks down at *collision points*, because there is a discontinuity in velocity. We define a collision point as a point on a trajectory whose location cannot be satisfactorily predicted by a second order motion equation (constructed with a Kalman filter). This definition discriminates between G^1 discontinuities in trajectory which are formed by, say, simple harmonic motion, and true collisions which are C^1 discontinuous (see Figure 3).

At a collision point we establish an orthonormal basis set aligned with the observed collision plane. We assume the angle of incidence and reflection are equal, and hence the unit vector which bisects this angle is taken to be the ordinate. The abscissa lies in the collision plane. We define this new basis set as an additional instance of a deformation basis, and write the mapping to and from the world frame using notation consistent with the curvilinear basis. In addition, we compute the *impact parameter* of the collision, which we define as the distance from the object's centroid to its boundary, in the direction of the negative ordinate. Note we compensate for temporal sampling around the true collision instant by intersecting extrapolated impact and rebound trajectories (Figure 3).

We will use deformation bases to drive the deformation cues, but first we explain the role of the centroid trajectory when creating a set of *correspondence trails*, which are used to render augmentation cues. A trailing edge is a subset of boundary points $\mathscr{T}(t) = \{\mathbf{p}(t) : \mathbf{m}(t)^T \dot{\mathbf{p}}(t) < 0\}$ where $\mathbf{p}(t)$ is a sampled boundary point and $\mathbf{m}(t)$ its outward normal.

We form a correspondence trail by establishing a correspondence between each pair of consecutive trailing edges: $\mathscr{T}(t)$ and $\mathscr{T}(t')$. Correspondence is established by first aligning $\mu(\mathbf{t})$ and $\mu(\mathbf{t}')$, and then aligning the unit tangents $\hat{\mu}(\mathbf{t})$ and $\hat{\mu}(\mathbf{t}')$, finally scaling using the scale parameter taken from the LCAT at time t. Nearest neighbours (Euclidean distance) in the transformed trailing edges are deemed to correspond. Correspondence trails are given a piecewise smooth representation by analogy with the centroid trajectory. There is no reason to suppose corresponding trails are parallel, which makes them ill suited to establishing deformation bases. We explain their use in rendering augmentation cues a little later (Section 3), but first round-off our discussion on video analysis elements.

We maintain a time-dependent *occlusion buffer* that records which pixels occlude the tracked object at any given instant. At each time instant, a pixel in the buffer is flagged

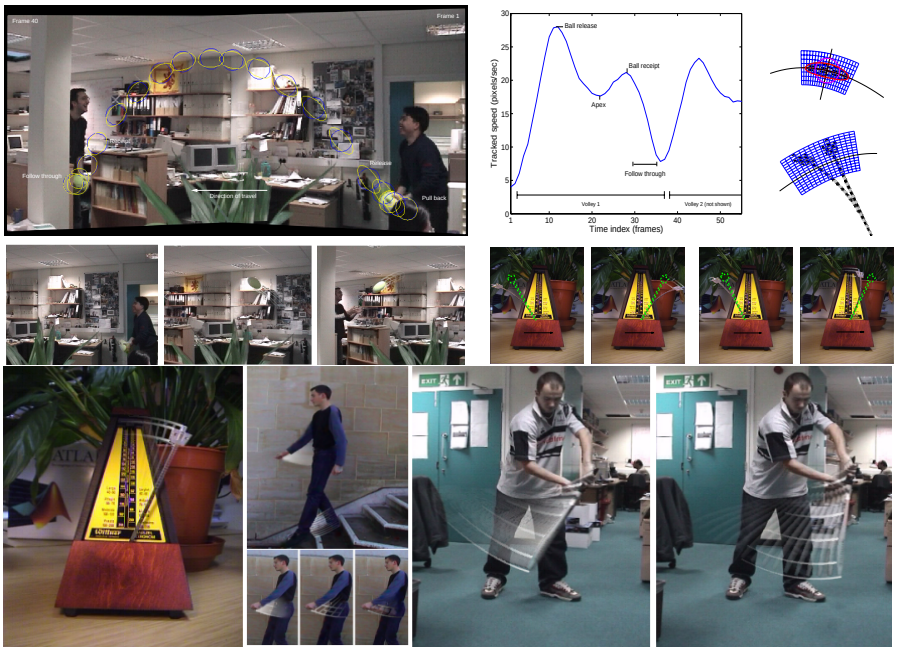

Figure 4: Top: Illustrating the squash and stretch effect; eccentricity varies as a function of tangential speed. A motion dependent curvilinear space is used to create deformation cues. Middle: Frames from the *VOLLEY* sequence exhibiting squash and stretch. Observe the system handles both large scale camera motion, and lighting variation local to the ball. Frames from *METRONOME* suggesting drag and inertia through (6). Bottom: Examples of the wide gamut of augmentation cues available. Ghosting lines may be densely sampled to emulate motion blur effects or more sparsely for traditional ghosting. Varying the overlap constant w influences spacing of streak lines through the objective function (7).

as occluding the object, if the Euclidean distance between observed colour and predicted colour (from the template) exceeds a threshold. Colour is represented by the hue and saturation components of the HSV colour model. The set of occlusion buffers is used during rendering, which we describe next.

3 Rendering motion cues

Rendering is performed on a per-frame basis, by compositing layers in back-to-front depth order. The background layer comprises each frame with the tracked objects removed; holes are filled-in by sampling texture from neighbouring frames.

The next layers contain the animation objects, each of which comprises a potentially deformed object and possibly augmentation cues. The final layer of pixels are composited from the occlusion buffer.

An animated object is, in general, a deformed version of the original. Squash and stretch tangential to instantaneous motion leads to visually unattractive results; it is better to not only squash and stretch, but also to bend the object along the arc of its centroid

trajectory. Let $\mathbf{x}(t)$ be any point in the object. We transform this point with respect to a single deformation basis into a new point $\mathbf{y}(t)$, given by

$$\mathbf{y}(t) = C(A_t[C^{-1}(\mathbf{x}(t))]) \tag{2}$$

where $A_t[.]$ is some deformation. In the case of squash and stretch the deformation is an area-preserving differential scale that depends on instantaneous speed $|\dot{\mu}(t)|$:

$$A = \begin{bmatrix} k & 0 \\ 0 & \frac{1}{k} \end{bmatrix} \tag{3}$$

$$k = 1 + \frac{K}{2}(1 - cos(\pi \frac{v^2+1}{2}))$$

$$v = \begin{cases} 0 \text{ if } |\dot{\mu}| < V_{min} \\ 1 \text{ if } |\dot{\mu}| >= V_{max} \\ (|\dot{\mu}| - V_{min})/(V_{max} - V_{min}) \text{ otherwise} \end{cases} \tag{4}$$

where K limits the eccentricity of the squash and stretch effect, and V_{max}, V_{min} define a velocity window for the effect; these constants are user defined. This stretches the object in its direction of motion (along the arc of its trajectory), and gives a compensatory squash in the normal direction for each point in the object.

Around collision points we need to squash and stretch the other way around – so that the object compresses on impact. We linearly interpolate between the deformation caused by a "standard" deformation basis with that caused by a "collision" deformation basis. Suppose $\mathbf{p}(t) \mapsto \mathbf{q}(t)$ and $\mathbf{p}(t) \mapsto \mathbf{q}'(t)$, respectively, then:

$$\mathbf{r}(t) = f(d)\mathbf{q}(t) + (1 - f(d))\mathbf{q}'(t) \tag{5}$$

where $f(d) = sin(\frac{\pi}{2} argmin(1, d/(sD))$ in which D is the impact parameter of the collision, and s is a user parameter which controls the spatial extent over which collision influences the deformation. As a note, the mapping to $\mathbf{q}'(t)$ not only has to scale the object but shift it toward the impact point, so that the edge of the deformed object touches the collision plane.

Non-linear deformations are also possible, and these can be used to create bending effects. We can form warping functions which depend on each point's velocity and acceleration as well as its position. We write $x' = C(A_t[C^{-1}(\mathbf{x}), \dot{\mathbf{x}}, \ddot{\mathbf{x}}])$, where A is a functional used, for example, to suggest increased drag or inertia. A typical functional operates on each component of $\mathbf{r} = (r_1, r_2)^T$ independently; to create effects suggesting drag we use:

$$r_1 \leftarrow r_1 - F(\frac{2}{\pi} atan(|\dot{\mathbf{x}}_i|))^P sign(\dot{\mathbf{x}}_i) \tag{6}$$

where F is a function of suggested mass, and P influences the apparent rigidity of the object. By substituting acceleration for velocity, and adding rather than subtracting from r_1 we can emphasise inertia of the feature (see Figure 4).

We now describe the rendering of augmentation cues. We place streak lines only on correspondence trails which, at any given instant, survive the following filtering process. Our filtering selects curves based on heuristics derived from the practice of traditional animators who favour placement of streak lines on sites of high curvature and on an object's convex hull. Long streak lines and streak lines associated with rapid motion are

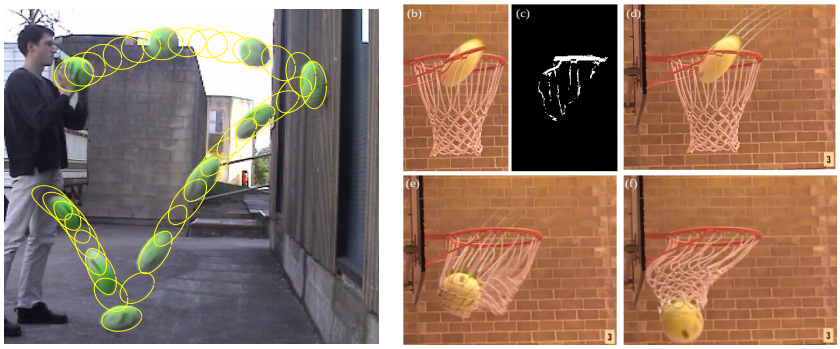

Figure 5: Left: (a) Time lapse image of the *BOUNCE* sequence (c.f. Figure 1c). Right: (b) naïve deformation creates artefacts, but our method does not (d). An occlusion buffer is constructed over time, allowing augmentation cues to be handled in the wake of the feature. The system breaks down (e) after impact erratic movement of the netting causes the occlusion buffer to empty and cues to be incorrectly drawn in front of that netting.

also preferred, but close proximity to other co-existing streak lines is discouraged. We select streak line curves, on each iteration i adding a new element to σ (initially empty) to maximise the recursive function $H(.)$

$$\begin{aligned} H(0) &= 0 \\ H(i+1) &= H(i) + (\alpha v(x) + \beta L(x) - \gamma D(x) - \delta \omega(x, \sigma; w) + \zeta \rho(x)) \end{aligned} \tag{7}$$

where x is the set of points associated with a piecewise smooth section of a correspondence trail. $L(x)$ is the length of a smooth section, $v(x)$ is the "mean velocity" defined as $L(x)/t(x)$ in which $t(x)$ is the duration of x. $\rho(x)$ is the mean curvature of feature boundary at points in x. $D(x)$ is the mean shortest distance of points in x from the convex hull of the feature. $\omega(x, \sigma; w)$ measures the maximal spatio-temporal overlap between x and the set of streak lines chosen on previous iterations. From each curve we choose points which co-exist in time, and plot the curves with width w returning the intersected area. Constant w is user defined, as are the constant weights $\alpha, \beta, \gamma, \delta$, and ζ; these give artistic control over the streak line placement (see Figure 4). Iteration stops when the additive measure falls below a lower bound.

Correspondence trails are used to render streak lines and ghosting effects. Both effects are spatio-temporal in nature. A streak line is made visible at some absolute time t and exists for a duration of time Δ. The streak line is rendered by sweeping a translucent disc (backwards in time) along a smooth section of a correspondence trail. The disc grows smaller and more transparent over time; these decays are under user control. Secondary streak lines may be generated at small spatio-temporal offsets to produce sketchy or turbulent effects.

Ghosting lines depict the position of an object's trailing edge along the path of the streak line, and are useful in visualising velocity changes over the course of the streak. Ghosting lines are rendered by sampling the trailing edge at regular time intervals as the streak line is rendered, interpolating if required. The opacity of ghosting lines is not only a function of time (as with streak lines) but also a function of speed relative to other points on the trailing edge; this ensures only fast moving regions of edge are ghosted. Users may

control the sampling rate, line thickness, and decay parameters to stylise the appearance of the ghosting lines.

Thus we can deform and augment an object to make an animated object, which is rendered on top of the general background. It remains to render pixels on top of the animated object to recover occlusion cues. This is complicated by the fact that because the animated object is a deformed version of the original – and includes augmentation cues, we need to consider occlusion buffers both forward and backward in time; in fact any buffer whose lit pixels intersect a mask covering the complete animated object must be considered. At each pixel we select that foreground colour from the occlusion buffer which is nearest in time to the instant (frame) being rendered. This is possible as the whole of the animated object lies within the locus of points described by all points of the moving object.

4 Results and concluding remarks

We have described and demonstrated a system for the artistic rendering of motion within video sequences. Motion cues closely approximating those used in traditional animation are produced automatically (Figure 1). The system can cope with a moving camera, lighting changes, and presence of occlusion. Users may stylise both the placement and appearance of motion cues using the parameterised framework described in Section 3. Key to our work is the analysis of trajectories, including collision detection and the formation of correspondence trails.

There are many aspects of the system that may be improved. For example, the tracker could benefit from contemporary methods, and occlusion handling could be improved. The robustness of the algorithm could be evaluated both with ground truth comparisons for measures such as velocity, as well as processing sequences exhibiting distinctly non-planar motion.

We believe the most productive avenues for future work will not be in incremental refinements to the current system, but rather will examine alternative uses for higher-level spatio-temporal analysis of video with applications to NPR. In particular we are investigating a spatio-temporal approach to producing coherent artistic effects in video [20], and would like to address additional forms of motion emphasis as described by Lasseter [11].

A selection of source and rendered video sequences are available for download at `http://www.cs.bath.ac.uk/~vision/cartoon`.

References

[1] A. Zisserman T. Werner, "Model selection from automated architectual reconstruction from multiple views," in *British Machine Vision Conference*, 2002, pp. 53–62.

[2] P. Fua R. Plankers, "Articulated soft objects for video-based body modeling," in *IEEE International Conference on Computer Vision*, 2001, pp. 394–401.

[3] R. Szeliski, "Image mosaicing for tele-reality applications," Tech. Rep., Digital Equipment Corporation, 1994.

[4] I. Reid K. Connor, "Novel view specification and synthesis," in *British Machine Vision Conference*, 2002, pp. 243–252.

[5] L. Williams S. Chen, "View interpolation for image synthesis," in *Proceedings Computer Graphics (ACM SIGGRAPH)*, 1993, pp. 291–298.

[6] D. H. Salesin I. Essa A. Schodl, R. Szeliski, "Video texture," in *Proceedings Computer Graphics (ACM SIGGRAPH)*, 2000, pp. 489–498.

[7] B. Meier, "Painterly rendering for animation," in *Proceedings Computer Graphics (ACM SIGGRAPH)*, 1996, pp. 447–484.

[8] P. Haeberli, "Paint by numbers: abstract image representations," in *Proceedings Computer Graphics (ACM SIGGRAPH)*, 1990, vol. 4, pp. 207–214.

[9] P. Litwinowicz, "Processing images and video for an impressionist effect," in *Proceedings Computer Graphics (ACM SIGGRAPH)*, 1997, pp. 407–414.

[10] A. Hertzmann and K. Perlin, "Painterly rendering for video and interaction," in *Proceedings NPAR Symposium*, 2000, pp. 7–12.

[11] J. Lasseter, "Principles of traditional animation applied to 3D computer animation," in *Proceedings Computer Graphics (ACM SIGGRAPH)*, July 1987, vol. 21, pp. 35–44.

[12] S. Chenney, M. Pingel, R. Iverson, and M. Szymanski, "Simulating cartoon style animation," in *Proceedings NPAR Symposium*, 2002.

[13] T. Strothotte, B. Preim, A. Raab, J. Schumann, and D. R. Forsey, "How to render frames and influence people," in *Proceedings Computer Graphics Forum (Eurographics)*, 1994, vol. 13, pp. C455–C466.

[14] S. C. Hsu and I. H. H. Lee, "Drawing and animation using skeletal strokes," in *Proceedings Computer Graphics (ACM SIGGRAPH)*, 1994, pp. 109–118.

[15] P. H. S. Torr, *Motion segmentation and outlier detection*, Ph.D. thesis, University of Oxford, 1995.

[16] C. J. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings 4th Alvey Vision Conference*, Manchester, 1988, pp. 147–151.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[18] D. Williams and M. Shah, "A fast algorithm for active contours and curvature estimation," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 14–26, 1992.

[19] J. P. Collomosse, D. Rowntree, and P. M. Hall, "Cartoon-style rendering of motion from video," in *Vision, Video and Graphics*, July 2003, pp. 117–124.

[20] J. P. Collomosse, D. Rowntree, and P. M. Hall, "Stroke surfaces: A spatio-temporal framework for temporally coherent non-photorealistic animations," Tech. Rep. 2003–01, University of Bath, UK, June 2003.