Video Object Annotation, Navigation, and Composition

Dan B Goldman[1] *Chris Gonterman*[2] *Brian Curless*[2] *David Salesin*[1,2] *Steven M. Seitz*[2]

[1]Adobe Systems, Inc.
801 N. 34th Street
Seattle, WA 98103
{dgoldman,salesin}@adobe.com

[2]Computer Science & Engineering
University of Washington
Seattle, WA 98105-4615
{gontech, curless, salesin, seitz}@cs.washington.edu

ABSTRACT

We explore the use of tracked 2D object motion to enable novel approaches to interacting with video. These include moving annotations, video navigation by direct manipulation of objects, and creating an image composite from multiple video frames. Features in the video are automatically tracked and grouped in an off-line preprocess that enables later interactive manipulation. Examples of annotations include speech and thought balloons, video graffiti, path arrows, video hyperlinks, and schematic storyboards. We also demonstrate a direct-manipulation interface for random frame access using spatial constraints, and a drag-and-drop interface for assembling still images from videos. Taken together, our tools can be employed in a variety of applications including film and video editing, visual tagging, and authoring rich media such as hyperlinked video.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Interaction styles.

General terms: Design, Human Factors, Algorithms

Keywords: Video annotation, video navigation, video interaction, direct manipulation.

1 INTRODUCTION

Present video interfaces demand that users think in terms of frames and timelines. Although frames and timelines are useful notions for many editing operations, they are less well suited for other types of interactions with video. In many cases, users are likely to be more interested in the higher level components of a video such as motion, action, character, and story. For example, consider the problems of attaching a moving annotation to an object in a video, or finding a moment in a video in which an object is in a particular place, or of composing a still from multiple moments in a video. Each of these tasks revolves around objects in the video and their motion. But although it is possible to compute object boundaries and to track object motion, present interfaces do not utilize this information for interaction.

In this paper we propose a framework using (2D) video object motion to enable novel approaches to user interaction. Although the concept of object motion is not as high-level as character and story, it is a mid-level aspect of video that we believe is a crucial building block toward those higher-level concepts. We apply computer vision techniques to understand how various points move about the scene and segment this motion into independently moving objects. This information is used to simplify the interaction required to achieve a variety of video manipulation tasks.

In particular, we propose novel interfaces for three tasks that are under-served by present-day video interfaces: annotation, navigation, and image composition.

Video annotation. Video annotation is the task of associating graphical objects with moving objects on the screen. In existing interactive applications, only still images can be annotated, as in the "telestrator" system used in American football broadcasting (see Figure 1). Using our system, however, such annotations can easily be attached to moving objects in the scene by novices with minimal user effort. Our system supports descriptive labels, illustrative sketches, thought and word bubbles communicating speech or intent, path arrows indicating motion, and hyperlinked regions (Figures 3–7). Given the pre-computed object motion, the system can also determine when objects are occluded or change appearance significantly, and modify the appearance of the annotations accordingly (Figure 8). We envision video object annotations being used in any field in which video is produced or used to communicate information.

Video navigation. The use of motion analysis also permits novel approaches to navigating video through direct manipulation. Typical approaches to navigating video utilize a linear scan metaphor, such as a slider, timeline, or fast-forward and rewind controls. However, using pre-computed object motion, the trajectories of objects in the scene can be employed as constraints for direct-manipulation navigation: In our system, clicking and dragging on objects in a video frame causes the video to immediately advance or rewind to a frame in which the object is located close to the ending mouse position. Contemporaneous work [7, 11] shows that this approach streamlines video tasks requiring selection of individual frames (for example, locating a "cut frame" on which to begin or end a shot). Our system also enables a novel mode of re-animation of video sequences using the mouse as input: We can direct the motion of an object such as a face by dragging it around the screen. This interface can be applied to "puppeteer" existing video, or to retime the playback of a video.

Video-to-still composition. Finally, we discuss the task of rearranging portions of a video to create a new still image. Agarwala *et al.* [1] have previously explored the problem of combining a set of stills into a single, seamless composite. However, when the source material is video, the task of identifying the appropriate moments becomes a problem in and of itself. Many consumer cameras now support a "burst mode" in which a high-speed series of high-resolution photos is taken. Finding Cartier-Bresson's "decisive moment" in such an image stream, or composing a photomontage from multiple moments, is a significant challenge. Rav-Acha *et al.* [19] have proposed editing videos using evolution of time-fronts, but have not presented a user interface for doing so. We demonstrate an interactive interface complementary to such algorithms, for composition of novel still images using a drag-and-drop metaphor to manipulate people or other objects in a video.

To achieve these interactions, our system first analyzes the video in a fully automatic preprocessing step that tracks the motion of image points across the video and segments those tracks into coherently moving groups. Although reliably extracting objects in video and tracking them over many frames is a hard problem in computer vision, the manipulations we support do not require perfect object segmentation and tracking, and can instead exploit low-level motion tracking and mid-level grouping information. Furthermore, aggregating point motion into coherent groups has a number of benefits that are critical for interaction: We can select a moving region using a single click, estimate object motion more robustly, and, to a limited extent, handle changes of object appearance, including occlusions. Our contributions include an automated preprocess for video interaction that greatly enriches the space of interactions possible with video, an intuitive interface for creating graphical annotations that transform along with associated video objects, a fluid interaction technique for scrubbing through video with single or multiple constraints, and a novel drag-and-drop approach to composing video input into new images by combining regions from multiple frames.

1.1 Related work

The telestrator [30], popularly known as a "John Madden-style whiteboard," was invented by physicist Leonard Reiffel for drawing annotations on a TV screen using a light pen. This approach has also been adopted for individual sports instruction using systems like ASTAR [3] that aid coaches in reviewing videos of athletic performance. However, as previously mentioned, annotations created using a telestrator are typically static, and do not overlay well on moving footage.

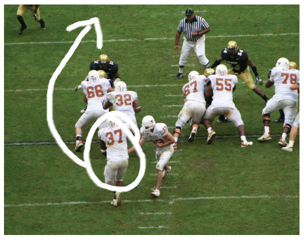


Figure 1: A typical telestrator illustration. (©Johntex, CC license [30])

In recent years, broadcasts of many professional sporting events have utilized systems supplied by Sportvision [28] to overlay graphical information on the field of play even while the camera is moving. Sportvision uses a variety of technologies to accomplish these overlays, including surveying, calibration, and instrumentation of the field of play or racing environment. Although this instrumentation and calibration allow graphics to be overlaid in real time during the broadcast, it requires expensive specialized systems for each different class of sporting event, and is not applicable to pre-existing video acquired under unknown conditions.

Tracking has previously been used for video manipulation and authoring animations. For example, Agarwala *et al.* [2] demonstrated that an interactive keyframe-based contour tracking system could be used for video manipulation and stroke animation authoring. However, their system required considerable user intervention to perform tracking. In contrast, our application does not require pixel-accurate tracking or object segmentation, so we can use more fully-automated techniques that do not produce pixel segmentations.

Our method utilizes the particle video approach of Sand and Teller [23] to densely track points in the video. Object tracking is a widely researched topic in computer vision, and many other tracking approaches are possible; Yilmaz *et al.* [31] recently surveyed the state of the art. However, particle video is especially well suited to interactive video applications because it provides a dense field of tracked points that can track fairly small objects, and even points in featureless regions. An important advantage of the particle video approach over other methods is that it produces tracks that are both spatially dense and temporally long-range.

Our grouping preprocess accomplishes some of the same goals as the object grouping technique of Sivic *et al.* [26], which tracks features using affine-covariant feature matching and template tracking, followed by a grouping method employing co-occurrence of tracks in motion groups. That method has shown significant success at grouping different views of the same object even through deformations and significant lighting changes. However, after some experimentation we found that it has several drawbacks for our application, which we discuss in depth in Section 6.

Balakrishnan and Ramos [18] also developed a system with a novel navigation and annotation interface to video. However, their approach is based on a linear timeline model, and annotations apply only to individual frames in a video.

Thought and speech balloons have previously been employed in virtual worlds and chat rooms [15, 13], in which the associated regions are known a priori. Kurlander *et al.* [13] specifically address the problem of balloon layout. However, their system was free to assign the placement of both the word balloons and the subjects speaking them, whereas our system is designed to associate thought and speech balloons with arbitrary moving video objects.

We are not the first to propose the notion of hyperlinked video as described in Section 4.1. To our knowledge, the earliest reference of this is the Hypersoap project [6]. However, the authoring tool proposed in that work required extensive user annotation of many frames. Smith *et al.* [27] taxonomized hyperlinks in dynamic media, but their implementation is limited to simple template tracking of whole objects. We believe our system offers a significantly improved authoring environment for this type of rich media.

Numerous previous works discuss composition of a still or a short video from a longer video, for the purposes of moving texture synthesis [25], modifying dialogue [4], animating video sprites [24], or video summarization [19, 20, 16, 17]. In this paper we advocate and demonstrate the approach of drag-and-drop manipulation as a tool for interactively directing such compositions.

Our system features a novel interface for scrubbing through video using direct manipulation of video objects. This technique is similar in spirit to the storyboard-based scrubbing approach of Goldman *et al.* [8], but permits manipulation directly on the video frame, rather than on an auxiliary storyboard image. The system of Goldman *et al.* required several minutes of user interaction to author a storyboard before this manipulation can be performed. In contrast, our approach requires only automated preprocessing to enable direct-manipulation navigation. Indeed, we propose that our navigation system can be used as part of an authoring system to create such storyboards, as described in Section 4.3. Our video navigation mechanism can be used to re-animate video of a person, in much the same way as the spacetime faces system [32], but without requiring a complex 3D shape acquisition system.

Kimber *et al.* [12] introduced the notion of navigating a video by directly manipulating objects within a video frame. They also demonstrate tracking and navigation across multiple cameras. However, their method uses static surveillance cameras and relies on whole object tracking, precluding navigation on multiple points of a deforming object such as we demonstrate in Figure 10.

Contemporaneous works by Karrer *et al.* [11] and Dragicevic *et al.* [7] use flow-based preprocessing to enable real-time interaction that is very similar to our approach described in Section 4.2. Their research demonstrates that direct manipulation video browsing permits significant performance improvements for frame selection tasks.

However, our work advances this concept in four important ways: First and foremost, this paper presents a general framework enabling a number of different kinds of direct video manipulations, not only navigation. Second, our preprocessing method achieves the long-range accuracy of object-tracking [12] and feature-tracking methods [7], while also retaining the spatial resolution provided by optical-flow-based techniques [11]. Our use of motion grouping also improves the robustness of the system to partial occlusions, and makes it possible to track points well even near the boundaries of objects, points which might otherwise "slip" off of one object onto another at some frame, causing an incorrect trajectory. Third, our "starburst" manipulator widget effectively represents the space of both simple and complex object motions, whereas the motion path arrows employed in earlier work are most effective at representing simple, smooth motion paths. The starburst widget is more effective in situations such as Figure 10, in which it is more important to convey the range of feasible motion than an object's specific path through the video (in this case, a spiral). Fourth, our introduction of multiple constraints enables simple access to more elaborate object configurations, also as shown in Figure 10. Finally, we introduce an inertial slider mechanism allowing access to frames in which an object is off-screen.

In some respects, the approaches of Karrer *et al.* and Dragicevic *et al.* have advantages over our framework. Most notably, both methods use a less expensive preprocessing approach than ours, potentially making them more widely applicable in practice. Both methods also include terms in their cost functions that discourage discontinuous jumps in time, which may be desirable for certain applications.

1.2 Overview

Our system consists of several off-line preprocessing stages (Section 2), followed by an interactive interface for video selection that is common to all our techniques (Section 3). The subsequent section describes applications; our interfaces for video annotation (Section 4.1), navigation (Section 4.2), and recomposition (Section 4.3).

2 PRE-PROCESSING

Our preprocessing consists of two phases. First, point particles are placed and tracked over time (Section 2.1). Second, the system aggregates particles into consistent moving groups (Section 2.2).

2.1 Particle tracking

To track particles, we apply the "particle video" long-range point tracking method [23, 22], which takes as input a sequence of frames and produces as output a dense cloud of *particles*, representing the motion of individual points in the scene throughout their range of visibility. Each particle has a starting and ending frame, and a 2D position for each frame within that range.

The key advantage of the particle video approach over either template tracking or optical flow alone is that it is both spatially dense and temporally long-range. In contrast, feature tracking is long-range but spatially sparse, and optical flow is dense but temporally short-range. Thus, particle video data is ideal for our applications, as we can approximate the motion of any pixel into any other frame by finding a nearby particle.

In the sections that follow, we will use the following notation: A particle track i is represented by a 2D position $\mathbf{x}_i(t)$ at each time t during its lifetime $t \in T(i)$. The total number of particles is denoted n.

2.2 Particle grouping

For certain annotation applications, we find it useful to estimate groups of points that move together over time. Our system estimates these groupings using a generative K-affines motion model, in which the motion of each particle i is generated by one of K affine motions plus isotropic Gaussian noise:

$$\mathbf{x}_i(t+\Delta t) = A_{L(i)}(t)[\mathbf{x}_i(t)] + \eta \tag{1}$$

Here $A_k(t)$ represents the affine motion of group k from time t to time $t+\Delta t$, and η is zero-mean isotropic noise with standard deviation σ. (For notational convenience we write $A_k(t)$ as a general operator, rather than separating out the linear matrix multiplication and addition components.) In our system, $\Delta t = 3$ frames. Each particle is assigned a single group label $1 \leq L(i) \leq K$ over its entire lifetime. The labels $L(i)$ are distributed with unknown probability $P[L(i) = k] = \pi_k$. We denote group k as $\mathbf{G}_k = \{i | L(i) = k\}$.

Our system optimizes for the maximum likelihood model

$$\Theta = (A_1, \ldots, A_K, \pi_1, \ldots, \pi_K, L(1), \ldots, L(n)) \tag{2}$$

using an EM-style alternating optimization. Given the above generative model, the energy function Q can be computed as:

$$Q(\Theta) = \sum_i \sum_{t \in T(i)} \left(\frac{d(i,t)}{2\sigma^2} - \log(\pi_{L(i)}) \right) \tag{3}$$

where $d(i,t) = ||\mathbf{x}_i(t + \Delta t) - A_{L(i)}(t)[\mathbf{x}_i(t)]||^2$, the residual squared error.

To compute the labels $L(i)$, we begin by initializing them to random integers between 1 and K. Then, the following steps are iterated until Q converges:

- Affine motions A_k are estimated by a least-squares fit of an affine motion to the particles $\mathbf{G}_k$.
- Group probabilities π_k are computed as the numbers of particles in each group, weighted by particle lifespan, then normalized such that $\sum_k \pi_k = 1$.
- Labels $L(i)$ are reassigned to the label that minimizes the objective function $Q(\Theta)$ per particle, and the groups $\mathbf{G}_k$ are updated.

In the present algorithm we fix $\sigma = 1$ pixel, but this could be included as a variable in the optimization as well.

The output of the algorithm is a segmentation of the particles into K groups. Figure 2 illustrates this grouping on one of our input datasets. Although there are some misclassified particles, the bulk of the particles are properly grouped. Our interactive selection interface, described in Section 3, can be used to overcome the minor misclassifications seen here.

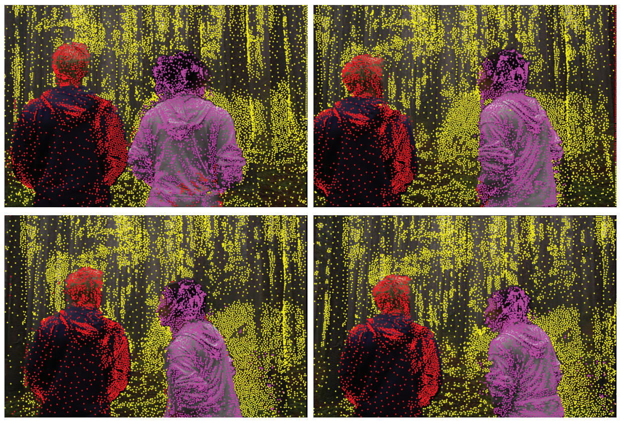


Figure 2: Four frames from a video sequence, with particles colored according to affine groupings computed in Section 2.2. (video footage ©2005 Jon Goldman)

3 VIDEO SELECTION

Several of our interactions require the selection of an object in the video, either for annotation or direct manipulation of that object. The object specification task is closely related to the problems of interactive video segmentation and video matting [2, 5, 14, 29]. However, these other methods are principally concerned with recovering a precise matte or silhouette of the object being selected. In contrast, the goal of our system is to recover the *motion* of a particular object or region of the image, without concern for the precise location of its boundaries.

In our system, the user selects video objects simply by painting a stroke or dragging a rectangle over a region $\mathbf{R}_s$ of the image. This region is called the *anchor region*, and the frame on which it is drawn is called the *anchor frame*, denoted t_s. The anchor region defines a set of *anchor tracks*. For some applications, it suffices to define the anchor tracks $\mathbf{M}(s)$ as the set of all particles on the anchor frame that lie within the anchor region:

$$\mathbf{M}(s) = \{i | t_s \in T(i), \mathbf{x}_i(t_s) \in \mathbf{R}_s\} \tag{4}$$

However, this simplistic approach to selecting anchor tracks requires the user to scribble over a potentially large anchor region. The system can reduce the amount of user effort by employing the particle groupings computed in section 2.2. Our interface uses the group labels of the particles in the anchor region to infer entire group selections, rather than individual particle selections. To this end, the system supports two modes of object selection. First, the user clicks once to select the group of points of which the closest track is a member. The closest track i^* to point $\mathbf{x}_0$ on frame t_0 is located as:

$$i^*(\mathbf{x}_0, t_0) = \text{argmin}_{\{i | t_0 \in T(i)\}} ||\mathbf{x}_0 - \mathbf{x}_i(t_0)||, \tag{5}$$

and the selected group is simply $\mathbf{G}_{L(i^*(\mathbf{x},t))}$. Second, the user paints a "sloppy" selection that includes points from multiple groups. The resulting selection consists of the groups that are well represented in the anchor region. Each group is scored according to the number $|\mathbf{M}_k(s)|$ of its particles in the anchor region s. Then any group whose score is a significant fraction T_G of the highest scoring group is accepted:

$$\mathbf{M}_k(s) = \mathbf{G}_k \cap \mathbf{M}(s) \quad \forall 1 \leq k \leq K \tag{6}$$

$$S_k(s) = \frac{|\mathbf{M}_k(s)|}{\max_{1 \leq k \leq K} |\mathbf{M}_k(s)|} \tag{7}$$

$$\mathbf{G}(s) = \bigcup_{\{k | S_k(s) \geq T_G\}} \mathbf{G}_k \tag{8}$$

The threshold T_G is a system constant that controls the selection precision. When $T_G = 1$, only the highest-scoring group $\text{argmax}_k |\mathbf{M}_k(s)|$ is selected. As T_G approaches 0, any group with a particle in the selected region will be selected in its entirety. We have found that $T_G = 0.5$ generally gives intuitive results.

Our system's affine grouping mechanism may group particles together that are spatially discontiguous. However, discontiguous regions are not always desired for annotation or manipulation. To address this, only the particles that are spatially contiguous to the anchor region are selected. This effect is achieved using a connected-components search through a pre-computed Delaunay triangulation of the particles on the anchor frame.

Both the group-scoring formula and connected components search take only a fraction of a second, and can therefore be recomputed on the fly while a paint stroke is in progress in order to give visual feedback to the user about the regions being selected.

By allowing more than one group to be selected, the user can easily correct the case of over-segmentation, such that connected objects with slightly different motion may have been placed in separate groups. In the case of annotation, if a user selects groups that move independently, the attached annotations will simply be a "best fit" to both motions.

Using groups of particles confers several advantages over independent particles. As previously mentioned, user interaction is streamlined by employing a single click or a loose selection to indicate a moving object with complex shape and trajectory. Furthermore, the large number of particles in the object groupings can be used to compute more robust motion estimates for rigidly moving objects. The system can also annotate objects even on frames where the original anchor tracks $\mathbf{M}(s)$ no longer exist due to partial occlusions or deformations. (However, our method is not robust to the case in which an entire group is occluded and later becomes visible again. This is a topic for future work.)

4 APPLICATIONS

Our interactive interface is patterned after typical drawing and painting applications, with the addition of a video timeline. The user is presented with a video window, in which the video can be scrubbed back and forth using either a slider or a direct manipulation interface described in Section 4.2. A toolbox provides access to a number of different types of annotations and interaction tool modes, which are applied using direct manipulation in the video window.

4.1 Video annotations

Telestrator-like markup of video can be useful not only for sports broadcasting but also for medical applications, surveillance video, and instructional video. Film and video professionals can use annotations to communicate editorial information about footage in post-production, such as objects to be eliminated or augmented with visual effects. Annotations can also be used to modify existing video footage for entertainment purposes with speech and thought balloons, virtual graffiti, "pop-up video" notes, and other arbitrary signage.

In this Section, we describe our system's support for five types of graphical annotations, distinguished by their shape and by the type of transformations (e.g., translation / homography / similarity) with which they follow the scene. An annotation's motion and appearance is determined by the motion of its anchor tracks: Transformations between the anchor frame and other frames are computed using point correspondences between the features on each frame. Some transformations require a minimum number of correspondences, so if there are too few correspondences on a given frame — for example because the entire group is occluded — the annotation is not shown on that frame.

At present, we have implemented prototype versions of "graffiti," "scribbles," "speech balloons," "path arrows," and "hyperlinks."

Graffiti. These annotations inherit a perspective deformation from their anchor tracks, as if they are painted on a planar surface such as a wall or ground plane. Given four or more non-collinear point correspondences, a homography is computed using the normalized direct linear transformation method described by Hartley and Zisserman [10].

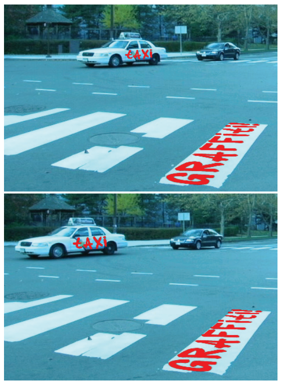


Figure 3: Graffiti

Depending on the length of the video sequence, it may be time-consuming to compute the transformations of an annotation for all frames. Therefore, after the user completes the drawing of the anchor regions, the transformations of graffiti annotations are not computed for all frames immediately, but are lazily evaluated as the user visits other frames. Further improvements to perceived interaction speed are possible by performing the computations during idle callbacks between user interactions.

Scribbles. These simple typed or sketched annotations just translate along with the mean translation of anchor tracks. This annotation is ideal for simple communicative tasks, such as local or remote discussions between collaborators in film and video production.

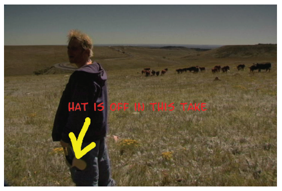


Figure 4: Scribbles (footage ©2005 Jon Goldman)

Word balloons. Inspired by Comic Chat [13] and Rosten *et al.* [21], our system implements speech and thought balloons that reside at a fixed location on the screen, with a "tail" that follows the annotated object. The location of the speech balloon is optimized to avoid overlap with foreground objects and other speech balloons, while remaining close to the anchor tracks.

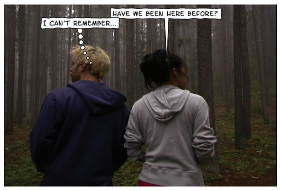


Figure 5: Word balloons (footage ©2005 Jon Goldman)

Path arrows. These annotations highlight a particular moving object, displaying an arrow indicating its motion onto a plane in the scene. To compute the arrow path we transform the motion of the centroid of the anchor tracks into the coordinate system of the background group in each frame. The resulting path is used to draw an arrow that transforms along with the background. By clipping the path to the current frame's edges, the arrow head and tail can remain visible on every frame, making it easy to determine the subject's direction of motion at any time. We believe this type of annotation could be used by surveillance analysts, or to enhance telestrator-style markup of sporting events.

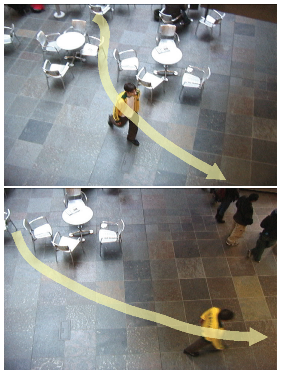
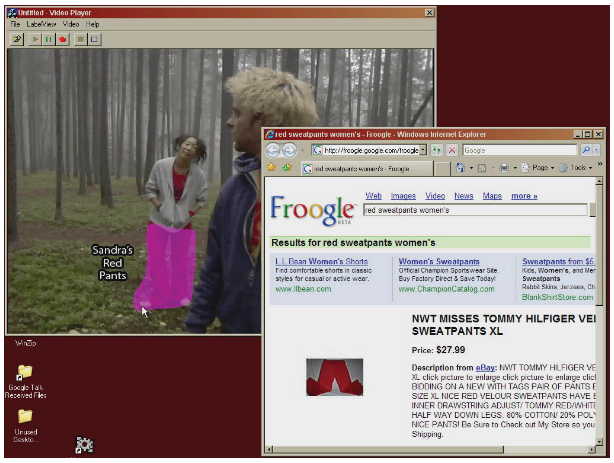


Figure 6: Path arrows

Video hyperlinks. Our system can also be used to author dynamic regions of a video that respond to interactive mouse movements, enabling the creation of hyperlinked video [6]. A prototype hyper-video player using our system as a front end for annotation is shown in Figure 7. When viewing the video on an appropriate device, the user can obtain additional information about annotated objects, for example, obtaining purchase information for clothing, or additional references for historically or scientifically interesting objects. As a hyperlink, this additional information does not obscure the video content under normal viewing conditions, but rather allows the viewer to actively choose to obtain further information when desired. The hyperlinked regions in this 30-second segment of video were annotated using our interface in about 5 minutes of user time.

Marking occlusions. When an object being annotated is partially occluded, our system can modify an associated annotation's appearance or location, either to explicitly indicate the occlusion or to move the annotation to an un-occluded region. One indication of occlusion is that the tracked particles in the occluded region are terminated. Although this is a reliable indicator of occlusion, it does not help determine when the same points on the object are disoccluded, since the newly spawned particles in the disoccluded region are not the same as the particles that were terminated when the occlusion occurred. Here again we are aided by the grouping mechanism, since it associates these points on either side of the occlusion as long as there are other particles in the group to bridge the two sets. To determine if a region of the image instantiated at one frame is occluded at some other frame, the system simply computes the fraction of particles in the transformed region that belong to the groups present in the initial frame. An annotation is determined to be occluded if fewer than half of the points in the region belong to the originally-selected groups. Figure 8 shows a rectangular annotation changing color as it is repeatedly occluded and disoccluded.


Untitled - Video Player
File LabelView Video Help
Sandra's Red Pants
red sweatpants women's - Froogle - Windows Internet Explorer
http://froogle.google.com/froogle
Google
red sweatpants women's - Froogle
Page
Tools
Froogle BETA
Web Images Video News Maps more »
red sweatpants women's
WinZip
Google Talk Received Files
Unused Deskto...

Results for red sweatpants women's

L.L.Bean Women's Shorts
Find comfortable shorts in classic styles for casual or active wear.
www.llbean.com

Women's Sweatpants
Official Champion Sportswear Site. Buy Factory Direct & Save Today!
www.ChampionCatalog.com

Sweatpants from $5.
Kids, Women's, and Mer
Sweatpants
Rabbit Skins, Jerzees, Ch
BlankShirtStore.com

NWT MISSES TOMMY HILFIGER VE
SWEATPANTS XL

Price: $27.99

Description from eBay: NWT TOMMY HILFIGER VE
XL click picture to enlarge click picture to enlarge clic
BIDDING ON A NEW WITH TAGS PAIR OF PANTS E
SIZE XL NICE RED VELOUR SWEATPANTS HAVE E
INNER DRAWSTRING ADJUST/ TOMMY RED/WHITE
HALF WAY DOWN LEGS. 80% COTTON/ 20% POL
NICE PANTS! Be Sure to Check out My Store so you
Shipping.

Figure 7: A video with highlighted hyperlinks to web pages. (video footage ©2005 Jon Goldman)

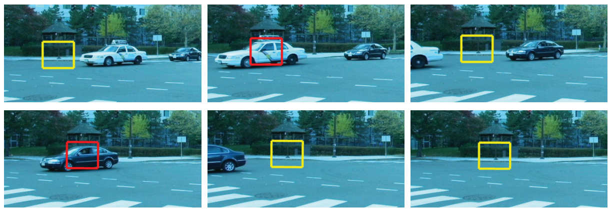


Figure 8: A rectangle created on the first frame sticks to the background even when its anchor region is partially or completely occluded. The annotation changes from yellow to red to indicate occlusion.

4.2 Video navigation using direct manipulation

Given densely tracked video, we can scrub to a different frame of the video by directly clicking and dragging on moving objects in the scene. We have implemented two variations on this idea: The first uses individual mouse clicks and drags, and the second uses multiple gestures in sequence.

The single-drag UI is implemented as follows: When the user clicks at location $\mathbf{x}_0$ while on frame t_0, the closest track i^* is computed as in equation (5), and the offset between the mouse position and the track location is retained for future use: $\mathbf{d} = \mathbf{x}_0 - \mathbf{x}_{i^*}(t_0)$. Then, as the user drags the mouse to position $\mathbf{x}_1$, the video is scrubbed to the frame t^* in which the offset mouse position $\mathbf{x}_1 + \mathbf{d}$ is closest to the track position on that frame:

$$t^* = \operatorname{argmin}_{\{t \in T(i^*)\}} \|\mathbf{x}_1 + \mathbf{d} - \mathbf{x}_{i^*}(t)\| \qquad (9)$$

Since this action mimics the behaviour of a traditional scrollbar or slider, we call it a "virtual slider." Figures 9 and 10(a-c) illustrate this behavior.

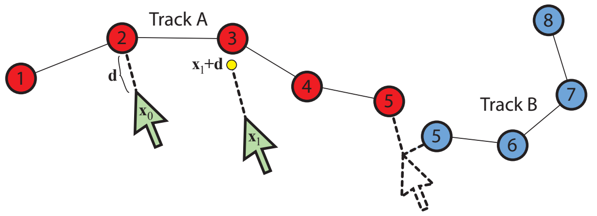


Figure 9: When the mouse is depressed at position $\mathbf{x}_0$ on frame 2, track A (shown in red) is selected as the virtual slider track. When the mouse moves to position $\mathbf{x}_1$, the location of that track at frame 3 is closer to the offset mouse position $\mathbf{x}_1 + \mathbf{d}$, so the video is scrubbed to frame 3. Track A ends at frame 5, but the virtual slider is extended to later frames using the next closest track (track B, shown in blue).

This approach can cause discontinuous jumps in time when manipulating objects that travel away from and then back to their original screen locations. Such jumps can be eliminated by computing distances in (x, y, t) [11] or $(x, y, \text{arc-length})$ [7] spaces, as demonstrated in contemporaneous work. However, the discontinuous jumps can actually be preferable in situations such as that shown in Figure 10, in which the user searches for a scene configuration without attending to exactly when in the video that configuration occurred. In such situations, ignoring temporal continuity gives more salient results and more immediate feedback.

Visualizing range of motion. To represent the range of motion afforded by a virtual slider, we can display it as a line or path arrow overlaid on the image. However, for complex movements this representation may be confusing. Therefore, we have developed a novel representation for displaying range of motion: A "starburst" widget is placed at the constraint location such that the length of the starburst's spikes represent the spatial proximity of nearby frames on the slider. For simple linear motions the starburst widget simply looks like a compass with one arm pointing to the direction of movement forwards in time and another pointing to the direction of movement backwards in time. However, for more complex motions additional arms begin to appear, indicating that portions of the range of motion lie in other directions.

To display the starburst widget, we first bin the slider positions according to their orientation from the current position $\mathbf{x}_{i^*}(t_0)$. Then, a weight h_θ is computed for each arm using all the positions in bin B_θ:

$$\mathbf{v}_t = \mathbf{x}_{i^*(t)} - \mathbf{x}_{i^*(t_0)} \tag{10}$$

$$h_\theta = \sum_{\{t | \mathbf{x}_{i^*(t)} \in B_\theta\}} \exp\left(-||\mathbf{v}_t||^2 / \sigma_h^2\right) \tag{11}$$

The length l_θ of the starburst arm in direction θ is then given by normalizing and scaling the weights:

$$l_\theta = k\sqrt{h_\theta / \sum_\omega h_\omega} \tag{12}$$

where k controls the absolute scaling of the starburst. Longer arms are given an arrowhead, and bins with no points do not display any arm at all. This formula is designed to weight the track locations by a Gaussian distance falloff, so that nearby points on the slider are weighted more heavily than distant ones. Since the binning described above can cause aliasing for slider paths that lie just along the boundaries of the orientation bins, we adjust the orientation of the bins on each frame using PCA, centering a bin on the dominant direction of the slider in the local region. Examples of the starburst manipulators in context can be see in Figures 10 and 11.

Handling occlusions. Because tracks can start and end on arbitrary frames, it is not always possible to reach a desired frame using a virtual slider constructed from a single track. Therefore we extend the virtual sliders using multiple tracks, as follows: If the last frame of the selected track $t_{\max} = \max\left[t \in T(i^*)\right]$ is not the end of the video, and the track is not at the edge of the frame, the system finds the next closest track j^* on frame $t_{\max}$ in the same group that extends at least to $t_{\max} + 1$. The portion of this track that extends beyond frame $t_{\max}$ is then appended to the virtual slider, offset by the displacement $\mathbf{x}_{i^*}(t_{\max}) - \mathbf{x}_{j^*}(t_{\max})$ between tracks in order to avoid a discontinuity (see Figure 9). This process is continued forwards until either the end of the video is reached, or the virtual slider reaches the edge of the frame, or the next-closest track is too distant. The process is then repeated in reverse for the first frame of the track, continuing backwards.

In most cases this algorithm successfully extends the virtual slider throughout the visible range of the object's motion, including partial occlusions. However, our system does not track objects through full occlusions, so the virtual slider typically terminates when the object becomes fully occluded. In order to overcome short temporary occlusions, we have added "inertia" to our virtual sliders, inspired by the inertial scrolling feature of Apple's iPhone: During mouse motion we track the temporal velocity of the virtual slider in frames per second. When the mouse button is released, this velocity decays over a few seconds rather than instantaneously. In this way, the user can reach frames just beyond the end of the virtual slider by "flicking" the mouse toward the end of the slider. This feature complements the accurate frame selection mechanism by providing a looser long-distance frame selection mechanism.

Some virtual sliders may have paths that fold back on themselves, or even spiral around, so that a linear motion of the mouse will jump nonlinearly to different frames. In this case, the temporal velocity of the slider when the mouse is released is not meaningful. Instead, we can retain the spatial constraint for several seconds, advancing its location using the *spatial* velocity of the mouse pointer before the button was released.

Unfortunately, spatial inertia cannot be used in all cases, because unlike temporal inertia, there is no way to reach past the end of the virtual sliders: Tracks do not extend beyond the boundaries of the video frame. Therefore our virtual sliders automatically choose between temporal inertia and spatial inertia by examining the predicted next position of the virtual slider: If they are in similar directions, temporal inertia is used. If they are in different directions, spatial inertia is used. We have found this heuristic tends to perform intuitively in most cases.

Multiple constraints. By extending the scrubbing technique described above to multiple particle paths, we also implement a form of constraint-based video control. The user sets multiple point constraints on different parts of the video image, and the video is advanced or rewound to the frame that minimizes the maximum distance from each particle to its constraint position. Here, c indexes over constraint locations $\mathbf{x}_c$, offsets $\mathbf{d}_c$, and constrained particles i_c^*, and $\mathbf{C}$ is the set of all constraints:

$$t^* = \operatorname{argmin}_{\{t \in T(i^*)\}} \max_{c \in \mathbf{C}} ||\mathbf{x}_c + \mathbf{d}_c - \mathbf{x}_{i_c^*}(t)|| \tag{13}$$

In our current mouse-based implementation, the user can sequentially set a number of fixed-location constraints and one dynamic constraint, controlled by the mouse. However, multiple dynamic constraints could be applied using a multi-

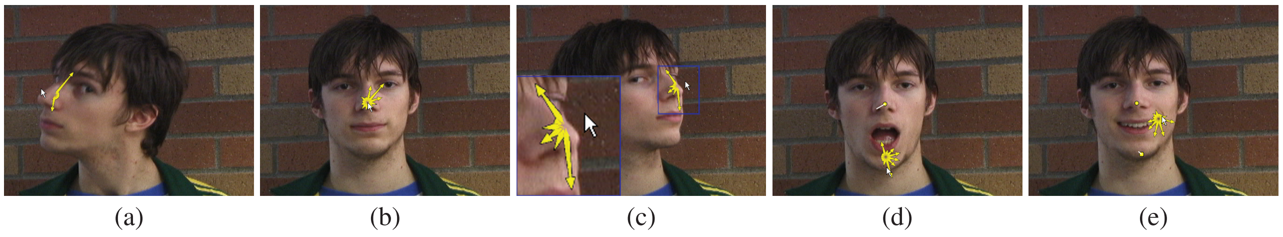


Figure 10: Our scrubbing interface is used to interactively manipulate a video of a moving head by dragging it to the left and right (a-c). Nearby frames are indicated by longer starburst arms. At the extremes of motion (c), some arms of the starburst disappear. Additional constraints are applied to open the mouth (d), or keep the mouth closed and smile (e).

touch input device. Figures 10(d) and 10(e) illustrate facial animation using multiple constraints.

To balance multiple constraints, we employ a max function in Equation 13 instead of a sum, because this function has optima at which the error for two opposing constraints is equal. As a simple example, consider two objects in a 1-dimensional scene that travel away from the origin at two different speeds: $x_a(t) = v_a t, x_b(t) = v_b t, v_a \neq v_b$. Now imagine that these objects are constrained to positions x_a^* and x_b^* respectively. If we were to use a sum of absolute values $|v_a t - x_a^*| + |v_b t - x_b^*|$ as our cost function, it would be minimized at either $t = x_a^*/v_a$ or $t = x_b^*/v_b$: One of the constraints will be met exactly while the other is ignored! Using the sum of squared errors for our cost function does not solve this problem, as it also meets the constraints unequally. However, by using the maximum distance in our cost function, the error is made equal for the two most competing constraints at the optimum.

4.3 Video-to-still composition

Another application enabled by our system is the seamless recomposition of a video into a single still image featuring parts of several different frames of the video. For example, one might wish to compose a still image from a short video sequence featuring multiple participants. Although there may be frames of the video in which one or two subjects are in frame, looking at the camera, and smiling, there may be no one frame in which all of the subjects look good. However, by using different frames of the video for different subjects we may be able to compose a single still frame that is better than any individual video frame in the sequence. Prior work [1] assumes that the number of frames is small enough that a user can examine each frame individually to find the best frame for each subject. Such interfaces are therefore less appropriate for video input.

In our system, we take a drag-and-drop approach to video recomposition. Virtual sliders, as described in the previous section, are used to navigate through the video. In addition, we display only the video object under the mouse at the new frame, and the rest of the objects are "frozen" at their previous frame. This is accomplished as follows:

First, if the camera is moving, we estimate the motion of the background as a homography, using the group with the largest number of tracked points to approximate the background. Subsequently, when other frames are displayed, they are registered to the current frame using the estimated background motion, and all virtual slider paths are computed using the registered coordinate frame[1]. As the mouse button is dragged and the frame changes, a rough mask of the object being dragged is computed using the same connected-components search described in Section 3. This mask is used to composite the contents of the changing frame over the previously selected frame. In this way, these regions appear to advance or rewind through time and the object moves away from its previous location. We also composite the new frame contents within the matte of the object at the frame upon which the mouse button was depressed, in order to remove its original appearance from that region of the image. Although this is not guaranteed to show the proper contents of that image region (for example, a different object may have entered that region at the new frame), it is simple enough to run at interactive rates and usually produces plausible results. When the mouse button is released, a final composite is computed using graph cut compositing [1] with the object mask as a seed region, removing most remaining artifacts. An illustration of drag-and-drop video recomposition is shown in Figure 11.

Our system also supports another type of still composition using video; the schematic storyboards described by Goldman *et al.* [8]. In that work, the user selected keyframes from an exhaustive display of all the frames in the video, and manually selected and labeled corresponding points in each of those frames. In contrast, our interface is extremely simple: The user navigates forward and backward in the video using either a standard timeline slider or the "virtual slider" interface, and presses a "hot key" to assign the current frame as a storyboard keyframe. When the first keyframe is selected, the interface enters "storyboard mode," in which the current frame is overlaid atop the storyboard as a ghosted image, in the proper extended frame coordinate system. Changing frames causes this ghosted image to move about the window, panning as the camera moves. The user can add new keyframes at any time, and the layout is automatically recomputed. As described by Goldman *et al.*, the storyboard is automatically split into multiple extended frames as necessary, and arrows can be associated with moving objects using the object selection and annotation mechanisms already described. The resulting interaction is much easier to use than the original Goldman *et al.* storyboards work.

5 INFORMAL EVALUATION

We believe the tools we have demonstrated are largely unique to our system. However, it is possible to associate annotations to video objects using some visual effects and motion

[1]This is identical to the "relative flow dragging" described by Dragicevic *et al.* [7]

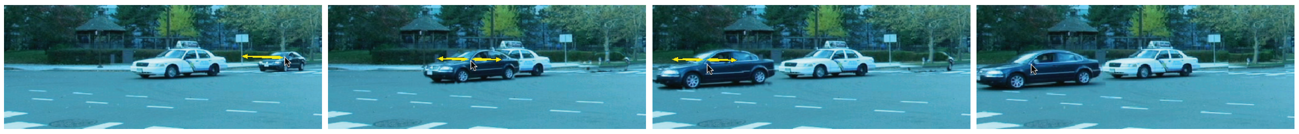


Figure 11: Drag-and-drop composition of a still from video. The black car is to the right of the white car on all frames of the input video. From left to right, the black car is dragged from its location on frame 1 to a new location on frame 66. The first three panels show the appearance during the drag interaction, and the fourth panel shows the final graph cut composite, which takes about 5 seconds to compute. Unlike a traditional image cut/paste, the black car appears larger as it is dragged leftwards, because it is being extracted from a later video frame.

graphics software. We asked a novice user of Adobe After Effects – a commercial motion graphics and compositing tool with a tracking module – to create a single translating "scribble" text annotation on the car shown in Figure 8. With the assistance of a slightly more experienced user, the novice spent over 20 minutes preparing this test in After Effects.

However, since After Effects is targeted at professionals, we also asked an expert user – a co-inventor and architect of After Effects – to perform the same annotation task using both After Effects and our "graffiti" tool. We gave him a quick 1 minute introduction to our tool, then asked him to annotate both the car and the crosswalk stripe with a deforming text label, as shown in Figure 3. Using our tool, which he had not previously used, he completed the task in just 30 seconds using 10 mouse clicks and 12 keystrokes. Using After Effects, the same task took 7 minutes and 51 seconds, using over 74 mouse clicks, 52 click-and-drags, and 38 keystrokes. In a second trial he performed the same task in 3 minutes and 55 seconds, using over 32 mouse clicks, 36 click-and-drags, and 34 keystrokes. Although After Effects offers more options for control of the final output, our system is an order of magnitude faster to use for annotation because we perform tracking beforehand, and we only require the user to gesturally indicate video objects, annotation styles and contents.

We also assessed the usability and operating range of our system by using it ourselves to create storyboards for every shot in the 9-minute short film, "Kind of a Blur" [9]. The main body of the film (not including head and tail credit sequences) was manually segmented into 89 separate shots. Of these, 25 were representable as single-frame storyboards with no annotations. For the remaining 64 shots, we preprocessed progressive 720×480 video frames, originally captured on DV with 4:1:1 chroma compression. Of the 64 shots attempted, 35 shots resulted in complete and acceptable storyboards. The remaining 29 were not completed satisfactorily for the following reasons (some shots had multiple problems). The particle video algorithm failed significantly on seventeen shots: Eleven due to motion blur, three due to large-scale occlusions by foreground objects, and three due to moving objects too small to be properly resolved. Of the remaining twelve shots, the grouping algorithm failed to converge properly for five shots. Six shots included some kind of turning or rotating object, but our system only effectively annotates translating objects. Nine shots were successfully pre-processed, but would require additional types of annotations to be effectively represented using storyboard notation. Although additional work is needed to expand our system's operating range, these results show promise for our approach.

5.1 Limitations

One important limitation of our system is the length of time required to preprocess video. In our current implementation, the preprocess takes up to 5 minutes per frame for 720×480 input video, which is prohibitive for some of the potential applications described here. Although most of the preprocess is highly parallelizable, novel algorithms would be necessary for applications requiring "instant replay."

Many of the constraints on our method's operating range are inherited from the constraints on the underlying particle video method. This approach works best on sequences with large textured objects moving relatively slowly. Small moving objects are hard to resolve, and fast motion introduces motion blur, causing particles to slip across occlusion boundaries. Although the particle video algorithm is relatively robust to small featureless regions, it can hallucinate coherent motion in large featureless regions, which may be interpreted as separate groups in the motion grouping stage.

Another drawback is that when a video features objects with repetitive or small screen-space motions — like a subject moving back and forth along a single path, or moving directly toward the camera — it may be hard or impossible to reach a desired frame using the cost function described in Equation 9. Other cost functions have been proposed to infer the user's intent in such cases [12, 7, 11].

6 DISCUSSION

We have presented a system for interactively associating graphical annotations to independently moving video objects, navigating through video using the screen-space motion of objects in the scene, and composing new still frames from video input using a drag-and-drop metaphor. Our contributions include the application of an automated preprocess for video interaction, a fluid interface for creating graphical annotations that transform along with associated video objects, and novel interaction techniques for scrubbing through video and recomposing frames of video. The assembly of a well-integrated system enabling new approaches to video markup and interaction is, in itself, an important contribution. Our system performs all tracking and grouping offline before the user begins interaction, and our user interface hides the complexity of the algorithms, freeing the user to think in terms of high-level goals such as the placements of objects and the types and contents of their annotations, rather than low-level details of tracking and segmentation.

We believe our preprocessing method is uniquely well-suited to our applications. In particular, the long-range stability of the particle video tracks simplifies the motion grouping algorithm with respect to competing techniques: We had initially implemented the feature-based approach described by Sivic *et al.* [26], but encountered several important drawbacks of their approach for our interaction methods. First, the field of tracked and grouped points is relatively sparse, especially in featureless areas of the image. This scarcity of features makes them less suitable for the types of interactive applications that we demonstrate. Second, affine-covariant feature regions are sometimes quite large, and may therefore overlap multiple moving objects. (Both of these drawbacks also are problematic in the method of Dragicevic *et al.* [7].) However, a drawback of our grouping method is that it does require the number of motion groups to be specified *a priori*.

Although they work remarkably well, we do not consider our tracking and grouping algorithms to be a central contribution of this work. However, we find it notable that so much interaction is enabled by such a straightforward preprocess. We have no doubt that future developments in computer vision will improve upon our results, and we hope that researchers will consider user interfaces such as ours to be an important new motivation for such algorithms.

In conclusion, we believe interactive video object manipulation can become an important tool for augmenting video as an informational and interactive medium, and we hope this research has advanced us several steps closer to that goal.

Acknowledgments
The authors thank Peter Sand for the use of his source code, Sameer Agarwal and Josef Sivic for discussions about motion grouping, Nick Irving at UW Intercollegiate Athletics for sample video footage, and Harshdeep Singh, Samreen Dhillon, Kevin Chiu, Mira Dontcheva and Sameer Agarwal for additional technical assistance. Special thanks to Jon Goldman for the use of footage from his short film *Kind of a Blur* [9]. Amy Helmuth, Jay Howard, and Lauren Mayo appear in our video. Funding for this research was provided by NSF grant EIA-0321235, the University of Washington Animation Research Labs, the Washington Research Foundation, Adobe, and Microsoft.

References

1. A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23(4):294–301, 2004.
2. A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz. Keyframe-based tracking for rotoscoping and animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23(3):584–591, 2004.
3. ASTAR Learning Systems. `http://www.astarls.com`, 2006. [Online; accessed 5-January-2008].
4. C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Proc. SIGGRAPH 97*, pages 353–360, 1997.
5. Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski. Video matting of complex scenes. *ACM Trans. Graph.*, 21(3):243–248, 2002.
6. J. Dakss, S. Agamanolis, E. Chalom, and V. M. Bove Jr. Hyperlinked video. In *Proc. SPIE*, volume 3528, pages 2–10, 1999.
7. P. Dragicevic, G. Ramos, J. Bibliowicz, D. Nowrouzezahrai, R. Balakrishnan, and K. Singh. Video browsing by direct manipulation. In *CHI*, pages 237–246, 2008.
8. D. B Goldman, B. Curless, S. M. Seitz, and D. Salesin. Schematic storyboarding for video visualization and editing. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 25(3):862–871, 2006.
9. J. Goldman. Kind of a Blur. `http://phobos.apple.com/WebObjects/MZStore.woa/wa/viewMovie?id=197994758`, 2005. [Short film available online; accessed 5-January-2008].
10. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*, page 109. Cambridge University Press, ISBN: 0521540518, second edition, 2004.
11. T. Karrer, M. Weiss, E. Lee, and J. Borchers. DRAGON: A direct manipulation interface for frame-accurate in-scene video navigation. In *CHI*, pages 247–250, 2008.
12. D. Kimber, T. Dunnigan, A. Girgensohn, F. Shipman, T. Turner, and T. Yang. Trailblazing: Video playback control by direct object manipulation. In *ICME*, pages 1015–1018, 2007.
13. D. Kurlander, T. Skelly, and D. Salesin. Comic chat. In *SIGGRAPH '96*, pages 225–236, 1996.
14. Y. Li, J. Sun, and H.-Y. Shum. Video object cut and paste. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24(3):595–600, 2005.
15. C. Morningstar and R. F. Farmer. The lessons of Lucasfilm's Habitat. In M. Benedikt, editor, *Cyberspace: First Steps*, pages 273–301. MIT Press, Cambridge, MA, 1991.
16. Y. Pritch, A. Rav-Acha, A. Gutman, and S. Peleg. Webcam synopsis: Peeking around the world. In *Proc. ICCV*, pages 1–8, 2007.
17. Y. Pritch, A. Rav-Acha, and S. Peleg. Non-chronological video synopsis and indexing. *IEEE Trans. PAMI*, 2008. (to appear).
18. G. Ramos and R. Balakrishnan. Fluid interaction techniques for the control and annotation of digital video. In *Proc. UIST '03*, pages 105–114, 2003.
19. A. Rav-Acha, Y. Pritch, D. Lischinski, and S. Peleg. Dynamosaicing: Video mosaics with non-chronological time. In *Proc. CVPR*, pages 58–65, 2005.
20. A. Rav-Acha, Y. Pritch, and S. Peleg. Making a long video short: Dynamic video synopsis. In *Proc. CVPR*, pages 435–441, 2006.
21. E. Rosten, G. Reitmayr, and T. Drummond. Real-time video annotations for augmented reality. In *Proc. Intl. Symp. on Visual Computing*, 2005.
22. P. Sand. *Long-Range Video Motion Estimation using Point Trajectories*. PhD thesis, MIT, 2006.
23. P. Sand and S. Teller. Particle video: Long-range motion estimation using point trajectories. In *Proc. CVPR '06*, pages 2195–2202, 2006.
24. A. Schödl and I. A. Essa. Controlled animation of video sprites. In *Proc. ACM/Eurographics Symp. on Comp. Animation*, pages 121–127, 2002.
25. A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *SIGGRAPH '00*, pages 489–498, 2000.
26. J. Sivic, F. Schaffalitzky, and A. Zisserman. Object level grouping for video shots. *Intl. J. of Comp. Vis.*, 67(2):189–210, 2006.
27. J. M. Smith, D. Stotts, and S.-U. Kum. An orthogonal taxonomy for hyperlink anchor generation in video streams using OvalTine. In *Proc. ACM Conf. on Hypertext and Hypermedia*, pages 11–18, 2000.
28. Sportvision. Changing The Game. `http://www.sportvision.com`, 2006. [Online; accessed 5-January-2008].
29. J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24(3):585–594, 2005.
30. Wikipedia. Telestrator. `http://en.wikipedia.org/w/index.php?title=Telestrator&oldid=180785495`, 2006. [Online; accessed 5-January-2008].
31. A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):13, December 2006.
32. L. Zhang, N. Snavely, B. Curless, and S. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 23(3):548–558, 2004.