TRAILBLAZING: VIDEO PLAYBACK CONTROL BY DIRECT OBJECT MANIPULATION

Don Kimber[1], Tony Dunnigan[1], Andreas Girgensohn[1],
Frank Shipman[2], Thea Turner[1], Tao Yang[1,3]

[1] FX Palo Alto Laboratory, Palo Alto, CA, 94304, USA
[2] Texas A&M University, College Station, TX 77843-3112, USA
[3] Northwestern Polytechnical University, Xi'an, 710072, China
lastname@fxpal.com shipman@csdl.tamu.edu yangtaonwpu@163.com

ABSTRACT

We describe a new interaction technique that allows users to control nonlinear video playback by directly manipulating objects seen in the video. This interaction technique is similar to video "scrubbing" where the user adjusts the playback time by moving the mouse along a slider. Our approach is superior to variable-scale scrubbing in that the user can concentrate on interesting objects and does not have to guess how long the objects will stay in view. Our method relies on a video tracking system that tracks objects in fixed cameras, maps them into 3D space, and handles hand-offs between cameras. In addition to dragging objects visible in video windows, users may also drag iconic object representations on a floor plan. In that case, the best video views are selected for the dragged objects.

1. INTRODUCTION

An important affordance of digital video is that it supports nonlinear viewing in whatever manner is most suitable to a given task. Particularly for purposes such as process analysis, sports analysis or forensic surveillance tasks, some portions of the video may be skimmed over quickly while other portions are viewed repeatedly many times, at various speeds, playing forward and backward in time. Scrubbing, a method of controlling the video frame time by mouse motion along a time line or slider, is often used for this fine level control, allowing a user to carefully position the video at a point where objects or people in the video are in certain positions of interest or moving in a particular way.

Scrubbing and off-speed playback, such as slow motion or fast forward, are useful but have limitations. In particular, no one playback speed, or scale factor for mapping mouse motion to time changes, is appropriate for all tasks or all portions of video. Adding play speed controls and the ability to zoom in on all or some of the timeline helps, and some authors describe variable scale scrubbing where mouse motion orthogonal to the slider sets the scale [5]. But these features can be confusing and distracting. Instead of directly controlling what they view, users spend time focusing on control features of the interface. Some researchers have addressed this using variable speed playback with speed determined by the amount of motion or new information at each time of the video [7]. These schemes essentially re-index the video from time t to some function $s(t)$, where for example $s(t)$ is the cumulative information by time t, or the distance a tracked object moved. We have experimented with such schemes for variable scale scrubbing, where in addition to a time slider, the user is provided with another slider for s. Since there are various measures of change, or various objects to track, multiple sliders can be provided.

However, rather than indirect scrubbing control by multiple sliders, a more natural interface is based on the reach-through-the-screen metaphor proposed in [2] — simply let users directly grab and move objects along their trails. This may be done directly in a video window or in a floor plan view which schematically indicates positions of people by icons (see Fig. 1). For example, a user reviewing surveil-

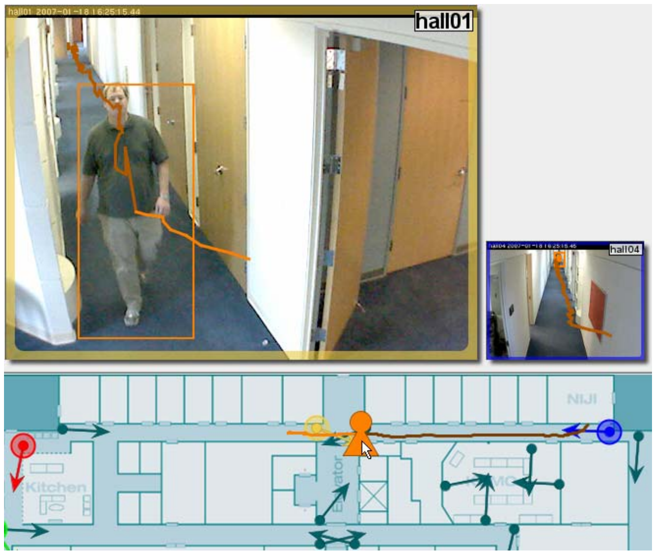

Fig. 1. Viewer showing trails in video and floor plan view. The user may scrub video by dragging a person along their path in the video window or on the floor plan.

lance video may drag a person along their trail to view video at any given point, or drag a parked car to move to the points in the video where the car was parked or left. The effect of the interface is to shift user experience from passively watching time-based video to directly interacting with it.

The interface mechanism described here is similar to Goldman *et al.*, where annotations for camera and subject motion on storyboards are used to control video playback [3]. They manually determine some processes that are automatic in ours, notably tracking and labeling. Their prototype interprets motion against multi-shot frame layouts while ours uses world geometry to enable track across cameras.

Next, we provide an overview of our system implementation. This is followed by a description of direct object manipulation in video displays or on floor plans and selection among candidate object trails. We conclude with a discussion of the advantages of our approach.

2. SYSTEM IMPLEMENTATION

The direct video manipulation methods described in this paper require metadata defining the trails of people and objects in the video. Additionally, manipulation of floor plan views and cross-camera manipulation requires association of objects across cameras as well as world coordinate trajectories. This metadata could be acquired by various means, but in our implementation we use a video system called DOTS (Dynamic Object Tracking System) developed at FXPAL [4]. The overall system architecture is shown in Fig. 2, and is comprised of video capture or import, video analysis, and user interface playback tools. The analysis algorithms are described in more detail in [9].

2.1. Single Camera Video Analysis

The essential requirement of the analysis is to produce object tracks suitable for indexing. At each frame time *t*, each tracked object is associated with an image region bounding box, which is entered into a database along with an object id.

The first processing step is segmenting people from the background. We use the Gaussian mixture model approach for pixel level background modeling (Stauffer *et al.* [8]). For foreground segmentation that is robust to shadows, similar colors, and illumination changes, a novel feature-level subtraction approach is used. First, we use the foreground density around each pixel to determine a candidate set of foreground pixels. Then, instead of comparing the difference in intensity value of each pixel, we find the difference from the neighborhood image using a normalized cross-correlation computed using the integral image method [6].

For single camera tracking in the data association module, we follow previous works [1] in the use of a correspondence matrix to classify an object's interactions with other objects into five classes: Appear, Disappear, Continue, Merge and Split. Identity maintenance is handled for occlusion by a track-based Bayesian segmentation algorithm using appearance features. Merges and Splits are entered into the database so that if tracked object regions are merged to form new regions, which are subsequently split to form other regions, it is possible to determine which future regions are descendents and, hence, candidates to be the tracked object.

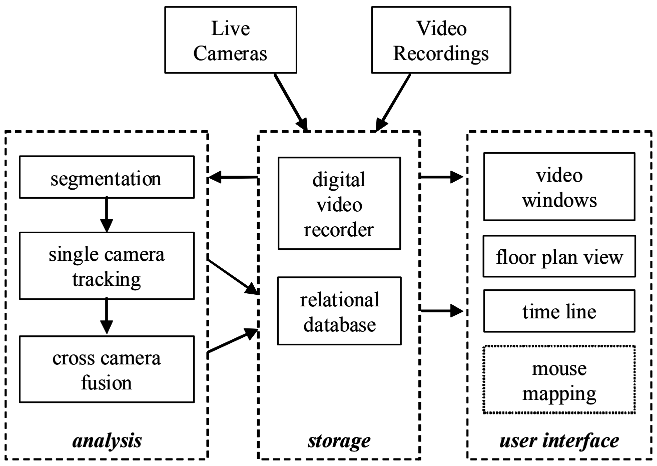

Fig. 2. System architecture.

2.2. Multiple Camera and Floor Plan-based Tracking

To support manipulation across different video views, or on a schematic floor plan, it is necessary to map object positions in video to world coordinates, and to determine associations of objects across cameras. In our system, cameras are mounted near the ceiling with oblique downward views. Estimates of world position are based on the assumption that the bottoms of the bounding boxes are from points on the floor plane. A model of building geometry is used to filter out nonsensical results, for example, where the resulting world coordinates would be occluded by walls.

Cross-camera object association is handled by searching for a hypothesis H of world object trajectories which maximizes the a posteriori probability P(H|O) of H given observations O, where O is the set of all tracked object regions in all cameras. This is equivalent to maximizing P(O|H)P(H) over H. The priors P(H) incorporate a Gauss-Markov object dynamics model and learned probabilities for object appearance and disappearance. P(O|H) is based on a Gaussian error model for an object at a given world position being tracked at a given image position. A result of the fusion is an estimate of world trajectories for tracked objects and an association of objects with tracked regions in images.

2.3. User Interface Components

We developed a multi-stream video player that combines video displays at different resolutions, a map indicating positions of cameras and tracked objects, and a timeline for synchronously controlling all video displays [4]. The system automatically selects video displays and enlarges more im-

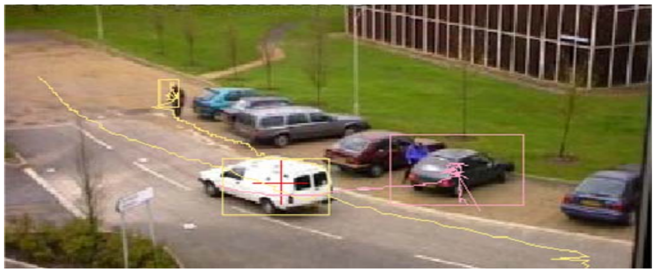

Fig. 3. Dragging objects with different speeds and times.

portant displays (e.g., those showing a tracked object better; see Fig. 1).

3. DIRECT MANIPULATION OF OBJECTS

Given the object trail information, whether in a single camera view or a floor plan view, we support the use of the mouse to move objects to different points on their trails. This controls the video playback position for all video views, such that the playback position is set to the time when the object occupied the position selected by the user.

In any view, clicking on an object selects the object and shows its trajectory. If the mouse click is not over an object, the system determines a set of candidate trajectories in the neighborhood of the click. Users may select a candidate by rapidly clicking multiple times to cycle through candidates. Once an object is selected, it may be dragged to different positions. The object motion is constrained by the object trail such that the object can only be moved to locations where it was observed at some time.

Picking the point in time where the object was closest to the mouse position can have undesirable consequences in situations where objects cross their own trail or where they loop back. In such cases, the method will create discontinuities in playback as the location closest to the pointer jumps between the different parts of the object's track. Additionally, when an object reaches the location where it starts to retrace its path, it is ambiguous whether dragging the pointer back indicates scrubbing forward or backward in time.

To solve these problems we use a distance function that includes weighted distances in location and time as well as a cost for changing temporal directions during a single dragging event. The following equation determines the distance for the object position p_o, the mouse position p_m, the object time t_o, and the video time t_v. The constant c_3 is added if the object time t_o would cause a reversal of playback direction. In response to a mouse movement to p_m, the video time is changed from t_v to the t_o that minimizes *d*.

$$d = c_1\left|\overrightarrow{p_o p_m}\right| + c_2\left|t_o - t_v\right| + \begin{cases} 0 \\ c_3 \end{cases}$$

One problem with this approach is that the user may have to drag the cursor fairly far to overcome the cost associated with changing time and/or direction of playback. To

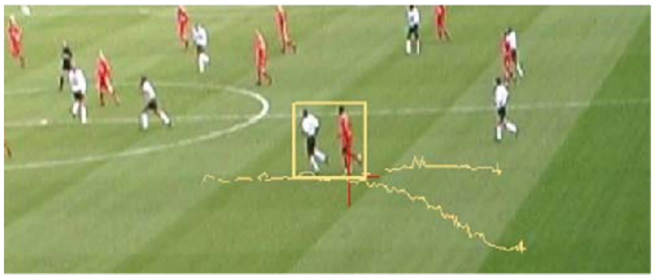

Fig. 4. Dragging regions which merge and split.

overcome this, we double the value of c_1 for every 0.5 seconds during which no new video frame has been displayed.

To indicate to the user which objects in the video may be dragged and which locations they may be dragged to, object trails may be shown in the views. We have found that, depending on the particular video and task, it is appropriate to show all trails for an extended period of time, trails for only objects visible at the current play time, trails for only an object currently being dragged, or for the object currently under the mouse. Also, for complex scenes, it may be desirable to show only a portion of trails for a fixed time into the future or past. These settings are configurable and may be temporarily overridden by key presses.

3.1. Direct Manipulation in the Video Window

The trailblazing interface method may be used in video and floor plan views. However, for some applications, the floor plan views may not be available either because camera calibrations are unavailable, or because the scenes are too complex for robust tracking analysis with cross-camera fusion. The method may still be applied to the video windows in those cases, however.

The method is particularly useful when different objects move with widely varying timescales are at different times. The scene in Fig. 3 (from the PETS 2000 test set) includes a pedestrian, a moving car, and a parked car. A user may drag the pedestrian to any position along the trail, and will see the white car move quickly. They also may drag the parked car to move back the time it was parked or ahead to the time it moves from its spot.

For complex scenes where tracking cannot robustly maintain object identity or in which occlusion causes regions to be merged and split, the method may still be used effectively by using the ancestry chain. For example, Fig. 4 shows a football scene where two players have moved close to each other and the tracking algorithm has merged their regions. Putting the mouse over the players shows the merged tracking region and the path for a few seconds into the future and past. The user may drag back along the path and when the mouse is moved to a position along the trail of either of the merged players, the video will move to the time the player is at the desired location.

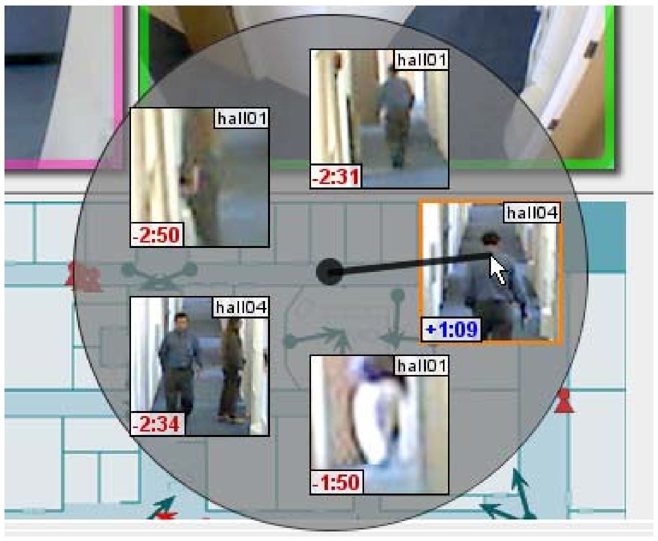

Fig. 5. Spatially and temporally close candidates.

3.2. Direct Manipulation in the Floor Plan

Dragging objects on a map based on their movement in location is another natural mode of interacting with video. As an object is placed in different positions along its trail, the playback position of all video displays is skipped to the corresponding time. Among all available video views, the ones that are displayed show the selected object best. The best video view is shown in a larger size. As the object is dragged, the video views are replaced (see Fig. 1).

The floor plan ties together multiple video displays. In addition to dragging an object within the floor plan or a single video display, the object may also be dragged between video displays or from the floor plan to a video display. The system interprets this as a request to locate a time where the object is visible in the destination video display at the final position of the dragging action.

3.3. Selection among Candidate Trails

It is often desirable to determine when an object reached a particular point or to see all objects that were near that point. In addition to dragging the selected object, the user may also click on a position not occupied by an object. That will move the selected object close to the selected position. If no object was selected, one of the objects that were near that point will be selected. Repeatedly clicking on the same position will cycle through candidate times and objects.

Right-clicking on a position displays a radial menu presenting the candidates (see Fig. 5). For each candidate, the video image is cropped to the outline of the object to provide a good view. After selecting one of the candidates, the corresponding object is selected and the video display skips to the corresponding time.

4. CONCLUSIONS

We presented a system that allows users to control video playback by directly manipulating objects shown in the video. This interaction technique has several advantages over scrubbing through video by dragging a slider on a time line. First, because tracked objects can move at different speeds (e.g., pedestrians and cars), it is more natural for the user to manipulate the object directly instead of a slider. Second, a slider does not provide a sufficiently fine control for long videos. Third, the start and end of an interval of interest where an object is visible is apparent to the user. Finally, our technique can also be used as a means for retrieval (e.g., check when a person was in a particular position or find all people who were near that position).

While our system relies on tracking, it deals with merging and splitting of objects through chains of ancestors for tracked objects. Our technique is in use in a research video surveillance system with more than 20 cameras. Currently, we are working on extensions of the techniques presented here to allow users to correct tracker errors.

5. REFERENCES

[1] R. Cucchiara, C. Grana, and G. Tardini. Track-based and object-based occlusion for people tracking refinement in indoor surveillance. *Proc. ACM 2nd International Workshop on Video Surveillance & Sensor Networks*, pp. 81-87, 2004.

[2] J. Foote, Q. Liu, D. Kimber, P. Chiu and F. Zhao. Reach-through-the-screen: A New Metaphor for Remote Collaboration. *Advances in Multimedia Information Processing – PCM 2004,* pp. 73-80, Springer, Berlin, 2004.

[3] D. Goldman, B. Curless, S. Seitz, and D. Salesin, Schematic Storyboarding for Video Visualization and Editing. *Proceedings of ACM SIGGRAPH* '06, pp. 862-871. New York, NY, 2006.

[4] A. Girgensohn, F. Shipman, T. Dunnigan, T. Turner, and L. Wilcox. Support for Effective Use of Multiple Video Streams in Security. *Proc. of the fourth ACM International Workshop on Video Surveillance & Sensor Networks,* Santa Barbara, CA, 2006.

[5] W. Hürst, G. Götz, P. Jarvers, Advanced User Interfaces for Dynamic Video Browsing, *Proceedings of ACM Multimedia 2004.*

[6] J.P. Lewis. Fast normalized cross-correlation. *In Vision Interface*, 1995.

[7] K. Peker and A. Divakaran, Adaptive fast playback-based video skimming using a compressed-domain visual complexity measure. *ICME 2004, Taipei, Taiwan*, pp. 2055-2058*, June 2004*,.

[8] C. Stauffer, W. Eric, and L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Volume 22, Issue 8, pp. 747-757, 2000.

[9] T. Yang, F. Chen, D. Kimber, and J. Vaughan. Robust People Detection and Tracking in a Multi-Camera Indoor Visual Surveillance system. *ICME 2007, Beijing, China, July 2007.*