Real-time Direct Manipulation of Screen-based Videos

Laurent Denoue, Scott Carter, Matthew Cooper, John Adcock
FX Palo Alto Laboratory, Inc.
3174 Porter Drive, Palo Alto, CA 94304 USA
{denoue, carter, cooper, adcock}@fxpal.com

ABSTRACT
We describe direct video manipulation interactions applied to screen-based tutorials. In addition to using the video timeline, users of our system can quickly navigate into the video by mouse-wheel, double click over a rectangular region to zoom in and out, or drag a box over the video canvas to select text and scrub the video until the end of a text line even if not shown in the current frame. We describe the video processing techniques developed to implement these direct video manipulation techniques, and show how they are implemented to run in most modern web browsers using HTML5's CANVAS and Javascript.

Author Keywords
direct manipulation; video processing; intelligent user interface; real-time; web browser; html5.

ACM Classification Keywords
H.5.2. User Interfaces

INTRODUCTION
Screen-based tutorial videos can help teach people how to program, how to use new software or web site, as well as how to perform a variety of other tasks. These videos can be screencasts (digital recordings of computer screens), recordings from specialized markup applications (e.g., Khan Academy videos), or even screen content extracted from live talks (e.g., using automated video analysis techniques [3]). Unfortunately, the standard video timeline is not optimal to navigate these types of video. Following work from [1], we describe direct video manipulation techniques that make it easy for users to watch and navigate such screen-based videos. Our techniques are purely image-based and do not require hooks into the operating system.

Our work in this paper focuses on three types of screen-based video interactions: detecting scrolling regions in the video so that users can manipulate them directly using a mouse-wheel; detecting text areas so that users can directly select text from the video canvas and paste it into a note-taking application; and detecting regions-of-interest in the video so that users can zoom into an important part of the video frame. Examples include zooming a dialog box to better see what is typed inside, and perhaps copying and pasting this part of the video frame as an image for later review or note taking.

IUI'13 Companion, March 19–22, 2013, Santa Monica, CA, USA.
ACM 978-1-4503-1966-9/13/03.

DIRECT VIDEO CONTENT MANIPULATION
Our system implements these interaction techniques using a client-side approach that runs on most modern browsers using HTML5 video, CANVAS and JavaScript to perform real-time video processing. In order to process video frames from a VIDEO tag inside our web page, the video is served through our proxy server that streams WebM or MP4 video from YouTube, solving the same-domain policy constraint.

A timer is used to continuously draw video frames onto a CANVAS element that the user sees. We bind mouse listeners to receive mouse-wheel, double-click, and mouse down, move and up events. To speed up image processing in JavaScript, we process a scaled down version of the video frames, currently 50%.

Image Processing for Region Detection
The system needs to detect rectangular regions, scrolling areas and text regions. We first binarize the incoming frame using a simplified edge detector that reports 1 if the grayscale difference between two horizontally adjacent pixels is greater than 32 and 0 otherwise. This filter detects vertical edges in the image. We apply a similar filter on vertical pixels to detect horizontal edges. Edges are found by counting lines and columns where 1 is found more than 40 times in a row or column.

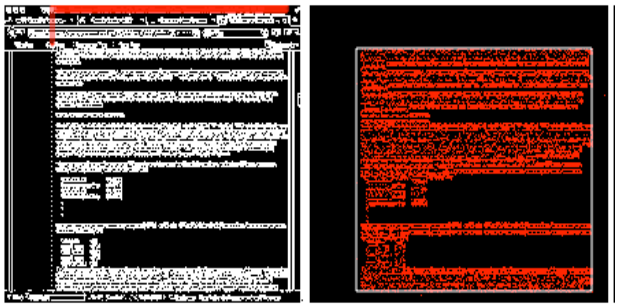

Figure 1. Left: binarized frame and red bars (top) showing the delta values computed with the previous frame. Right: image difference showing the bounding box (in white) used to restrict the region where the system computes the delta values for each column.

When the user double clicks on the main canvas, the system looks for two vertical lines on the left and right of the click, and then two horizontal lines that form a rectangle with the given vertical lines. If a rectangle is found, the system starts a zooming animation that stretches the main CANVAS element by modifying its size and offset left and top, thereby giving the illusion that the video is being zoomed

in. This step very fast as it is handled by the browser DOM and backed up by the graphics processing unit (GPU).

Image Processing for Content Scroll Detection

In order to detect scrollable regions, we compute the difference between the previous and current frame, both binarized in the first step. Vertical and horizontal projection profiles are computed to find the changed region (shown in white in Figure 1, right). We then compute the best correlation between vertically shifted pairs of pixel columns within this region, taken from the previous and current frame. If most columns agree on a similar shift, we use the average as the scroll value, and store the frame in an array along with the current video playback time and scroll value.

When the user mouse-wheels over the video canvas within 2 seconds after the last scroll was detected, we compute the cumulative scroll value corresponding to the drawn frame and pick the frame in the array that corresponds to a cumulative scroll less or greater than the current cumulative scroll, depending on the direction of the mouse-wheel. It appears that the video content magically scrolls up or down.

Figure 2. Connected components and their bounding boxes are used to draw highlighted rectangles over the main video canvas, giving users the illusion that they are selecting text from the video canvas.

Image Processing for Text Selection

When the user clicks over the main canvas, the system a) binarizes the current video frame; b) computes the connected components and their bounding boxes using [2]; c) defines the initial selection box rectangle as {x,y,x+1,y+1} and visually highlights the boxes under this rectangle, giving the user the illusion of having selected this text. When the user expands her selection box, the system updates the corresponding boxes underneath, as shown in Figure 2.

All the while, the video plays in the background and the system accumulates incoming frames that the user does not see. To her, it looks as if the video is paused. However, if she extends her selection past a region that does not include a connected component, the system looks into the accumulated frames to find one that contains a connected component under this new area. If found, it draws the frame over the main canvas, giving the illusion that the video advanced to this point in time in order to reveal more text on that line.

When the user lifts the mouse pointer, the system copies the selected fragment as an image into a new hidden CANVAS element and generates a PNG image representation. Currently, this image can be pasted as is into a word processor, but could also be sent to an optical character recognition engine to extract its text content (e.g., [3]).

CONCLUSION

We described techniques to directly manipulate video content that allow users to 1) easily scroll up and down over a video to automatically rewind or fast-forward, 2) easily select text content from video frames, including automatic skipping of frames until the desired text segment is shown, and 3) automatic region detection to zoom into interesting areas of the video, such as rectangular areas that can correspond to common widgets such as windows and dialog boxes. The system performs these video processing tasks in real-time inside a modern web browser using HTML5 video, CANVAS element and Javascript. We are working on overlaying cues to let users know they can scroll or select text by showing faded out versions of future frames. Other extensions could include tracking windows being moved over the video and navigating the video timeline accordingly, or listening to keyboard events such as backspace or cursor arrows to rewind or fast-forward the video to corresponding text elements.

REFERENCES

1. Dragicevic, P., Ramos, G., Bibliowitcz, J., Nowrouzezahrai, D., Balakrishnan, R., and Singh, K. 2008. Video Browsing by Direct Manipulation. In Proceedings of ACM CHI 2008. 237-246.
2. Fu Chang and Chun-jen Chen and Chi-jen Lu. A linear-time component-labeling algorithm using contour tracing technique. Computer Vision and Image Understanding, 93(2). 2004. 206-220.
3. Adcock, J., Cooper, M., Denoue, L., Pirsiavash, H., and Rowe, L.A. TalkMiner: A Lecture Webcast Search Engine. In Proceedings of ACM Multimedia 2010. 241-250.