Making Software Tutorial Video Responsive

Cuong Nguyen
Portland State University
Portland, OR 97207-0751
cuong3@pdx.edu

Feng Liu
Portland State University
Portland, OR 97207-0751
fliu@cs.pdx.edu

ABSTRACT

Tutorial videos are widely available to help people use software. These videos, however, are viewed by users as captured and offer little direct interaction between users and software. This paper presents a video navigation method that allows users to interact with software tutorial video as if they were using the software. To make the tutorial video responsive, our method records the user interaction events like mouse click and drag during capturing the video. Our method then analyzes, selects, and visualizes these user interaction events at the event locations. When a user directly interacts with an event visualization, our method automatically navigates to the proper video frame to provide the visual feedback as if the software were responding to the user input. Thus, our method provides the experience of "interacting" with the software through directly manipulating the tutorial video. Our study shows our method can better help users follow tutorial videos to complete tasks than the baseline timeline interface.

Author Keywords

Video Navigation and Browsing; Software Tutorial Video

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces

INTRODUCTION

Tutorial videos are widely employed to help people use software. These tutorial videos are often produced by screen capturing the software-use workflow. When watching a software tutorial video, users often need to frequently navigate to relevant frames to recall parameters, check results, or skip certain steps [12]. These navigation goals are important for users to understand the operations and perform them using the software; however, the classic timeline-based players do not suit these navigation goals very well. Using the timeline for video navigation, users need to focus on two different locations: the timeline and the video content simultaneously. It is often slow to locate frames of interest. More importantly, the timeline interface does not provide direct interaction between users and software. It does not support users to directly work with the software through the software interface and does not provide the visual feedback to the user input from the software. That is, the tutorial video is not responsive with the timeline interface. These problems can compromise the user's performance in learning to use the software from the tutorial video.

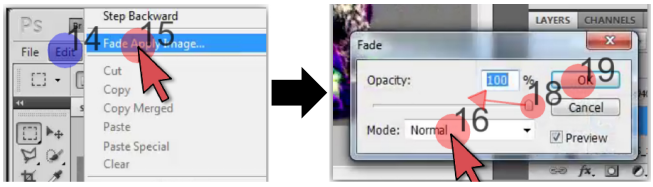


Figure 1. Responsive tutorial video navigation. When a user clicks Event 15, our system navigates to the proper frame that shows a dialog to respond to the user input.

This paper presents a method that makes a software tutorial video responsive to the user input and allows users to navigate the tutorial video as if they were using the software[1]. To make the tutorial video responsive, our method records the user interaction with the software when screen-capturing the tutorial author's software-use process. Our method then analyzes these interaction events and selects the relevant ones at the current frame. These events are finally visualized close to the event locations, such as buttons and menu items. Users can navigate the tutorial video by directly interacting with the event visualization as if they were interacting with the software. Figure 1 shows an example of a user navigating a *Photoshop* tutorial video. In the current frame, our system displays the relevant events 14 and 15. Event 15 indicates that the tutorial author clicked the menu item *Fade Apply Image*. When watching this tutorial video, a user clicks Event 15 and our system automatically navigates to the frame that provides the visual feedback to the user input as if *Photoshop* is responding to the user input. Here the *Fade* dialog window appears after Event 15 is clicked which is exactly what happened after the tutorial author clicked *Fade Apply Image*.

This responsive tutorial video navigation method has advantages over timeline methods in that it merges two processes together: navigating a software tutorial video and using the software. Moreover, this method allows users to quickly reach frames of interest by directly "interacting" with the software through interacting with the event visualization. Our study shows that our method can better help users follow tutorial videos to complete tasks than timeline interfaces.

RELATED WORK

The timeline interface has been enhanced with event visualization to better support software tutorial video navigation. Some methods detect and display tool invocation events on timeline [1, 11, 12] and can help users quickly find frames of

[1] `http://graphics.cs.pdx.edu/project/tutorDMVN`

CHI 2015, April 18–23, 2015, Seoul, Republic of Korea.

http://dx.doi.org/10.1145/2702123.2702209

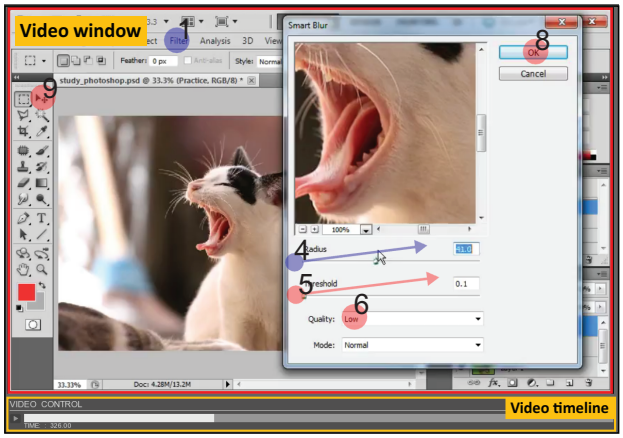


Figure 2. System overview.

interest based on the tool usage. *Chronicle* captures software workflow histories and visualizes them on timeline to support workflow exploration in a tutorial video [7]. *Toolscape* explores crowdsourcing to display step-by-step instructions on timeline [9]. Instead of enhancing timeline, our work visualizes user interaction events right in the video window where these events actually happen and allows users to navigate the tutorial video through these in-context event visualization as if they were directly interacting with the software.

Direct manipulation video navigation, which allows users to navigate a video by dragging an object of interest along its motion trajectory [5, 6, 8], has been extended to assist software tutorial video navigation. Denoue *et al.* developed a text-based tutorial video navigation method that enables users to directly select text content and scroll text in the video as if they were using a text viewer [4]. The *Waken* video player [1] displays tooltips and menus when a user hovers on corresponding tool icons in the video and allows the user to directly click on an icon to search for moments in the video when that tool was used in a separated side panel. Our work is inspired by these methods and further extends direct manipulation video navigation to develop a responsive software tutorial video navigation method. Our method automatically navigates to the proper frame to provide the visual feedback as if the software were responding to the user input, providing a navigation experience of directly using the software.

RESPONSIVE SOFTWARE TUTORIAL VIDEO

Our software tutorial video navigation method consists of two components: *event processing* that captures, analyzes, and visualizes user interaction events in the tutorial video and *user interface* that supports users to directly interact with the visualized events as if they were using the software.

Figure 2 shows an overview of our system. Our system has a video window and a timeline. Unlike timeline video players, our system depicts a selected set of user interaction events like mouse click and drag as visual annotations and overlays them on their corresponding software interface elements, such as buttons and menu items. These events are color-coded and labeled to convey the temporal order. The set of selected events are dynamically adjusted to provide enough information of the tutorial video around the current frame while not cluttering the screen and overwhelming users. The event annotations help users understand the operations performed and the order in which they happen. Users can navigate the video to events of interest by directly interacting with their annotations in the video window. When users click or drag on a software interface element hinted by an event, our system automatically navigates to the proper frame as if the software were responding. For example, clicking the *File* menu on top of the screen will show a drop down menu, or dragging along the arrow will gradually show the brush motion. Note, while our system visualizes the event order, users can click and navigate to any event without necessarily respecting the order. The capability of non-linear navigation provided by our system allows users to quickly locate frames of interest.

Event Processing

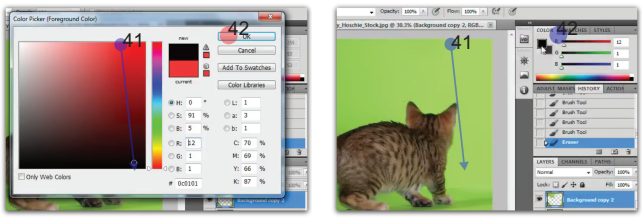
Our system records the events of user interaction with the software when screen-capturing the tutorial author's software-use process. Our system currently focuses on mouse activities, namely click and drag. Click events are often used to invoke a tool or select an object and drag events are often used for complex operations like drawing or adjusting parameters. Each extracted event contains the event label (click or drag) and its corresponding trigger location (x, y) and time t. We find that these two interaction events can already capture a range of user interactions with software. More interaction events will be included in future. These events can be gathered using publicly available APIs [3] or through computer vision analysis of the captured tutorial video [1].

Event Analysis and Selection

A software tutorial video often carries a large number of user interaction events. At each frame, visualizing all the events will clutter the video content and overwhelm users. Therefore, our system only selects relevant events to visualize.

Temporal event culling. In software tutorial videos, users often do not need to see all the interaction events in the video at once to perform learning tasks. They often watch a tutorial video for a few seconds, pause, and then switch to the software to perform the operations they just watched [12]. Thus, the interaction events in the local temporal interval around the current frame often provide users enough learning information since they are directly related to the software-using task that users will perform using the software. Therefore, at any video frame, we limit the number of events that will be finally visualized to be 10 in the time interval around the current frame. This time interval is dynamically adjusted to always include at most 10 events at any time. We empirically find that 10 events can already provide users with enough cues for learning the use of the software, while avoiding cluttering the video window and overwhelming users.

Relevant event selection. At each frame, our system displays the current events as well as some past and future events. In this way, the visual annotations of some events will overlay software interface elements that are invisible at the current frame. As illustrated in Figure 3 (b), Event 41 and 42 in the dialog box are still visible after the dialog box disappears later on. These events appear *out of context* and can confuse users. Our system displays an event annotation only if its context is visible. The event context is defined as the visual content around the event location when it is triggered, such as the



(a) Frame 2632 (b) Frame 2657

Figure 3. Relevant event selection. Event 41 and 42 are in context in (a), but are out of context in (b) where the dialog window disappears.

software interface elements and the image that a user draws on. This task cannot be simply achieved by checking whether the event is triggered at the current frame or not as some event context lasts for an extended amount of time.

We use a visual template matching algorithm to determine if an event context is visible or not. Specifically, we take the pixel patch centered at the event location in the frame when the event is triggered and compare it with the corresponding patch centered at the same location in the current frame. The pixel patch for a click event is a 20×20 window centered at the event location. The pixel patch for a drag event is the bounding box around the dragging path. We use the template matching algorithm in *OpenCV* to compute the similarity between two pixel patches. If the matching score is no less than 0.6, we consider the event context is visible.

In software tutorial videos, some interface elements may change states or appearances when being clicked or dragged. This may cause problems if we only compare the pixel patch in the current frame with that in the event-triggered frame. To address this problem, we also consider frames that contain other states of the event such as idle, hover, and pressed. For each event, we take the event-triggered frame as the pressed-state frame. We search back in the video to find its corresponding hover-state and idle-state frames. These frames are also used for template matching. The maximum matching score between any of these frames and the current frame is used to determine the visibility of an event annotation.

Data adjustment. Sometimes, consecutive mouse activities may occur at nearly the same location [10]. When these events are visualized, their visual annotations will overlap with each other. Our system detects these spatially co-located events and offset their coordinates so that their annotations are separated horizontally by at least 5 pixels. Multiple events can also be triggered in a short time, but at different locations, we merge these events as a single drag event to simplify the visualization and make navigation of these events easier.

Event Visualization

We use graphical annotations to visualize the selected user interaction events. The annotations provide users with navigation cues to quickly locate frames of interest and help them understand the operations performed in the tutorial video. Figure 2 shows the annotations of the click and drag events in our system. Similar to DemoWiz [3], we annotate a click event as a circle centered at the event location and a drag event as a straight arrow that connects the starting and ending point of the path. The start of the arrow is annotated as a circle and the end is annotated as an arrow-head. For the drag event, we uniformly map the frames along the arrow so that users can drag along the arrow to mimic the drag operation in the tutorial video and obtain the corresponding instantaneous feedback, as detailed later. We choose not to visualize the drag event as the exact dragging path to simplify the annotation for both clear visualization and easy navigation. As reported in direct manipulation video navigation research [2], a complex path needs to be simplified for users to drag easily.

To convey the event order, we label each event with a unique number and render it next to the event. We also color-code each event such that the past and future events are rendered in blue and red, respectively.

User Interaction

Our system supports users to directly interact with the event annotations to navigate a software tutorial video as if they were interacting with the software. This is achieved by replicating the software behavior by automatically navigating to a target video frame where the software responds to the user action. Such a target frame is available in the tutorial video after an event is triggered.

We pre-process the event data to find the target frame of each event. For each event, we measure the difference between the event-triggered frame and the following frames. We compute the difference between two frames by counting the number of pixels that are different between two frames. We select the target frame as the frame with the highest difference value within the next 10 frames after the event was triggered. This target frame shows the software's response to the event. For example, the target frame of a right click event on the canvas in *Photoshop* brings in a drop down menu.

After a user clicks an event annotation in the video window, our system calculates and selects the event with the closest spatio-temporal distance to the click location. Then, the video is navigated to the associated target frame of that event, creating an illusion that the software is responding to the user interaction. For example, when watching a tutorial video showing the use of the Gaussian Blur tool in *Photoshop*, the user may want to locate the frame where the tool's dialog box is presented to recall its settings. Instead of scrubbing the timeline in a video player, the user can quickly locate the frame with the dialog box by invoking the same set of operations as in *Photoshop*. That is, the user clicks on the *Filter* menu, then clicks *Blur*, and finally invokes the dialog box by clicking *Gaussian Blur*. Each step is fast and intuitive because the user is guided by the annotated events and can easily click these events to navigate to the desired video segment.

Our system also supports direct scrubbing along the arrow annotation of a drag event to allow a user to mimic the dragging user input. When a user is dragging along the arrow, our system will automatically navigate to the following frames to provide the instantaneous feedback.

EVALUATION

We conducted a preliminary user study in our lab to evaluate how our responsive video navigation method supports users to navigate tutorial videos. We compared our method with a classic timeline interface (*Timeline*).

We asked participants to learn from software tutorial videos to perform two tasks, one using *Excel* and the other using *Photoshop*. We captured and narrated two videos for our study. The *Excel* video (1.85 minutes) shows how to create a column chart, correct its entries, and relocate the chart legends. The *Photoshop* video (1.83 minutes) shows how to apply the *Emboss Filter*, *Apply Image*, and *Fade* tools on an image. The video frame size is 860×546 pixels. We used a desktop PC with two 23-inch monitors, a speaker, a standard mouse and keyboard. Similar to a previous study [12], we displayed the video player in one monitor and the actual software in another one to minimize the effects of the limited screen space.

We used a between-subjects design to avoid the strong learning effect from having a participant perform the same task aided by two methods. 12 users from the university campus participated in our study. Their majors include Computer Science, Accounting, Arts, etc. Their ages range from 18 to 28 years. These participants were randomly assigned to one of the two methods, resulting in 6 participants per method.

Before the study, we showed the final results of the tasks and asked participants if they could produce exactly the same results using the software. All participants responded that they were not sure how to perform the tasks and would need the tutorial videos although about half of them are familiar with *Photoshop* and *Excel*. They were trained to navigate tutorial videos using the method that they would use. The training session used a practice video that lasts 46 seconds and shows how to use some tools in *Photoshop*. These tools are different than those used in the *Photoshop* tutorial video in the formal study. Participants then proceeded to perform the *Excel* and *Photoshop* task. They were told to complete the task with the software as fast and as accurate as possible. Participants click the start and end buttons to record the task completion time.

Results. As reported in Table 1, participants were faster using our method to complete both tasks than *Timeline*. We analyzed the task completion time with an independent-samples t-test. The difference between our method and *Timeline* in the *Excel* task is not statistically significant ($p = 0.34$) and that in the *Photoshop* task is statistically significant ($p < 0.02$).

The subjective feedback shows that most participants who used our method consider that having the tutorial video respond to the user input like a software is very intuitive and helps them easily find frames of interest. They felt that our method helps them perform the tasks faster in the software as they can practice the interactions beforehand in the video.

Limitations. For a long video with a large number of events, it is difficult for our method to provide a concise summary of all the events at once although our method can enable users to quickly navigate through it by directly interacting with the event visualizations. Moreover, our method currently only captures mouse click and drag events. We will incorporate more user interaction events like keyboard inputs in future.

CONCLUSION

This paper presented a dedicated method for software tutorial video navigation. Our method captures user interaction events and visualizes a set of appropriately selected events in the event locations. Our method allows users to navigate a tutorial video and mimic the captured user interactions as if they were directly interacting with the software. Our preliminary study showed that our responsive video navigation method enables users to follow tutorial videos to complete software-use tasks quickly.

	Excel: mean	std	*Photoshop*: mean	std
Timeline	211.0	78.6	218.6	28.8
Ours	177.8	19.5	155.6	45.9

Table 1. The task completion time (in seconds).

Acknowledgments. Figure 2 and 3 use cat images from Pixabay user wilkernet and DevianArt user Oliver Pietern under a Creative Commons license. This work was supported by NSF IIS-1321119, CNS-1205746 and CNS-1218589.

REFERENCES

1. Banovic, N., Grossman, T., Matejka, J., and Fitzmaurice, G. Waken: reverse engineering usage information and interface structure from software videos. In *ACM UIST* (2012), 83–92.
2. Brockly, C. Evaluation of direct manipulation techniques for in-scene video navigation. *Master's thesis, RWTH Aachen University* (2009).
3. Chi, P., Lee, B., and Drucker, S. DemoWiz: re-performing software demonstrations for a live presentation. In *ACM CHI* (2014), 1581–1590.
4. Denoue, L., Carter, S., Cooper, M., and Adcock, J. Real-time direct manipulation of screen-based videos. In *IUI Companion* (Mar. 2013), 43–44.
5. Dragicevic, P., Ramos, G., Bibliowitcz, J., Nowrouzezahrai, D., Balakrishnan, R., and Singh, K. Video browsing by direct manipulation. In *ACM CHI* (2008), 237–246.
6. Goldman, D. B., Gonterman, C., Curless, B., Salesin, D., and Seitz, S. M. Video object annotation, navigation, and composition. In *ACM UIST* (2008), 3–12.
7. Grossman, T., Matejka, J., and Fitzmaurice, G. Chronicle: capture, exploration, and playback of document workflow histories. In *UIST* (2010), 143–152.
8. Karrer, T., Weiss, M., Lee, E., and Borchers, J. Dragon: a direct manipulation interface for frame-accurate in-scene video navigation. In *CHI* (2008), 247–250.
9. Kim, J., Nguyen, P., Weir, S., and Guo, P. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *ACM CHI* (2014), 4017–4026.
10. Lafreniere, B., Grossman, T., and Fitzmaurice, G. Investigating the feasibility of extracting tool demonstrations from in-situ video content. In *ACM CHI* (2014), 4007–4016.
11. Matejka, J., Grossman, T., and Fitzmaurice, G. Ambient help. In *ACM CHI* (2011), 2751–2760.
12. Pongnumkul, S., Dontcheva, M., Li, W., and Wang, J. Pause-and-play: automatically linking screencast video tutorials with applications. In *ACM UIST* (2011).