Pause-and-Play: Automatically Linking Screencast Video Tutorials with Applications

Suporn Pongnumkul[1], *Mira Dontcheva*[2], *Wilmot Li*[2], *Jue Wang*[2], *Lubomir Bourdev*[2], *Shai Avidan*[2], *Michael Cohen*[3]

[1]University of Washington
suporn@cs.washington.edu

[2]Adobe Systems
{mirad, wilmotli, juewang, lbourdev, avidan}@adobe.com

[3]Microsoft Research
mcohen@microsoft.com

ABSTRACT

Video tutorials provide a convenient means for novices to learn new software applications. Unfortunately, staying in sync with a video while trying to use the target application at the same time requires users to repeatedly switch from the application to the video to pause or scrub backwards to replay missed steps. We present Pause-and-Play, a system that helps users work along with existing video tutorials. Pause-and-Play detects important events in the video and links them with corresponding events in the target application as the user tries to replicate the depicted prodedure. This linking allows our system to automatically pause and play the video to stay in sync with the user. Pause-and-Play also supports convenient video navigation controls that are accessible from within the target application and allow the user to easily replay portions of the video without switching focus out of the application. Finally, since our system uses computer vision to detect events in existing videos and leverages application scripting APIs to obtain real time usage traces, our approach is largely independent of the specific target application and does not require access or modifications to application source code. We have implemented Pause-and-Play for two target applications, Google SketchUp and Adobe Photoshop, and we report on a user study that shows our system improves the user experience of working with video tutorials.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: video tutorial, screen-cast, instructions

INTRODUCTION

Screencast video tutorials are becoming increasingly prevalent as a means of helping novice users perform procedural tasks across a variety of software applications. On YouTube, only one of many video sharing sites, there are over 100,000 video tutorials for Adobe Photoshop, over 30,000 video tutorials for Microsoft Word, and over 12,000 video tutorials for Apple's Final Cut Pro. Google reports 1.8 million global monthly queries for "video tutorial(s)." There are several potential reasons for the growing popularity of such tutorials. First, video tutorials are relatively easy to create; typically, the author simply records a demonstration of the procedure (using widely available screen capture software) and then adds some narration explaining the steps in detail. In contrast, creating well-designed step-by-step tutorials (that include static text and images) often requires the author to capture all of the relevant images of the application user interface, add visual annotations to show complex mouse/pen interactions, lay out the appropriate images for each step, and write clear, concise prose describing the procedure. Furthermore, in contrast to static step-by-step tutorials, videos provide time-varying visual and audio information that can help users learn and understand dynamic procedures (e.g., brushing on a canvas). Finally, unlike human tutors, screencast video tutorials are available on demand to a large audience of users via a variety of Internet video sharing channels.

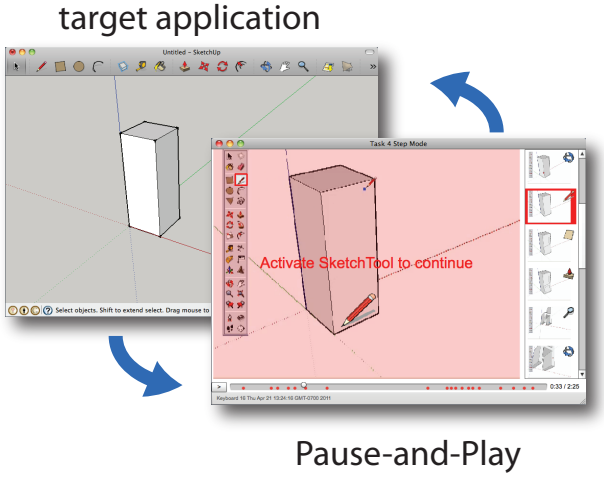

Figure 1: Pause-and-Play links the target application with the progress-aware video player. The video player automatically pauses when the user lags behind the video tutorial, and displays an annotation informing the user how to proceed.

Once the user finds a relevant tutorial, a natural next step is for him to replicate the depicted procedure on a local version of the target application. Typically, users employ what we refer to as a *work along* strategy, where they try to perform

UIST'11, October 16-19, 2011, Santa Barbara, CA, USA.

the procedure while simultaneously playing the relevant portions of the video. While this approach allows the user to see the details of each step as he attempts to complete it, working along with a video involves several important difficulties. First, the user has to manage two pieces of software (the video player and the target application) at the same time. Second, since the user typically ends up working at a different pace from the video tutorial, he often has to switch focus from the target application to the video player in order to pause, rewind or fast forward the tutorial. Finally, if the user misses or forgets a step, it can be difficult to find the relevant portion of the video by scrubbing the video timeline.

In this work, we present Pause-and-Play, a video tutorial system that directly addresses these three problems. Our approach takes an existing video tutorial as input, automatically detects important events within the video, and then links these to corresponding events in the target application as the user works along with the tutorial. This linking allows our system to automatically adapt video playback based on the working pace of the user by pausing and playing the video at the appropriate times. In addition, knowing the locations of important events enables video navigation controls that let users quickly skip to relevant portions of the video.

The three key challenges in realizing such a system are 1) developing an effective automatic approach for detecting important events in a screencast video, 2) defining a lightweight method for tracking the progress of the user in the target application, and 3) designing a set of navigation controls that helps the user work along with the video. To detect important events in the video, we use computer vision to extract relevant metadata. This approach allows us to analyze video tutorials in a way that is largely independent of the specific target application, which makes it easier to extend our technique to a broad range of applications. To track user progress in the target application, we leverage its extensibility infrastructure (e.g., the plug-in architectures and/or scripting support provided by many mature, full-featured applications) to detect relevant events without having to do real-time computer vision processing or to modify the target application's source code. Finally, to design effective navigation controls, we identify key user requirements based on an observational study and then develop a set of automated and interactive controls that addresses these requirements.

While we believe our high level approach generalizes to most types of applications, this paper focuses on generating effective video tutorials for design software, such as Adobe Photoshop, GIMP, Google SketchUp, etc. This is an important class of software with many existing video tutorials targeting these applications. Most design applications share a few key characteristics. They typically adopt a tool-based paradigm in which users can activate and then apply a wide variety of tools to create or edit content. In addition, they often have rich graphical user interfaces (GUIs) that provide visual feedback and support many interaction techniques. We leverage these characteristics in the design of our algorithms for automatically detecting important events.

We should note that we are not the first to present work to enhance video tutorials, as in for example, the Chronicle [7] system. However, in contrast to previous work, our methodology does not rely on having access to the application source code. Rather, we perform instrumentation solely via application scripting plug-ins. While some plug-in architectures might expose enough detailed information to support the extensive instrumentation support necessary for a video-based history system, like Chronicle, the plug-in architectures we have encountered are limited and do not give access to dialog interactions or allow major modifications to the application interface. In a related effort that aims to make it easier to use video tutorials, the AmbientHelp [11] system shows how an application plug-in architecture can be used to create a video recommender system. AmbientHelp, just like Pause-and-Play relies on application plug-ins to track user behavior.

Our work makes the following specific contributions:

- We identify and explicitly describe the challenges of working along with a screencast video tutorial.
- We propose an automated technique for detecting and linking important events in a video to corresponding events in the target application.
- We present automatic and interactive navigation controls that help users work along with a video tutorial.
- We demonstrate our approach with two working prototypes: one for Photoshop and one for SketchUp.
- We describe user evaluations that indicate the effectiveness of our video tutorials approach.

RELATED WORK

While previous work [15] has argued for the benefits of expert demonstrations as a learning aid, there is some debate in the research community about the effectiveness of *video* demonstrations (i.e., tutorials) compared to other types of instructional material. Some evaluations of video instruction (e.g., [12, 8, 9]) found little benefit of video tutorials over traditional text tutorials, despite the fact that videos provide more explicit links between user input and system response and show animated depictions of application dynamics. More recent work has shown that short (10-25 second) contextually available videos that provide "a bare minimum demonstration of how to use a specific tool" are more effective in helping users accomplish tasks and retain what they learned than traditional text-based tutorials [6]. In contrast, researchers studying longer (2-3 minute) task-oriented tutorials found that users performed better with text tutorials than videos because users could not work at the same pace as the video and either missed steps or had to pause, rewind and scrub the timeline to stay in sync with the tutorial [5]. Despite the differences between previous findings, almost all research on instructional videos points to a need for segmented videos that emphasize each step of the task because without segmentation one "may not be able to identify and encode specific aspects of the procedure [8]."

To this end, interactive tutorial systems, like Stencils [10] and DocWizards [1], and systems that offer rich visual histories, such as Chronicle [7] and Interactive Storyboards [16], focus specifically on steps and allow the user to interactively step through or query a procedure. These systems may also be able to automate parts or all of the tutorial procedure, similar

to application macros. Another type of interactive learning aid is an Intelligent Tutoring System (e.g., [9, 4]), which uses artificial intelligence to provide adaptive, customized feedback to students. Although all of these solutions offer compelling alternatives to static text-based tutorials, they typically require significant instrumentation of the application source code and thus incur a heavy cost for supporting new applications. We focus on a more lightweight, application-independent approach that also works for existing videos.

Our approach for detecting important events in existing video tutorials is inspired by recent vision-based techniques for analyzing screencast images, such as Sikuli [17] and Prefab [3]. Much like those existing systems, we use computer vision to extract useful metadata from images (i.e., video frames) of a user interface. In particular, we use a simple analysis technique that is similar in spirit to the visual matching in Sikuli's screenshot search engine. While our informal experiments suggest that our method works more reliably than Sikuli for many of the low resolution and highly compressed screencast videos typically found on the Web, the main contribution of our work does not lie in the specifics of our vision algorithm. Instead, we focus on understanding the challenges of working along with video tutorials and designing a system that helps users overcome these challenges.

Finally, our work is also related to previous research on content-aware video playback and navigation. For example, Petrovic et al. [13] analyze videos using a statistical scene model that enables automatic fast-forwarding to preferred frames, Cheng et al. [2] modify playback speed based on video content, and Pongnumkul et al. [14] present a content-aware dynamic timeline control. Our approach also uses content analysis to improve video playback and navigation. However, we focus specifically on optimizing how users work along with video tutorials of software applications, whereas the systems mentioned above propose techniques for viewing general-purpose videos. Thus, we propose more targeted playback and navigation features designed to help users stay in sync with videos and quickly skip between tutorial steps. In addition, one key distinguishing characteristic of our system is that it automatically links playback to the state of the target application as the user works along with the video.

UNDERSTANDING THE WORK ALONG TASK

In order to develop effective design goals for our video tutorials system, we first conducted a small observational study to help us identify the specific difficulties users face when working along with instructional videos. We recruited 5 participants (4 men, 1 woman) and brought each one into the lab for a 30 minute session. In each session, we asked the participant to perform two red eye removal tasks (one simple and one more complex) in Adobe Photoshop by following video tutorials we found on YouTube. The participants reported some to no previous experience with the software. We chose red eye removal because it is a common task that many users attempt. The videos were 0:37 and 4:11 minutes long, and both had audio narration in English. The short video showed how to use Photoshop's fairly automated "red eye removal tool," while the longer video showed an alternative technique for situations where the automatic tool is not effective. Participants performed the tasks using a single display setup with a 17" 1680x1050 pixel flat screen monitor, a keyboard, and mouse. In addition to observing participants as they worked along with the videos, we also conducted a short interview at the end of each session to determine what additional information and interactions they would have found useful.

Findings

Our observations led us to several key findings about how users work along with video tutorials.

Syncing. Since users almost always end up working at a different pace than the video, the main challenge they face is staying in sync with the tutorial. Thus, the most common interaction pattern we observed is for users to play the video for a few seconds, pause, switch to the target application to replicate the operations they have just watched in the video, and then continue playing the video. The frequency of the pauses varies across subjects but seems fairly consistent for each user. In general, more novice users tend to pause more frequently in order to keep the state of their target application relatively close to that of the video, while more experienced users often watch longer stretches of video to gain more context about a particular set of steps. In either case, users end up switching back and forth between the application and video several times in order to stay in sync with the tutorial.

Seeking. The second most common behavior is to seek along the video timeline to find important events. For example, users often scrub backward to replay specific parts of the video, such as when a menu dialog opens or how the parameters within a dialog are set. Replaying these events reminds the user how to perform the relevant step in the application and also confirms that he has performed the step correctly. In addition, novice users often scrub backwards to replay brief operations that are hard to see and/or not explicitly described in the narration. While less common, scrubbing forwards seems to occur in two main scenarios. Some users will scrub forwards to skip a long introduction to a tutorial, and more experienced users will occasionally scrub ahead when they think they know what is going to happen in a step and want to confirm their theory. In all of these situations, users typically have to scrub around for a few seconds before they find the desired location along the timeline.

Steps. In the closing interviews, users indicated that having information about the meaningful steps in the video could help them work along with the tutorial. Consistent with previous findings [8, 12], knowledge about steps, or video segments, enables users to make informed pauses at the end of step boundaries. Also, having access to the comparison between before and after images for a given step could be useful in confirming that the each step is executed correctly.

Audio narration. The quality of the audio narration for a video tutorial plays a big part in the overall quality of the video. We observed that the higher the expertise of a participant, the more they relied almost exclusively on the audio to accomplish the task. This may be because they already knew where the tools were located and only needed instruction on how to use them. In contrast, novices rely on the visuals to

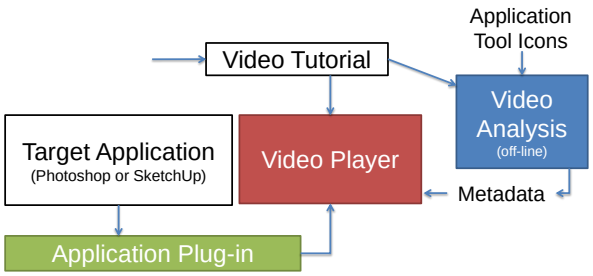

Figure 2: The Pause-and-Play system architecture has three main components: a video analysis module, an application plug-in and a progress-aware video player. It takes in as input the video tutorial.

show them where new tools were located. Additionally, audio narration allows the instructor to generalize the technique and share how it might be used in other contexts.

Screen management. As is often the case when users attempt to multitask by operating two applications simultaneously, managing screen real estate becomes an issue. Most users make the video player relatively large so that they can clearly see the changes that happen in the video. Some try to arrange the screen so that they can see both the video and application at the same time. Others maximize both the video and application and then bring one or the other to the front.

Design Goals

Based on these findings, we have developed several design goals for our system. We focus on optimizing syncing and seeking interactions, as well as enabling step-based navigation. We leave audio integration and screen management to future work.

1. ***Automate play/pause when possible.*** Since playing and pausing is a very common interaction that takes the attention away from the target application and task, we would like our video player to automatically pause and play the tutorial at the appropriate points. As we observed, the frequency of pausing and playing depends on the expertise of the user; thus, any automated play/pause functionality should adapt to the pace of the user.

2. ***Easy access to important frames.*** To prevent users from spending too much time looking for specific locations within the video, we aim to provide easy access to key events within the tutorial.

3. ***Visual highlights***. Since some of the operations in a video tutorial may be brief, subtle and thus easily missed (especially by novice users), we aim to detect and visually emphasize such operations.

4. ***Low transaction cost***. Several of our findings indicate that users frequently switch back and forth between the video and the target application. Thus, we seek to provide an interface where users can focus on the task at hand within the application and still be able to easily navigate the video.

SYSTEM OVERVIEW

Our system, Pause-and-Play, has three main components (see Figure 2): a video analysis module that processes videos and extracts metadata, an application plug-in that generates a trace of user behavior, and the user-aware video player, which automatically pauses and plays a video tutorial as the user interacts with the application and completes each step described in the video.

As noted in the introduction, an important aspect of our work is that it does not rely on access to the application source code. Thus, overall, our approach is largely independent of the specific target application, which means that applying our method to enable smart video playback for new applications is relatively straightforward. That said, we do make some assumptions about the applications we support:

Graphical. We focus on GUI-based applications that are visual in nature, and we assume that all important events have a visual representation in the application. This allows us to take a vision-based approach to detect important events in video tutorials.

Tool-based. Our approach also assumes that the target application adopts a tool-based paradigm in which users activate and then apply various tools to create or edit content. More specifically, we assume that changes in the active tool constitute important events in application usage. While many applications are tool-based (e.g., Microsoft Word and PowerPoint, Adobe Photoshop and Illustrator, Google SketchUp), command-line software (e.g., MATLAB) and form-based applications (e.g., TurboTax) are outside the scope of this work.

Supports extensibility. Many complex applications offer an SDK or scripting API so that third party developers may add new features through plug-ins. We leverage such extensibility capabilities to automatically detect important events in the application while the user works along with a tutorial. In particular, our plug-ins register event listeners and log the stream of user events.

There are a variety of applications that conform to these three criteria, including Microsoft Word, Adobe Photoshop, Google SketchUp, and Autodesk's Maya. Using the overall architecture shown in Figure 2 and the specific techniques described in the remainder of this section, we developed working video tutorial prototypes for two target applications, SketchUp (Figure 3) and Photoshop (Figure 7). The only application-specific components of these two prototypes are the application plug-ins and the tool icons used by the video analysis module.

Interacting with the Pause-And-Play video player

We describe our video player interface in the context of a user task. Julie is learning how to use Google SketchUp, a 3D modeling application, and wants to build a chair. She has found a video tutorial and is following along using the Pause-and-Play video player. The video shows how to select the *rectangle tool* to make a square. At this point, Pause-and-Play automatically pauses (indicated with a transparent red overlay) and instructs Julie to select the *rectangle tool* to continue. Since Julie is not familiar with SketchUp's interface, she looks for the icon that Pause-and-Play emphasizes and highlights in the application toolbar (see Figure 3a). When she selects the rectangle tool the video continues playing.

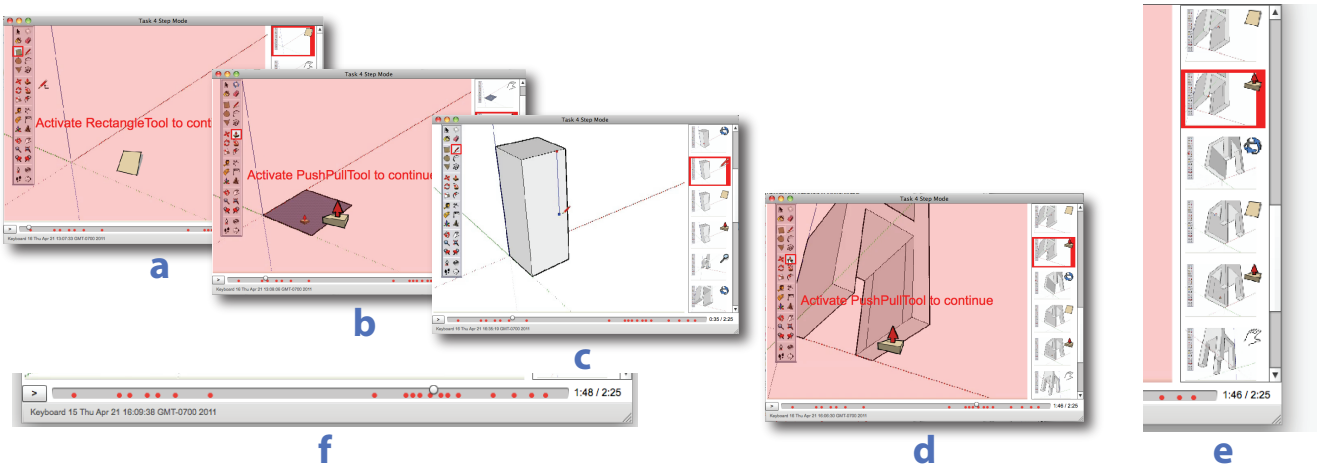

Figure 3: This figure illustrates Pause-and-Play's video player. (a, b, d) The video player automatically pauses and instructs the user to activate the *rectangle tool* or *push-pull tool*. (c) When the user works at the same pace as the video, the video player does not pause. (e) A visual table of contents that shows how the canvas has changed allows the user to jump to different sections of the video more easily. (f) The timeline indicates different segments of the video using red dots. When the cursor hovers over a red dot, the tool name corresponding to that segment is shown.

Julie makes a square as instructed by the video instructor.

When the video moves to the next step in the tutorial, the player automatically pauses and instructs Julie to select the *push-pull tool* (see Figure 3b) to continue. Julie wants to see how to use the *push-pull tool*, so she presses the space bar. She doesn't have to leave SketchUp to interact with Pause-and-Play; she interacts with it as though it is the foreground application. The video shows her how to make a cube. Julie is able to make the cube and follow along at the pace of the video. The narrator says that next he will draw the back of the chair with the *pencil tool*. Julie switches to the *pencil tool* at the same time as the video. This time Pause-and-Play does not pause and keeps going since Julie is keeping up (see Figure 3c).

Later, when Julie is working on the legs (Figure 3d) she gets confused. Her chair legs are not the same width. Julie jumps back to the beginning of the chair leg section using the visual table of contents in Pause-and-Play, which shows visualizations of how the model has changed in each step (Figure 3e). Julie realizes her mistake and begins fixing the legs. She presses the left arrow key to rewind a step she wants to see a couple of times. The timeline of Pause-and-Play shows the video steps with dots (Figure 3f). The left and right arrow keys move backwards and forwards one step at a time.

To enable the Pause-and-Play interface we created a system that could segment a video tutorial into steps and track user behavior. Next we describe these two components in detail.

Segmenting video tutorials

There are a number of recent computer vision approaches for analyzing screen captured images including the Sikuli [17] and Prefab [3] systems. These approaches inspired our initial investigations in processing screencast videos. Much like Sikuli and Prefab, our video analysis module performs pixel-based analysis of screencast images (i.e., video frames) to identify user interface elements and detect important changes in application state. The main challenge in applying existing techniques directly is that most video tutorials found on the Web are low resolution and include many compression artifacts, which make it impractical to use the exact pixel-by-pixel matching methods of Prefab. While Sikuli's approach of using SIFT features could theoretically handle compressed, low resolution input, our informal experiments using the Sikuli implementation published online produced unreliable matching results. Our approach uses a robust template matching designed to handle the noise due to resampling and compression.

In designing our video analysis algorithms, we made the following observations:

- tool palettes in SketchUp and Photoshop are made up from a grid of icons,
- the active tool is visually emphasized,
- and there are limited number of icons.

For a given target application, the video analysis module takes as input a tool palette template and the application tool icons and produces a metadata file that lists tool changes with video timecodes. A tool palette template is an image of the tool palette that can be captured from the running application at any resolution. For example, we took a screenshot of Photoshop to get its tool palette template. The tool icons are part of an application's installation and thus are easily accessible.

In each frame of the video, the system first finds the tool palette by comparing to the tool palette template. The comparison is done in greyscale using a blurred version of the template, which adds robustness to noise, color and scale variations. We use the sum of the differences of the grayscale pixel values. As the resolution of the video may be different from our template, we conduct the search in multiple resolution and find the best match.

Next, the system identifies the active tool position by analyzing the frame differences between adjacent frames. The sys-

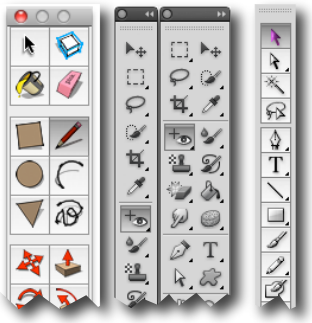

Figure 4: Tool palettes for Google SketchUp, Adobe Photoshop (two versions), and Adobe Illustrator

tem considers all differences in the tool palette region that are rectangular and grid aligned as tool changes. Notice how in Figure 4 the *pencil tool icon* in the Sketchup tool palette appears differently than the rest of the icons. Once the selected tool position is identified, the system searches through the available tool icons to identify the selected tool.

Some applications, such as Photoshop, use very similar icons for different tools, which can make it difficult to identify the correct selected tool in the video. To make our analysis more robust, we associate a preferred tool palette grid position with each tool icon as part of the input, which limits the number of icons the system must match against for a given selected tool position.

Some video tutorials include special effects such as camera cuts, zooms and pans. While we mainly experiment with "static camera" screencast videos, our template matching does not strictly assume a static camera. Since we perform template matching at multiple resolutions, we can handle cases where the camera cuts or zooms/pans to a different portion of the screen, as long as enough of the tool palette is still visible to find a good match. To support a variety of camera effects, we may need to search over a wider range of resolutions, which would slow the performance of our video analysis module.

The video analysis module is implemented in MATLAB. Our current implementation does not process video in real-time but it could be optimized to work in real-time. We expect that each video would be processed off-line once, and its metadata would be available for anyone who wants to use the Pause-and-Play system.

Linking video playback to user behavior

For Pause-and-Play to respond to user interactions it requires a real-time trace of user events such as keyboard use and tool use. While a user trace can be generated through computer vision or application-specific instrumentation, using application plug-ins that register callbacks for application events provides a fairly lightweight alternative that does not require modification of an application's source code, while still providing rich application-specific information.

To allow users to control the video playback from within the target application, we customized application shortcuts to trigger the plug-in to write commands to the user trace. In our studies with Google SketchUp, we used the space bar and arrow keys to control the video player, but a user can customize other keyboard shortcuts to control the Pause-and-Play player.

The application plug-in generates a log file and stores two types of events: the active tool and any keyboard shortcut events. The active tool events are detected by registering to the appropriate callback events. For example, in Google SketchUp we register a callback for the `onActiveTool-Changed` event. The log includes each tool or key name and a timestamp. The video player reads from the log file to check for new events every 0.5 seconds. Therefore, it gets real-time updates of application events. As the video plays, the video player checks the metadata associated with the video. When there is a tool change in the video the player checks whether the user's active tool matches the current video tool. If not, the player automatically pauses the video to wait for the user to catch up, as it expects that the user is still working on the previous step. As Figure 3d shows, the player prompts the user with a message asking him to activate the appropriate tool to continue. Once the user activates the tool, the video player automatically plays the video, as this means that the user has caught up and completed the previous step. The user can always override the video player and play or pause the video by pressing the spacebar, clicking on the play/pause button in the video player, or clicking on the video itself.

After experimenting with many videos, we modified the video player to automatically pause not at the exact moment that the instructor changes the tool but two seconds later. We found that often instructors select a tool and then say the tool's name. This two second buffer worked well for most videos, but a more robust approach would be to analyze the audio stream and find appropriate audio boundaries.

Limitations

Although our techniques for segmenting videos and linking them to user behavior work well for many of the SkecthUp and Photoshop video tutorials we found on the Web, our approach does have some limitations:

Other types of important events. Our video analysis module segments videos by detecting active tool changes. However, some tutorials include other types of events that represent important steps for completing the task, such as opening a dialog box or changing the parameters of a tool. Our system currently does not detect such events.

Single tool tasks. A few of the video tutorials we encountered include long sequences of complicated actions that only involve a single tool (e.g., creating a painting in Photoshop using just the brush tool). Since we do not analyze changes in the actual document or 3D model in the video, our system would only extract a single step for such sequences, even if there are several semantically meaningful steps in the procedure. While this is a limitation, we found tool changes to be a reasonable indicator of step boundaries for most SketchUp and Photoshop videos.

Incorrect playback. The fact that our adaptive playback mechanism only considers the currently active tool in the target

application and not whether a user has completed a step may result in the video resuming playback at the wrong time. For example, the user might undo a few steps and select the *rectangle tool* in order to redraw a rectangle. Since undo operations are not tracked by our system, the system will proceed playing the video, as it thinks that the user is ready for the next step, while he is in fact a few steps behind.

Work along errors. Finally, our playback design does not detect whether the user makes mistakes when completing a step. Thus, Pause-and-Play will allow users to continue through a tutorial even if they make errors along the way. We leave automatic error detection for future work.

USER STUDY

To evaluate Pause-and-Play we carried out a qualitative user study comparing a traditional video player to our Pause-and-Play video player. Based on feedback from this study we improved the design of the Pause-and-Play video player and gathered a second round of user feedback to assess the effects of the changes. We carried out both studies with Google SketchUp because compared to Adobe Photoshop, fewer people are familiar with this application and 3D modeling, and thus it was easier to control for previous experience.

Our goal was to answer the following questions:

- Is automatic pause and play useful?
- Does Pause-and-Play pause at appropriate times?

Methodology

We recruited 15 participants (11 male, 4 female) ranging in age from 18 to 49 using a university distribution list. We targeted participants with little or no experience with Google SketchUp or other 3D modeling tools. All participants were compensated with a $20 gift card.

We used a within-subjects experimental design. Each participant was asked to complete four tasks: two with a traditional video player (control condition), and two with Pause-and-Play (experimental condition). To minimize the effects of the look and feel of existing commercial video players, we created a basic version of Pause-and-Play that did not include any of the newly proposed features but allows users to pause, play, and scrub, much like most standard video players. We independently varied the order of the interfaces.

For the four tasks, we chose video tutorials that illustrate how to create an object: a dog house (6 minutes), a sphere (1.8 minutes), a house (3.7 minutes), and a chair (2.4 minutes). All of the videos had audio and were available online at YouTube when we did the study. All participants performed the tasks in the same order.

The participants completed the tasks in one-hour sessions, with a moderator leading the session. The participants were told to follow the tutorials as best as possible. At the beginning of each session, the participant filled out a short questionnaire about his or her previous experience with 3D modeling applications. After filling out an introductory questionnaire, the participants were introduced to Google SketchUp through a short (18 seconds) video that shows how to create a box. The participants were asked to replicate the box task. Each of the four subsequent task lasted 5-10 minutes, and the participants completed a short questionnaire after each one. No assistance was provided, but if after 20 minutes a participant had not completed the task, he or she was asked to to continue on to the next task. At the end of the four tasks, the participants filled out a final questionnaire comparing the two video player interfaces and provided free-form comments. We asked each user to complete a spatial cognition test at the end of the session to control for variability in 3D spatial cognition.

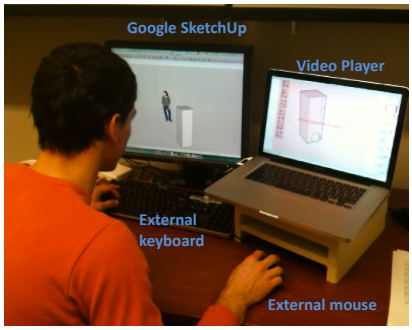

Figure 5: This figure illustrates the user study set up. One participant is working on the chair task.

The laboratory setup included a 2.66GHz Intel Core i7 macbook pro running Mac OS X 10.6, with an external monitor, a standard external keyboard and a standard external mouse, as shown in Figure 5. The video audio played via the laptop speakers. Google SketchUp ran on the external monitor which is set as a main display at 1280x1024, while the video displayed on the laptop monitor at 1280x1024. We chose a two-display setup because we wanted to minimize the effects of limited screen space. Although many users have two displays, those who only use one display face a real challenge in using two applications simultaneously. We don't focus on this challenge in this work.

Results

Overall the participants were very positive about Pause-and-Play. Although we did not see any significant differences in completion time or error rate, 13 out of the 15 participants preferred the Pause-and-Play video player over the traditional player.

We observed a wide variety of behaviors for using video tutorials. With the traditional player 13 of the 15 participants preferred to work at the same time as the video was playing, while the remaining two preferred to watch the entire video before executing any of the steps. In both cases the participants did end up syncing and seeking when they were using the target application. Only one person was not able to complete a task (sphere task). Fourteen participants used the space bar to play and pause the video. Eight participants used the left arrow key to move backward. Only one participant did not use any keyboard shortcuts. And only one of the participants used the visual outline to move to a different segment of the video.

Is automatic pause and play useful? Thirteen participants

found the automatic pausing and playing useful for learning new tools. For example P1 said, *"It pauses on the tool change–I find this extremely helpful when there is a new tool being introduced that I haven't used before and don't know by name. Pausing and 'unpausing' was also straightforward."*

The participants also mentioned that the automatic pausing made it easier for them to follow the instructions because it made it impossible for them to fall behind. P3 mentioned, *"I liked the automatic stops, because if I got behind I didn't have to stop the video myself."* P9 said *"Basically the step mode automates what I already do in manual mode. Also, the tasks were broken into logical pieces; since the video paused until I chose the correct tools, there was no way to fall behind. There were many times in manual mode when the video played further ahead, but I really wasn't listening because I was too far behind. In fact, the sound of the video continuing made me anxious to catchup and I was less efficient with my activity."*

Even participants who when using the traditional video player would watch the entire video before executing the steps, found Pause-and-Play useful and changed their behavior. For example, P5 said *"I liked [Pause-and-Play] because usually I try to see the whole video once and then try out the tasks. Because if I start on the task, I will miss what is being said in the video. But this step video removed that problem and encouraged me to work on the software side by side."*

On the other hand, this model of work was not favored by everybody. Two of our participants preferred the traditional video player because it was familiar and appeared to provide more control. P12 mentioned *"I prefer this one [traditional] because it is easier to play, pause and rewind at will, I think because it looks similar to other video tools I have previously used. ... I don't think I knew whether I could pause and rewind at will, I felt like I could only rewind/forward from one section to the next/previous one and not in between."* P15 said *"The traditional player is more smooth and easier to follow because I can pause whenever I want. I sometimes find Pause-and-Play disturbing for pausing at unnecessary places (such as selecting a tool that's commonly used) or at weird places (in the middle of a phrase). However, for a more complicated project, Pause-and-Play may be more useful, if it involves using a lot of new tools, but for simple designs like the ones that's done in this study, the traditional player is enough."*

Does Pause-and-Play pause at appropriate times? Given that 13 out of 15 participants found the pausing useful, it seems that even our simple model of keeping track of the active tool helps users stay in sync with the video. However, there is room for improvement. Seven participants mentioned that there were some pauses that were not necessary. For example, pausing for camera tools seemed unnecessary since the camera tools were often used to show different sides of the 3D model and were not specific steps in the procedure. Also for some videos, such as the chair video, the tool changes happen so quickly that the video pauses too often, resulting in choppy short clips of video that don't include a complete step. The study made it clear that the automatic pausing needs to be sensitive to step duration.

Controlling the video from the application. Four participants mentioned that the ability to control the video from Google SketchUp was useful. P7 said *"... the complete integration with the SketchUp program was great. I mean, not having to switch over was nice"* P1 said *"The good thing of Pause-and-Play is that I can start and stop the video with the space key, which makes it easier than having to move the mouse all the way to the video window, find the pause/play button and press it."*

The segmented timeline was also mentioned by five people. One user mentioned that *" [Pause-and-Play] provided convenient 'chapters' so i didn't have to manually scrub to a random point in the video."*

We also looked at whether users complete video tutorials faster and with fewer errors with Pause-and-Play. However, we did not find significant differences in completion time or error rate. We think this may be due to two factors. First, the chosen videos vary significantly in difficulty, teaching style and speed. Second, users struggled with different steps and varied in their attention to detail. While we recruited participants with no experience in 3D software, we found varying levels of expertise from novice to intermediate, perhaps influenced by expertise with GUIs and 3D concepts. Both of these issues and the fact that we did not tell the participants to go ask quickly as possible confound direct comparisons of completion time and errors.

PAUSE-AND-PLAY DESIGN ITERATION

In response to the feedback from our participants, we sought to improve the Pause-and-Play design. We made the following modifications.

- We removed pauses for camera and canvas navigation tools, including the *zoom tool*, *hand tool*, *orbit tool*, and *dolly tool*. These tools do not change the 3D model or image and using them is not required to complete the tutorial.
- Since only one participant used the visual table of contents, we removed it.
- We added a display for the upcoming tool for the users who were able to keep up with the pace of the video (Figure 6a). By selecting the next tool early, users avoid any pauses in the video playback.
- We observed that when the steps were short and the video quickly switched tools, users often got confused. To assist with fast tool changes, we removed automatic pausing for tool changes that were less than five seconds apart. To show the user that the video had now moved on to a new tool, we added a visual list that shows the user the steps he must complete to catch up to the video (Figure 6b). The video player continues playing until the user has fallen behind a maximum of three steps. If the user falls behind three steps, the player automatically pauses and waits for the user to catch up (Figure 6c). As the user accomplishes each of the steps, the player shows his progress in the video display (Figure 6d).

Second round of user feedback

Our goal for the second user study was to understand whether the updated design led to pausing in more appropriate parts

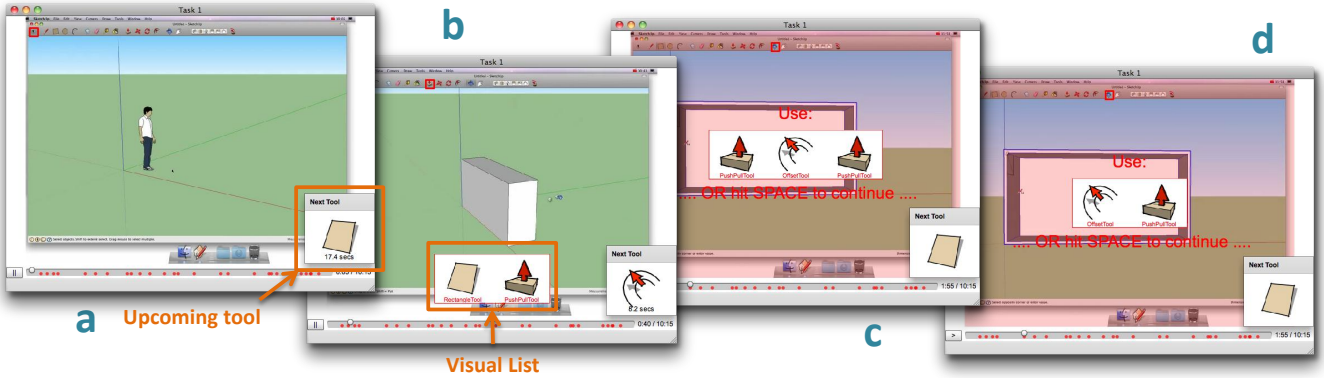

Figure 6: We changed the Pause-and-Play video player interface in response to user feedback. (a) For advanced users we show the next tool in the bottom right corner. (b) We remove pausing for short steps and show the user the steps as a visual list. (c) After a maximum of three missed steps, the video pauses. (d) As the user makes progress, the display updates to show the remaining steps.

of the video. We considered bringing back some of our participants from the first study, but we were concerned that too much time (a month) had passed since the original study. To get a sense for whether we were on the right track, we carried out a pilot study with the new redesigned Pause-and-Play video interface.

We recruited 4 participants (2 male, 2 female) ranging in age from 18 to 65. The laboratory set up and the recruiting criteria were the same as the first study. Instead of 4 tasks with two video players, we asked them to perform only two tasks (the house and chair tasks) with only one interface (the redesigned Pause-and-Play video player). Again we considered the house task as a practice task. After they accomplished both tasks, we asked users to rate the frequency of the pauses on a scale of 1 (too infrequent) to 5 (too frequent) where 3 is about right and asked for open ended feedback on the video player experience. Each participant session lasted 30 to 50 minutes, and all participants were compensated with a $15 gift card.

All of the participants completed the tasks and used the keyboard shortcuts to play and pause the video. Three of the participants used the left arrow to move back to previous steps. The participants rated the frequency of pauses as 2.8, 3, 2.5 and 2.5 and expressed that there were times when they wished the video has paused when it didn't. S1 mentioned *"One specific place I got lost was in selecting materials. I missed the step of selecting 'bricks' instead of the default 'roofing', and so picked the most likely looking material, which turned out to be some shingles. It might be good to pause until the user picked the correct material. Otherwise, the pauses seemed well placed. Waiting just until I picked the correct tool seemed helpful."* S2 said *"there was one instance where I wished it had paused earlier."* S4 mentioned *'I used Space-bar pausing - so this means that it was not frequent enough for me."*. So although no one mentioned that the video paused too often, three of the four participants felt that perhaps there weren't enough automatic pauses.

The display of the upcoming tool (Figure 6a) was not heavily used, perhaps because the pace of the video was such that the participants were focused on the task and were not looking ahead. Although the display did not seem to prompt users to pick the next tool, at least one of the participants found it useful. S2 mentioned that *"knowing which tool comes up next is good to keep in mind. Mentally helps you know where the tutorial is heading."* S4 found *"the box showing the Next tool was not helpful but not really too distracting either."*

All of the participants observed a list of upcoming tools (Figure 6b) two or more times . While it seemed that the visual list was useful, there were mixed feelings about the lack of an automatic pause between two different steps in the tutorial. S4 expected a pause after each tool and was distracted when *"the 'teacher' would do two things, and the pause would happen after both instead of after each one."*. S4 found the visual list particularly useful when she wanted to go back and fix mistakes. *"When it was waiting - it would list the tools I needed next. This was especially useful when I went back to fix mistakes, and came back - it helped me remember what I had to do next."*

In conclusion, the second round of user feedback showed us that we improved the ability of the video to stay in sync with the user by removing navigational tools, though there is room for improvement in situations when tools are used in quick succession. Since the participants expected that the video player would pause for every tool, they found the aggregation of up to three tools distracting. On the other hand, the visual list was found to be useful when users replay the video, reminding them of the steps.

CONCLUSION AND FUTURE WORK

We have demonstrated an approach for helping users replicate the steps in existing video tutorials. Based on an observational study, we identified key challenges that users currently face when working along with videos, and we designed an interactive video tutorial system, Pause-and-Play, that addresses these problems. Pause-and-Play detects and links important events in existing videos to corresponding events in the target application, which enables progress-aware video playback based on the working pace of the user and convenient navigation controls for skipping directly to rel-

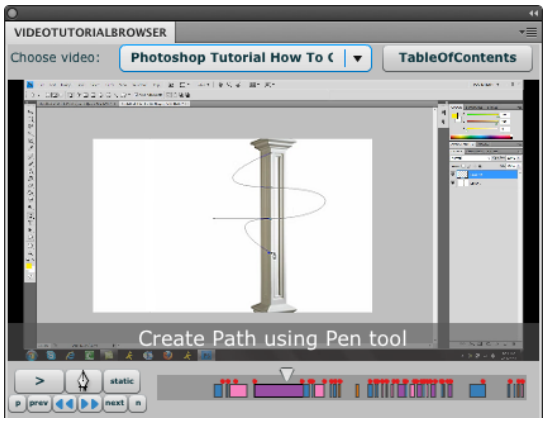

Figure 7: In addition to the core adaptive playback and navigation features described in the System Overview, our Pause-and-Play prototype for Photoshop allows the tutorial author to manually specify additional metadata for each step, including customized text descriptions (as shown here), visual annotations, and zoom locations that focus on specific parts of the interface.

evant portions of the video. Since our method does not require access or modifications to the application source code, there is a relatively small overhead for supporting new applications.

We see several interesting directions for future work. In order to deploy a system like Pause-and-Play in real world settings for many actual users, we need more investigation. As mentioned earlier, one limitation of our system is that we only detect active tool changes in the input video. Thus, we could extend our vision-based video analysis to detect other events (e.g., menu selection, opening a dialog, drawing operations) and include audio input to avoid placing step boundaries in the middle of a sentence. However, while such techniques would certainly improve our video segmentation, a completely automatic analysis will likely never be perfect. As a result, we plan to enable tutorial authors to improve automatic segmentation by adding, deleting or changing step boundaries, and enhancing the tutorial with additional metadata. For example, the author could add annotations that enable expertise-dependent step boundaries, ease search and navigation, and indicate the most salient regions of the user interface for specific steps. We have experimented with adding some of these manual authoring features to our Pause-and-Play prototype for Photoshop (Figure 7).

One common problem with following a tutorial occurs when there is a mismatch between the user's content and the content used in the tutorial. For example, the method for selecting the red region in a red-eye removal tutorial may not work well if the eye is too small or the image is blurry. In this case an inexperienced user may get stuck, and his only option is to look for another more suitable tutorial. A future extension would allow users to create a discussion thread associated with a video segment in order to share any difficulties they may be experiencing. More expert users could provide alternative solutions, for example, a link to another tutorial that handles "blurred red eye removal".

In conclusion, we believe Pause-and-Play is only a first step in bridging the gap between applications and video tutorials. We are confident there are many other unexplored ways of leveraging video demonstrations to encourage and empower users to learn a wide variety of software applications.

ACKNOWLEDGMENTS

We thank the user study participants for their time and valuable comments.

REFERENCES

1. L. Bergman, V. Castelli, T. Lau, and D. Oblinger. Docwizards: a system for authoring follow-me documentation wizards. In *Proc. ACM UIST*, pages 191–200, 2005.
2. K.-Y. Cheng, S.-J. Luo, B.-Y. Chen, and H.-H. Chu. Smartplayer: user-centric video fast-forwarding. In *Proc. SIGCHI*, pages 789–798, 2009.
3. M. Dixon and J. Fogarty. Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proc. SIGCHI*, pages 1525–1534, 2010.
4. S. Gilbert, S. B. Blessing, and S. Kodavali. The extensible problem-specific tutor (xpst): Evaluation of an api for tutoring on existing interfaces. In *Proc. the conf. Artificial Intelligence in Education*, pages 707–709, 2009.
5. F. Grabler, M. Agrawala, W. Li, M. Dontcheva, and T. Igarashi. Generating photo manipulation tutorials by demonstration. In *Proc. ACM SIGGRAPH*, pages 1–9, 2009.
6. T. Grossman and G. Fitzmaurice. Toolclips: an investigation of contextual video assistance for functionality understanding. In *Proc. SIGCHI*, pages 1515–1524, 2010.
7. T. Grossman, J. Matejka, and G. Fitzmaurice. Chronicle: capture, exploration, and playback of document workflow histories. In *Proc. ACM UIST*, pages 143–152, 2010.
8. S. M. Harrison. A comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphical user interface. In *Computer Human Interaction*, pages 82–89, 1995.
9. C. Hategekimana, S. Gilbert, and S. Blessing. Effectiveness of using an intelligent tutoring system to train users on off-the-shelf software. In *Proc. Society for Info. Tech. and Teacher Education Intl Conf., AACE*, 2008.
10. C. Kelleher and R. Pausch. Stencils-based tutorials: design and evaluation. In *Proc. SIGCHI*, pages 541–550, 2005.
11. J. Matejka, T. Grossman, and G. Fitzmaurice. Ambient help. In *Proc. SIGCHI*, pages 2751–2760, 2011.
12. S. Palmiter and J. Elkerton. An evaluation of animated demonstrations of learning computer-based tasks. In *Proc SIGCHI*, pages 257–263, 1991.
13. N. Petrovic, N. Jojic, and T. S. Huang. Adaptive video fast forward. *Multimedia Tools Appl.*, 26:327–344, 2005.
14. S. Pongnumkul, J. Wang, G. Ramos, and M. Cohen. Content-aware dynamic timeline for video browsing. In *Proc. ACM UIST*, pages 139–142, 2010.
15. B. Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57 –69, 1983.
16. S. L. Su. *Enhanced Visual Authoring Using Operation History*. PhD thesis, Massachusetts Institute of Technology, Boston, Massachusetts, 2009.
17. T. Yeh, T.-H. Chang, and R. C. Miller. Sikuli: using gui screenshots for search and automation. In *Proc. ACM UIST*, pages 183–192, 2009.