Swifter: Improved Online Video Scrubbing

Justin Matejka, Tovi Grossman, and George Fitzmaurice
Autodesk Research, Toronto, Ontario, Canada
{firstname.lastname}@autodesk.com

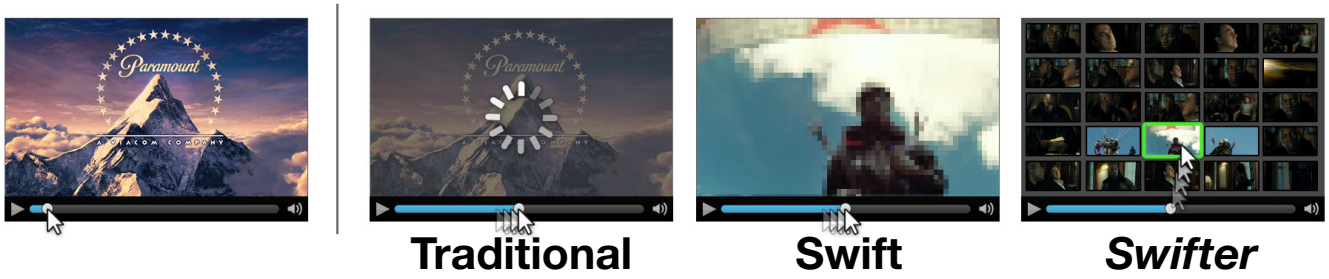

Figure 1. Scrubbing behavior of a traditional streaming video player, the *Swift* interface [16], and our new *Swifter* interface, which shows multiple frames around the active timeline location and allows for direct selection of each frame.

ABSTRACT
Online streaming video systems have become extremely popular, yet navigating to target scenes of interest can be a challenge. While recent techniques have been introduced to enable real-time seeking, they break down for large videos, where scrubbing the timeline causes video frames to skip and flash too quickly to be comprehendible. We present *Swifter*, a new video scrubbing technique that displays a grid of pre-cached thumbnails during scrubbing actions. In a series of studies, we first investigate possible design variations of the *Swifter* technique, and the impact of those variations on its performance. Guided by these results we compare an implementation of *Swifter* to the previously published *Swift* technique, in addition to the approaches utilized by YouTube and Netfilx. Our results show that *Swifter* significantly outperforms each of these techniques in a scene locating task, by a factor of up to 48%.

Author Keywords
Video; Video Navigation; Online Streaming

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Graphical User Interfaces.

INTRODUCTION
Online streaming video systems have become extremely popular over the past ten years. Streaming video sites allow users to instantly view content without having to first wait to download large video files. Today, video streaming sites account for a significant portion of all internet traffic, with some estimates suggesting that soon 90 percent of Web traffic will be video [22].

CHI 2013, April 27–May 2, 2013, Paris, France.

However navigation of streaming videos can be difficult. When a video is local, it is possible to "scrub" the video, by moving a slider along the timeline and having the current frame of the video update in real-time. But when videos are streamed, the latency required to obtain the current frame makes real-time scrubbing impractical.

Recently, several techniques have been introduced, such as Swift [16], as well as the YouTube[1] and Netflix[2] online video players, that support real-time seeking in streaming videos by displaying low resolution thumbnail images during scrubbing operations. While such techniques address the latency problem during navigation tasks, they break down for longer videos, where scrubbing the timeline causes video frames to skip and flash too quickly to be comprehendible [17]. Addressing this problem for long videos is more important now than ever, as it has become common for full length movies and TV episodes to be streamed on sites such as Netflix, YouTube, and Hulu[3].

In this paper, we present *Swifter*, a new video scrubbing technique that displays an interactive grid of pre-cached thumbnails during scrubbing actions (Figure 1). Swifter is designed under the principal that providing a large set of thumbnails during navigation will make it easier for user to visually identify a desired target region [8, 14]. The technique also reduces the chances of a scene being skipped during navigation, and requires no modification to the video player's standard layout.

Our work contributes studies that inform, evaluate and quantify the design parameters of the *Swifter* technique. We first investigate possible design variations of the *Swifter* technique, and the impact of those variations on the performance of the technique. We then compare *Swifter* to the previously published Swift technique, in addition to the

[1] www.youtube.com
[2] www.netflix.com
[3] www.hulu.com

approaches utilized by YouTube and Netfilx. Our results show that Swifter significantly outperforms each of these techniques in a scene locating task, by up to 48%. We conclude by discussing issues surrounding deployment, and opportunities for future work.

RELATED WORK

In this section we discuss related work which has aimed to improve video navigation. We also discuss previous use of gridded thumbnail layouts.

Video Navigation

Numerous projects have explored video navigation improvements. A simple enhancement is to modify the traditional timeline slider with different dynamics. The AV-ZoomSlider [13] uses the additional dimension of the y-axis to enable more precise control, while the PVSlider [18] provides an elastic scrubbing experience for more fluid control. These techniques can prevent frames from being skipped, but only if the video is being navigated slowly.

The content-aware dynamic timeline [17] uses an elastic slider and presents salient scenes, based on a key clip hierarchy, at consumable speeds during seeking. This can improve the video seeking experience. However, entire scenes could be skipped during seeking, making it less appropriate for scene finding tasks.

Another approach for aiding video navigation has been to augment the video timeline with information and graphical elements, such as low-level features [19], scene boundaries [3] workflow meta-data [11], or user contributed metadata [7, 15]. While these techniques can aid video navigation, they do not improve the real-time scrubbing experience and require modifications to the video player layout.

Collections of scene thumbnails have also been used to provide illustrated summaries of videos [1, 12, 14]. We refer the reader to Truong and Venkatesh [21] who provide a survey of work on the creation of thumbnails for video skimming and abstraction. Such techniques include hierarchical thumbnails [10], thumbnails lists [5], fisheye views [9], and video tapestries [4]. These techniques provide interesting mechanism to quickly obtain overviews of a video's content, but do not necessarily improve performance for scene finding tasks [4].

To this end, the Swift technique was recently introduced [16]. Swift improves scene-finding of online videos by overlaying a low resolution copy of the video during seeking, to support real-time scrubbing. While the Swift technique works well for short videos, it may break down when trying to find short scenes within long videos, as frames can flash by too quickly and entire scenes can be skipped. The *Swifter* technique was designed to specifically address this shortcoming.

Existing Deployed Technologies

Many different streaming web-players can be found on the internet. The most common and status-quo behavior during scrubbing is to not update the rendered frame until seeking stops (Figure 1, "Traditional"). Some players, such as Hulu's, provide a thumbnail preview when the cursor hovers over the timeline. However, these thumbnails are not pre-loaded so they still suffer from latency effects.

More recently, both Netflix and YouTube have begun using small thumbnails that are visible during video seeking actions. With YouTube, a strip of thumbnails are displayed at the bottom of the video player, with the current frame enlarged and in the center (Figure 2A). With Netflix, a single Thumbnail is displayed at the bottom of the player (Figure 2B). Both techniques can be considered variations of Swift's design, although YouTube's design of showing multiple thumbnails is an interesting variation. And as with Swift, both techniques can still suffer from indiscernible flashing thumbnails when the user scrubs longer videos.

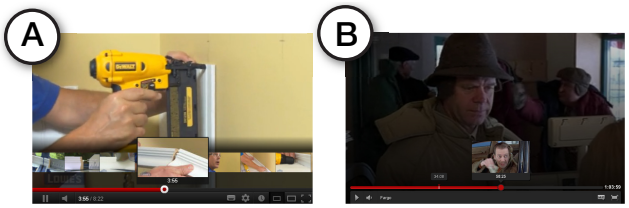

Figure 2. Screenshots of the YouTube (A) and Netflix (B) browser based video players.

Gridded Thumbnail Layouts

An active area in research related to our own is in document navigation [1, 5]. Space filling thumbnails [8] provide an interesting alternative to traditional scrolling when navigating through large documents. With Space-Filling Thumbnails, users switch between a normal reading view to a thumbnail view where all document pages are scaled and tiled to fit the window. An advantage of this technique is that users can rely on visual features to navigate to desired pages at unknown locations in the document. This technique serves as inspiration in the design of *Swifter*.

Gridded layouts are also commonly in file browsers and photo management software on PCs, mobile devices, and even some camera models, but have not been used before to improve streaming video navigation.

SHORTCOMINGS OF EXISTING SOLUTIONS

We have found that when working with feature length films existing solutions for high-latency video scrubbing suffer in two main phases of a video navigation task: searching for a desired scene, and then once it is located, selecting it.

The Searching Phase

The searching phase of a video navigation task requires the user to navigate through the frames and visually recognize the scene they are trying to find. This is accomplished by dragging the playhead slider along the timeline, and monitoring the displayed frame. Due to software and screen refresh rates, not all of the available frames are displayed. For example, if we consider a 854 pixel wide timeline (the "medium" size on YouTube), with the user moving from the

beginning to the end of the timeline at a constant speed over a period of 2 seconds, with the display updating at 30 frames/second, then only 30 frames/sec × 2sec = 60 frames out of the 854 will be displayed on the screen during that scrubbing action (Figure 3).

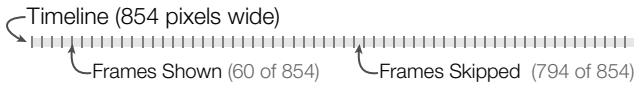

Figure 3. Frames shown (dark lines) and skipped while moving over a 854 pixel wide timeline in 2 seconds with an update rate of 30fps.

If the 60 displayed frames are distributed evenly across the timeline, then between each two displayed frames, there will be 14 frames that are skipped. For a 120 minute movie, those 14 skipped frames account for approximately 118 seconds of film (14×120×60/854). At that rate of movement, any visual segment shorter than 2 minutes in length runs the possibility of falling in one of the gaps and not even having an associated frame displayed. The YouTube player begins to address this problem by displaying multiple thumbnails representing associated frames (Figure 2). But even then, when displaying 9 frames at a time, in the above scenario only 540 (60×9) of the 854 frames would be displayed, creating gaps of 5 frames each, or roughly 42 seconds long (Figure 4).

Figure 4. Frames shown (dark lines) and skipped while moving over a 854 pixel wide timeline in 2 seconds with an update rate of 30fps with a YouTube-style controller.

However, even those frames that are displayed are only on screen for 1/30th of a second, or, 33 milliseconds, making it feasible for a frame from the desired scene to be displayed, but not recognized by the viewer.

The Selection Phase

Once the target scene has been located, the user has to select a frame to begin playback from. This is done by tweaking the position of the cursor/playhead slider until the desired frame is selected. If precise positioning is required, this could mean moving the playhead slider to a target as small as one pixel wide along the timeline. Even selecting within an area of 30 seconds requires releasing the cursor within a target 4 pixels wide, which can be quite difficult.

The YouTube player supports a precision selection mode for longer videos, similar in nature to the AV-ZoomSlider [13]. However, this is only available while hovering over the timeline, not during scrubbing, and still requires precise positioning of the cursor before entering this mode.

SWIFTER

To overcome the limitations of existing solutions, we developed *Swifter*. The *Swifter* technique improves the *searching* phase by displaying an array of thumbnails, representing frames in the video, in a grid, making it potentially easier to find the target scene (Figure 5). It improves the *selection* phase by allowing the user to choose between selecting a frame indirectly from the timeline, or by directly clicking on a frame's thumbnail.

As with previous techniques [16], the total number of thumbnails which can be displayed is equal to the pixel width of the timeline. The thumbnails are organized in increasing order from left-to-right, and top-to-bottom, representing frames evenly distributed throughout the video.

Scrubbing the Video

When scrubbing is activated, by clicking on the timeline or playhead slider, the grid of thumbnail images is displayed. The actively selected thumbnail is indicated with a thick contrasting outline, and as the cursor is moved from left-to-right, the selection moves from left-to-right and top-to-bottom through the array of thumbnails. If the cursor is moving quickly (>50 pixels/sec) we dim the active thumbnail outline so as not to distract the viewer from the thumbnail content. When the cursor movement slows down, we draw the selection outline at full opacity again.

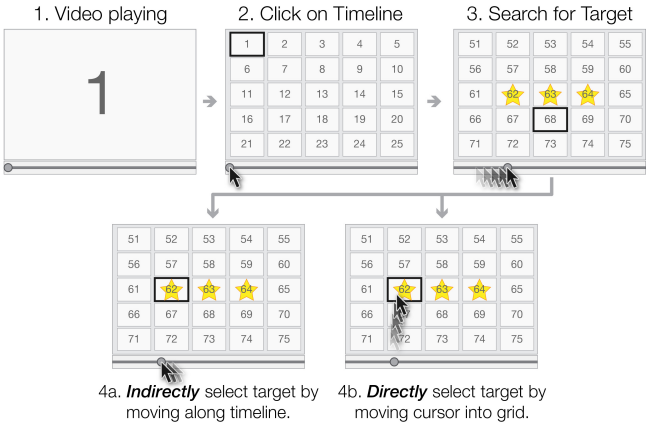

Figure 5. Mechanics of selecting a thumbnail using either the *direct* or *indirect* selection method. The cell with the thick outline indicates the currently selected thumbnail.

Scrolling Direction

Since there are more thumbnails available to be displayed than can fit on the screen at once, *Swifter* requires a method to scroll through them. Early stages of the design process explored both vertical and horizontal scrolling. Our initial pilot tests indicated that users found the vertical scrolling arrangement much easier to use, so we used vertical scrolling for our implementation and all of the studies.

Scrolling Techniques

We consider three possible techniques for scrolling, adapted from document and file browser scrolling paradigms. The performance differences of these scrolling techniques will be explored in Study One.

Page-at-a-Time Scrolling

In *page-at-a-time* scrolling, as the timeline slider moves left-to-right, the selected thumbnail moves left-to-right and top-to-bottom until the thumbnail in the bottom-right corner is reached. Next, the entire grid of thumbnails is replaced

with the thumbnails for the next set of associated frames. This mode maintains the most stable view of the thumbnails as the view only updates after the timeline slider has moved enough pixels to cover each of the visible frames in the grid (Figure 6). For example, with a 5x5 grid, the collection of thumbnails would only update every 5×5=25 pixels moved horizontally along the timeline.

Figure 6A shows how the set of visible thumbnails changes as the playhead moves across the timeline. For timestamps *t* and *t+1*, the set of thumbnails remains the same, and once the playhead moves out of the current range (*t+2*), a new grid of thumbnails is displayed.

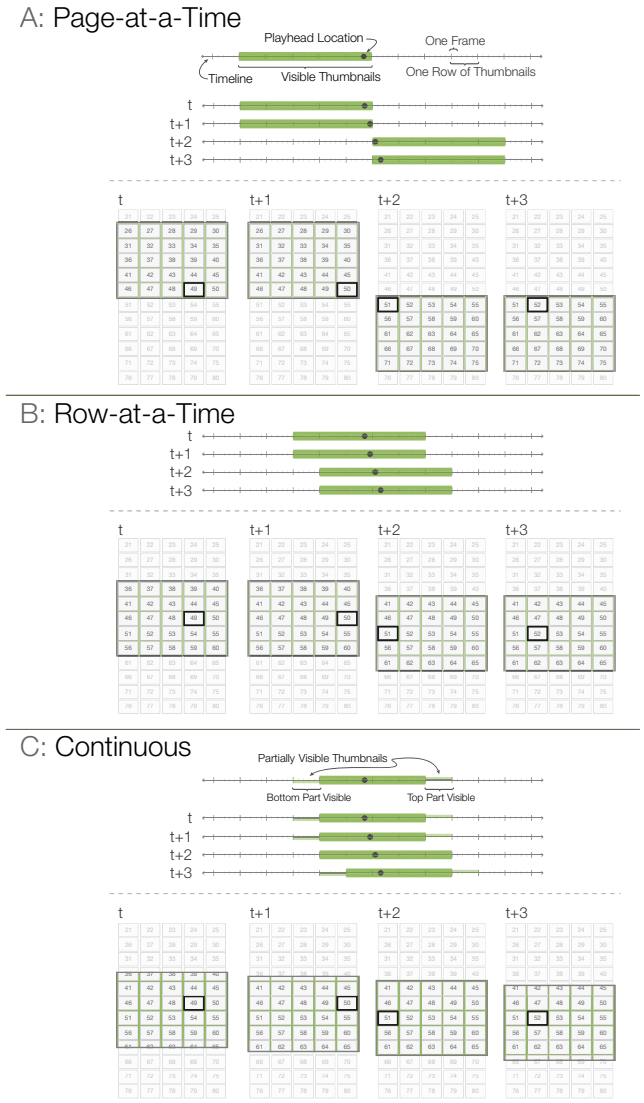

Figure 6. Position of visible thumbnails as the playhead location updates in each of the *scrolling techniques.*

Row-at-a-Time Scrolling

Row-at-a-time scrolling maintains a stable view while the selected thumbnail moves within a single row. Once the selection reaches the end of the row the grid moves up one row (Figure 6B). In this mode the set of thumbnails is updated once enough pixels are traversed to cover the number of columns, so with a 5x5 grid, the view updates every once for every 5 pixels moved horizontally.

Continuous Scrolling

Continuous scrolling moves the grid of thumbnails continuously up and down as the playhead location moves left and right. In this mode the view updates slightly with every horizontal movement along the timeline. Partial thumbnails can be visible at the top and bottom of the grid (Figure 6C).

Selecting a Thumbnail

Once the target scene has been located the *Swifter* technique provides two options for selecting the desired thumbnail.

Indirect Selection

The first method, which we refer to as *indirect* selection, is the standard video navigation technique. The user highlights the desired thumbnail by moving the playhead slider left and right along the timeline, and releases the mouse button to confirm the selection (Figure 5, Step 4a). This technique works well for large target areas, but for smaller targets, precisely positioning the slider is difficult.

Direct Selection

For smaller target scenes the user can choose to use the *direct* selection method. This is accomplished by moving the cursor vertically up into the thumbnail grid, placing the cursor directly over the desired thumbnail, and releasing the mouse button to confirm the selection (Figure 5, Step 4b). With this method the user is able to position the cursor over a relatively large thumbnail image to select a frame rather than relying on pixel-perfect accuracy on the timeline.

To prevent users from drifting into the *direct* selection mode area accidentally, direct selection mode is only entered if the cursor moves upwards into the grid area at an angle steeper than 45° (Figure 7). While the user is in *direct* selection mode the collection of thumbnails remain in a constant position and the playhead does not move along the timeline. *Indirect* selection mode is re-enabled when the cursor moves back into the timeline area.

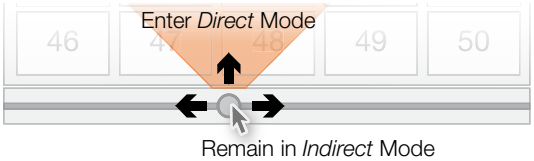

Figure 7. Transitioning from *indirect* to *direct* modes of frame selection.

Thumbnail Sizes

An important design consideration of Swifter is the size of the thumbnails to be displayed. Larger thumbnail images allow for more detail to be seen, but less images can be displayed in the grid. On the other hand, smaller thumbnail images allow for a denser grid of images to be displayed, at the expense of less detail on each individual image and perhaps an increased mental cost when trying to interpret so many images at once. The effects of the dimensions of the grid and thumbnail sizes are explored in Study Two.

Thumbnails Displayed During Scrubbing

Displaying more thumbnails on the screen makes it less likely that frames will be passed over and not displayed. Consider the previously described example, where the timeline has a pixel width of 854 pixels, the cursor moves at a constant speed over a period of 2 seconds, and the video player updates at a rate of 30 frames per second. If a 5x5 grid of thumbnails is used, each of the 854 thumbnail images will be displayed for at least 1/30th of a second, regardless of the scrolling technique. This is because the number of thumbnails displayed at once (25) is greater than the number of pixels moved along the timeline between successive redraws (854px/60updates = 14).

Figure 8. Frames displayed while moving over a 854 pixel wide timeline in 2 seconds with an update rate of 30fps with a *Swifter* grid of 5x5.

STUDY CONSIDERATIONS

Before describing a series a studies conducted to better understand the *Swifter* technique, we describe some important considerations for our evaluations.

Scene Ordering

Matejka et al. [16] introduced three main video ordering types within the context of video navigation: sequential, ordered, and random. The random condition is the most difficult because the user does not know if the target scene is before, or after, their current location. We chose to focus on this *random* scene ordering type because it is in this situation where improved video navigation could be the most helpful.

Latency

The *Swifter* technique is designed to work in high-latency situations, and like Swift, can overcome the problem of streaming latency by quickly downloading a collection of low-resolution thumbnails to be used while scrubbing. Thus, while the user is scrubbing through the timeline, the network latency is not an issue. Latency only becomes an issue when the user has selected their target destination and the full video needs to buffer and start playing from that location. We are exploring the video searching/scene selection part of this process, so the amount of latency present in the system does not factor into our studies.

Video and Timeline Size

The prototype video player used for the study was sized to an overall size of 854 pixels wide by 504 pixels tall, with 40 vertical pixels dedicated to the timeline slider. This size was chosen to match the size of the "medium" YouTube video player. Since each horizontal pixel on the timeline allows for the selection of one frame, this allows for the selection of 854 unique frames from the timeline.

STUDY ONE

The primary purpose of the first study is to see if there are any performance or subjective preference differences between the three scrolling techniques we have described.

Participants and Apparatus

Twelve paid volunteer participants (2 female) between the ages of 20 and 63 (μ = 32) were recruited through an online classified ad posting. The study was conducted in a private office on a 3.16GHz quad-core desktop computer running Windows 7 64-bit Edition. The graphics card was an nVidia Quadro FX 5600 and was driving a 24" Dell LCD monitor at a resolution of 1920 by 1200.

Design

A repeated measures within-participant design was used with the independent variables being *scrolling technique* (*page-at-a-time, row-at-a-time, continuous*), *grid dimensions* (*5x5, 10x10*), and *scene length* (*3 frames, 19 frames*). The ordering for each of the independent variables was counterbalanced, and the fully crossed design resulted in 12 total conditions grouped first by *scene length*, followed by *scrolling technique*, and *grid dimensions*.

Each participant performed the study in one session lasting approximately one hour. Each user performed one practice trial, and then 8 timed trials per condition.

Video Player and Content

For this study, the movie used was *The Intouchables* (Figure 9), which serves as a reasonable representative feature length film, in terms of overall running time, and the count and visual uniqueness of the cinematic scenes.

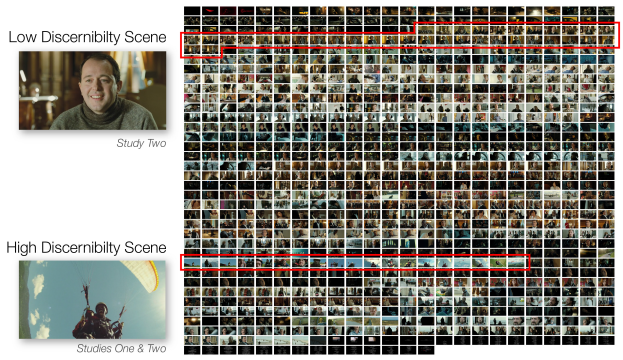

Figure 9. Thumbnail images from the movie *The Intouchables*, used for studies one and two. The *low* and *high discernibility* scenes are highlighted.

Target Scene Length and Location

With a running time of 112 minutes, each of the 854 locations on the timeline represent a time period of approximately 7.9 seconds. Converting the *scene length* conditions from number of thumbnail frames into time gives us approximately 24 seconds for the *3 frame* condition, and a 2 minute, 30 second long scene with the *19 frame* condition.

The target scene for this study was a "parachute scene" which had a visually distinguishable blue sky background (Figure 9). We chose a distinguishable scene, since this study was focused primarily on the scrolling mechanisms. This scene had 19 associated thumbnail frames.

To counter the learning bias of participants memorizing the movie, the target scene was removed and placed at a new location within the timeline. The video was divided into four equal sections, and the final location of the target scene was controlled to be placed within each of the quadrants for 2 trials per condition. In the *19 frame* condition, the entire target scene was inserted back into the timeline, while in the *3 frame* condition, only 3 thumbnails were re-inserted and a second generic scene was duplicated to bring the total number of thumbnails back to the original count of 854.

Procedure

The study began by viewing the target scene in a standard video player, and explaining that for every trial they would be searching for the same scene, but that it will have been removed from its original position in the timeline and inserted into a different random location each time.

The examiner then showed each participant the prototype video player and demonstrated both the *direct* and *indirect* frame selection techniques. Next, the participants were shown the behavior of each *scrolling technique* and performed four practice selection tasks with each scrolling style, and executed two selections with each of the *direct* and *indirect* selection styles. Users were told they could use either selection technique for any trial, and to use whichever one they felt would be quicker and more accurate for that particular trial.

Each trial began when the cursor was clicked on the timeline slider. Early pilot tests indicated that performing repetitive mouse drag operations with the left mouse button depressed became tiring, so participants were instructed to click once to begin the trial, and click again to make the frame selection. During the trial, the static beginning and end frames of the target scene were displayed beside the video player. Each trial ended once a frame within the target scene was successfully selected.

Results

Scrolling Technique

The primary dependent variable was *completion time* for each task. For *scrolling technique*, the *continuous* technique had the lowest mean completion time of 8.9 seconds and *row-at-a-*time had the highest at 10.0 seconds (Figure 10). However, repeated measure analysis of variance did not find the difference to be significant ($F_{1,11} = 0.415$, ns).

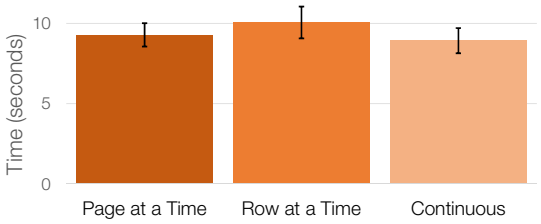

Figure 10. Scrolling techniques performance. (Note: error bars in all graphs report standard error)

In addition to the quantitative results, participants were also asked to rate each *scrolling technique* based on how much they liked the technique overall, ranging from *1*:*did not like*, to 7:*liked a lot*. The results here were also inconclusive (Figure 11) with 5, 4, and 3 participants preferring *row-at-a-time*, *continuous*, and *page-at-a-time* respectively.

Those who preferred *page-at-a-time* liked that the thumbnails stayed in a stable position for longer and that the grid was less likely to shift on them when they moved into the grid to perform a direct selection. Some users felt the *continuous* mode with its tight connection between the cursor position and grid movement made the overall technique easier to understand.

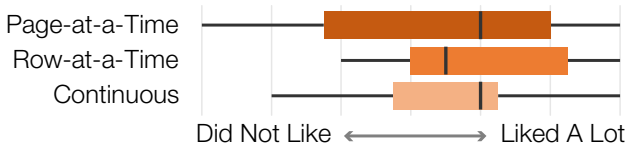

Figure 11. Subjective preference results from Study One.

For the next two studies we will use the *continuous* scrolling method since it may be most familiar with users, however our evidence indicates that any of the scrolling techniques would be viable candidates.

Scene Length and Grid Dimensions

There was a main effect for *scene length* ($F_{1,11} = 46.7$, $p < .0001$), with means of 7.2s for *19 frame* and 11.2s for *3 frame*. There was also a main effect for *grid dimensions* ($F_{1,11} = 49.5$, $p < .0001$) with means of 10.5 seconds for *5x5* and 7.9 seconds for *10x10*. Neither *scene length* nor *grid dimension* produced a significant interaction with *scrolling technique*. Subjectively, 9 of the 12 participants preferred the *10x10* grid size, while 2 users preferred *5x5* and 1 had no preference.

Selection Technique

Besides task completion time, we also looked at which *selection technique* (*indirect,* or *direct*) participants choose to use. One of the participants performed all tasks using *direct* selection, while the rest used a combination of the two techniques.

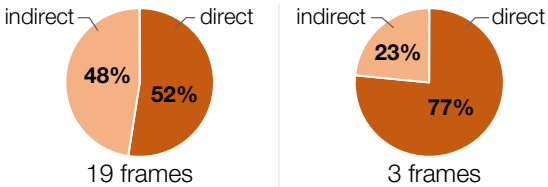

Figure 12. Selection technique usage based on *scene length*.

We had anticipated that for long *scene length* conditions users would be more likely to use *indirect* selection since the target on the timeline is relatively large in those cases, but when the *scene length* was short we expected it would be more advantageous to move the cursor into the thumbnail grid and select a frame from the target scene directly. This turned out to be true, with a ratio of *direct*:*indirect* selections in the *19 frames* condition of

nearly 1:1, while in the *3 frames* condition the ratio was approximately 3:1 (Figure 12). A paired t-test with an arcsin correction found the difference in the percentage of *direct* vs. *indirect* selections to be significant. ($t_{11} = 3.23$, $p < .01$). This data indicates both selection techniques are useful, so they will both be included in the remaining studies.

STUDY TWO

For the second study we wanted to more closely explore the effect of varying the *grid dimensions*, that is, the effect of varying the number and size of thumbnails displayed.

We hypothesize that in moving from a few, large thumbnails to many, small thumbnails the task performance curve will create a U-shape, where too few thumbnails will hamper performance because not enough information is being displayed at once, and too many thumbnails will degrade performance because the user will be overloaded with information and the thumbnails are each too small.

This optimal value should lie somewhere between, and may depend on several factors, but we hypothesize that a major factor will be the discernibility of the target scene. If a scene is very easy to recognize we believe it will be better to have a larger number of small thumbnail images. On the other hand, if a scene is difficult to recognize, then a smaller number of larger thumbnails might be better so that the user can see more details in each thumbnail image.

Participants and Apparatus

Eight paid volunteer participants (3 female) aged 18 to 37 ($\mu = 31$) were recruited from the same online community as used in the first study and was also conducted in the same location and on the same hardware as the first study.

Design

A repeated measures within-participant design as used with the independent variables being *discernibility* (*high, low*) and *grid dimensions* (*5x5, 6x6, 7x7, 8x8, 9x9, 10x10, 11x11, 12x12*) (Figure 13). The ordering of the *scene discernibility* factor was counterbalanced and the presentation order of *grid dimensions* was randomized. The *scene length* was fixed at 8 frames (~63 seconds) and the *scrolling type* was fixed to be *continuous*. Each participant performed 8 repetitions per condition, for a total of 128 trials per user. The study took approximately 1 hour to complete.

Discernibility

To test the impact of discernability, we chose two scenes from *The Intouchables*. For *high discernibility* we used the "parachute scene" from study one. The *low discernibility* scene contained several people talking to each other in a particular room, and was much more visually consistent with the other scenes in the film (Figure 9).

Procedure

The trials were ordered so each participant did all trials for one *discernibility* level first, before moving to the other scene. As in the first study the participant was shown the scene they would be looking for in a standard video player and then the examiner explained how the *Swifter* technique works. Participants then spent two minutes becoming accustomed to the software and performing practice trials finding the target scene before beginning the timed trials. Once the first *discernibility* condition was done, the participants were shown the new scene they would be searching for, performed several practice trials, and then continued on with the timed trials.

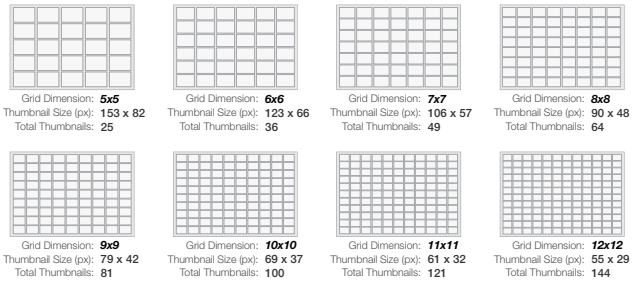

Figure 13. The *grid dimension* conditions in Study Two.

Results

As in the first study, the primary dependent variable was *completion time* for each task. Repeated measure analysis of variance showed a main effect for *discernibility* ($F_{1,7} = 440.5$, $p < .0001$). The effect of *grid dimensions* and its interaction with *discernibility* was not significant. However, visual inspection of the *completion time* graph (Figure 14) does indicate the existence of a U-shaped relationship. In the *low discernibility* condition the performance curve reached a minimum around the *7x7* or *8x8* grid size, and the *high discernibility* performance curve is minimized somewhere between *9x9* and *11x11*.

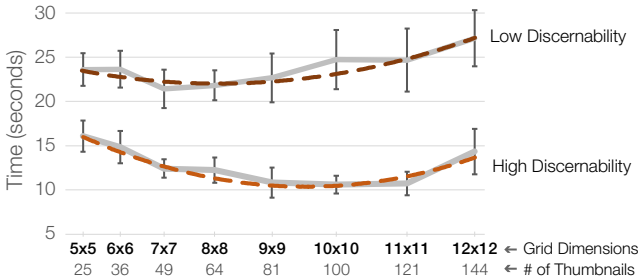

Figure 14. Task performance for High and Low Discernibility scenes with various grid sizes. The dashed lines represent the quadratic best-fit curves.

Participants also rated the 8 *grid dimensions* for each of the *discernibility* conditions on a scale from *1:liked least* to *5:liked most*, and were instructed to have at least one *grid dimension* scored a *1*, and at least one scored a *5* (Figure 15). For the *low discernibility* scene, the subjective results line up well with the timing data – the *7x7* grid was well received, and the high dimensionality grids (*10x10, 11x11, 12x12*) which performed poorly, were subjectively disliked.

The *high discernibility* subjective results saw positive ratings through the middle dimensions (*7x7, 8x8, 9x9, 10x10*). While the *11x11* grid had the fastest average *completion time*, it was scored relatively poorly in the

subjective ratings, indicating the small viewing size of those thumbnails were not comfortable to users.

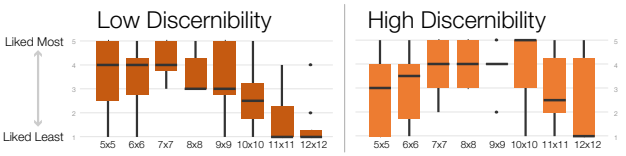

Figure 15. Subjective preference results for the *grid dimensions* for each of the *discernibility* conditions.

Users reported liking that they could see more thumbnails in the higher dimension grids, but found visually searching through the smaller thumbnail images more difficult. Ultimately, for different types of videos and target scenes, various *grid dimensions* could be the most appropriate, and it should perhaps be an option that users can set. However, we believe the *8x8* grid is a good starting point combining fast performance with high subjective ratings, and we will use this *grid dimension* for Study Three.

STUDY THREE

In the final study, we wanted to test the *Swifter* technique against existing online video navigation techniques designed to combat the effects of latency.

Techniques

The four tested techniques (Figure 16) all used the same collection of 854 thumbnails with a resolution of 134×72 pixels scaled as necessary to fit the required size.

Small Thumbnail: The small thumbnail technique (*Small*) is based on the behavior of the Netflix video player. A thumbnail image is displayed above the playhead location while dragging. We used a thumbnail dimension of 240×130 pixels which corresponds to the thumbnail size used by Netflix player.

Swift: The Swift technique (*Swift*) [16] displays the same information as the *small* condition, but enlarges the thumbnail image to fill the entire display.

Row of Thumbnails: The row-of-thumbnails technique (*Row*) is based on the behavior found at YouTube when dragging on the timeline slider. The selected frame is shown in the middle, while the next three, and previous three frames are displayed with smaller thumbnails on either side.

The precision selection mode from the YouTube hover interaction for long videos (similar in nature to the AV-ZoomSlider [13]) was piloted and resulted in slower task completion times, and as such, was not included.

Swifter: Based on the results of the first two studies, the *Swifter* technique (*Swifter*) was implemented using *continuous* scrolling, with an *8x8* grid dimension.

Participants and Apparatus

Sixteen paid volunteers (8 female) aged 20 to 50 (μ = 32) were recruited from the same online community as used in the first two studies. Participants reported using a computer between 2 and 14 hours per day (μ = 7) and watching between 7 and 300 online videos per month (μ = 42).

Design

A repeated measures within participant design was used with the independent variables being *technique* (*Small, Swift, Row, Swifter*), scene *discernibility* (*high, low*), and *scene length* (*5 frames, 19 frames*). The presentation order of all three independent variables were counterbalanced and a fully crossed design resulted in 16 conditions which were grouped first by *discernibility*, followed by *scene length* and *technique*. In addition to 4 practice trials with each *technique*, each user performed each condition 8 times resulting in a total of 16×16×8 = 2048 timed trials which took approximately one hour to complete. Trials were again equally distributed among the four quarters of the timeline.

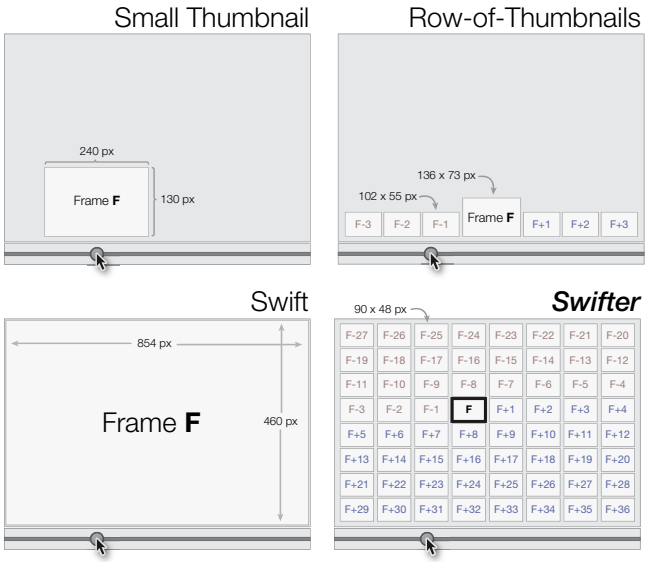

Figure 16. Pictographic representation of each of the techniques. 'F' represents the active frame, while the red and blue labels indicate the offset of frames before and after the selection.

Video Content

To avoid the risk that the *Swift* technique was over-fit to the video used in the first two studies, a different movie, *The Adjustment Bureau*, was used for Study Three (Figure 17). Overall, the scenes are more uniform in color, and less saturated than the movie used in the previous studies.

The closing credits were chosen as the *high discernibility* scene. The *low discernability* scene was an "apartment scene", which had a man and a woman talking in front of a floral-patterned headboard.

Procedure

As in the second study, the trials were ordered so each participant did all trials for one *discernibility* level first, before switching to the other scene, and before each *discernibility* condition began the users were shown the full motion video for the scene they would be searching for. In order to first experience the techniques without the additional difficulty of searching for a scene, the users

performed four warm-up trials with each technique, with a fabricated target scene composed of bright red frames.

During the study, and on the post-study questionnaire, the Swift technique was referred to as "*large thumbnail*", and *Swifter* was called "*grid of thumbnails*".

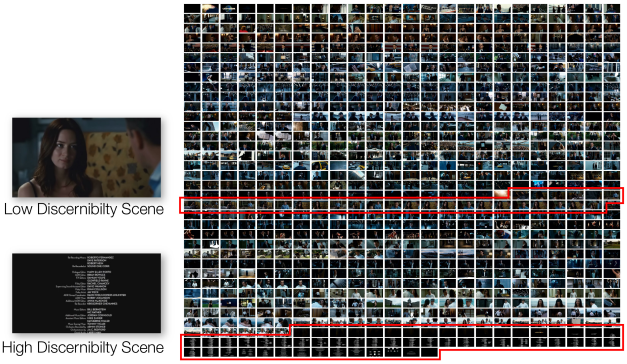

Figure 17. Thumbnail images from *The Adjustment Bureau*, used in the third study. The *high* and *low discernibility* scenes are highlighted.

Results

The primary dependent variable was *completion time*. Overall completion times were 8.79s for *Swifter*, 14.03s for *Row*, 14.04s for *Swift*, and 13.41s for *Small*. Repeated measure ANOVA showed a main effect for *technique* ($F_{3,45}$ = 32.1, p < .0001), *discernibility* ($F_{1,15}$ = 36.9, p < .0001), and *scene length* ($F_{1,15}$ = 102.5, p < .0001). Additionally, interaction effects were found for *discernibility*×*technique* ($F_{3,45}$ = 11.7, p < .001) and *scene length*×*technique* ($F_{3,45}$ = 10.1, p < .001). These results suggest that as tasks become more difficult (i.e., lower in discernibility or shorter in length) completion times for *Swifter* increase at slower rate than with the other techniques (Figure 18).

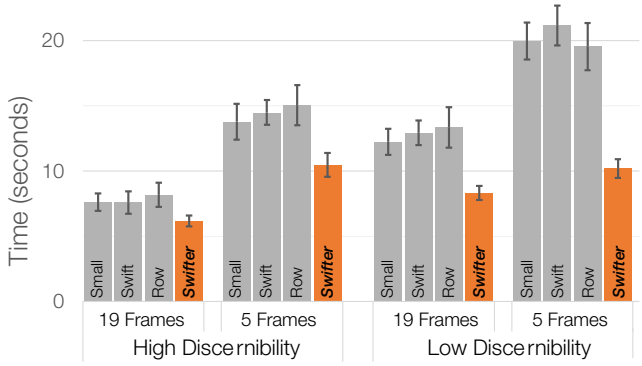

Figure 18. *Completion time* results from the third study.

Within each *discernibility*/*scene length* combination, post-hoc pairwise comparison with Bonferroni correction showed that *Swifter* was significantly faster than the other techniques (p < .05) except in the *high discernibility*/*19 frames* condition. There were no significant differences among the other three techniques. In the *low discernibility*/*5 frames* condition, *Swifter* was faster than the next fastest technique, *Row*, by 48%. Error rates were between 4.3% and 6.2% for all techniques, but did not reach significance at the .05 level.

Users were also asked to rate each technique on a scale between *1:did not like* to *7:liked a lot*, based on their overall assessment across all tasks (Figure 19). In general the techniques which display multiple thumbnails at once (*Swifter* and *Row*) did better in the subjective rankings than the single thumbnail techniques.

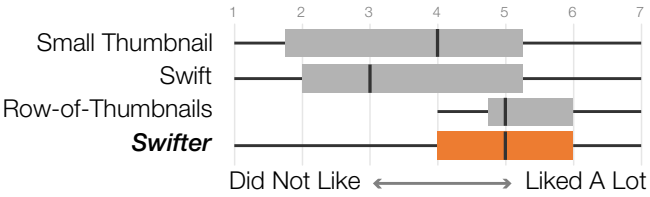

Figure 19. Overall subjective feedback on the *techniques* used in Study Three.

Additionally, users were asked which technique they liked the most for each *discernibility*/*scene length* combination. In those results we can see that *Swifter* was the most preferred in each condition, but each of the other *techniques* was preferred by at least two users for each *discernibility*/*scene length* condition (Figure 20).

Participants generally thought that *Swifter* made it easier to find the scene they were looking for than the other techniques, and also appreciated the ability to directly select a frame from the thumbnail grid. Users who preferred the other techniques felt that with less information on the screen they could better focus on the searching task, and some found focusing on a smaller number of thumbnail images prevented them from being distracted.

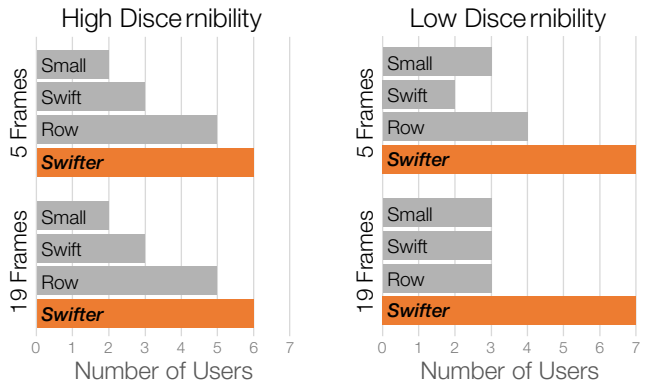

Figure 20. Results for the question *"Which technique did you like the best"* for each *discernibility*/*scene length* pair.

DISCUSSION AND FUTURE WORK

For a real-world deployment in an online video player, we would suggest a "small thumbnail" preview when hovering over the timeline (much like YouTube and Netflix do now) which provides a useful way to casually explore a video without pausing playback. Once the user clicked on the timeline, the *Swifter* technique would be activated.

Swifter has been designed to address the issues which arise when navigating a feature length film using a streaming

video player. We hope to conduct further studies to see how the technique performs on shorter length videos and videos with other types of content [20]. We would imagine that in a deployment, Swifter could be enabled only for videos of a certain threshold length, and one of the other techniques from Study Three could be used for shorter videos. It would also be useful to gather statistics to see how often various navigation tasks are performed in normal usage scenarios.

Our first study found that three different scrolling techniques produced similar quantitative and qualitative results. We believe the *continuous* scrolling method may be slightly easier for new users to understand, but each of the techniques are viable candidates. In an actual deployment, a widget on the player could be provided to allow users to switch to their preferred scrolling technique.

Similarly, our second study indicated that the best thumbnail size will be task dependent. A widget could also be provided to adjust the grid size, or alternatively, the mouse wheel could be used to adjust the thumbnail sizes. Given the prominence of mobile devices it would also be interesting to adapt the swifter technique for touch input. A future study with that input modality might be interesting.

With the *indirect* selection mode from the timeline, *Swifter* is limited to accessing only as many frames as the timeline is pixels wide. However, with the *direct* selection mode we could access many more frames by associating multiple thumbnail images with each location on the timeline, and it would be interesting to see what effect including additional thumbnail frames has on navigation tasks.

Another line of future work is to investigate how scene detection algorithms could enhance the thumbnail images used for Swifter. We used a collection of 854 thumbnail images spread evenly over the duration of the video. We experimented with instead using thumbnails generated from scene detection algorithms, but the experience seemed equivalent, and the number of produced thumbnails could not be controlled. We also prototyped displaying a partial grid of thumbnails (ex. 3 rows by 8 columns), animating the transition between page views, and various algorithms for switching between indirect and direct selection mode. Future explorations into these other dimensions of the design space could lead to interesting findings.

CONCLUSION

We have introduced a new interaction technique for navigating videos in high-latency situations. Results of our study found that *Swifter* was significantly faster than existing research and commercial techniques in a scene locating task, by a factor of up to 48%. Given the recent increased popularity of video streaming, we believe our work represents a timely and useful contribution.

REFERENCES

1. Alexander, J., A. Cockburn, S. Fitchett, C. Gutwin and S. Greenberg (2009). Revisiting read wear: analysis, design, and evaluation of a footprints scrollbar, *ACM CHI*. 1665-1674.
2. Bachmann, T. (1991). Identification of spatially queatized tachistoscopic images of faces: How many pixels does it take to carry identity? *Eur. Journal of Cog. Psychology.* 3:85-103.
3. Bailer, W., Schober, C., and Thallinger, G. (2006). Video Content Browsing Based on Iterative Feature Clustering for Rushes Exploration. *TRECVID Workshop*. 230-239.
4. Barnes, C., Goldman, D.B., Shechtman, E., Finkelstein, A. (2010). Video tapestries with continuous temporal zoom. *ACM SIGGRAPH*. Article 89.
5. Byrd, D. (1999). A scrollbar-based visualization for document navigation, *ACM Digital libraries*. 122-129.
6. Chang, L., Yang, Y., and Hua, X.S. (2008). Smart Video Player. *IEEE Multimedia and Expo*. 1605-1606.
7. Chen, L., Chen, G.C., Xu, C.Z., March, J., and Benford, S. (2008). EmoPlayer: A Media Player for Video Clips with Affective Annotations. *Int. with Comp*. 20:17-28.
8. Cockburn, A., Gutwin, C., and Alexander, J. (2006). Faster document navigation with space-filling thumbnails. *ACM UIST*. 1-10.
9. Divakaran, A. and Forlines, C. and Lanning, T. and Shipman, S. and Wittenburg, K. (2005). Augmenting Fast-Forward and Rewind for Personal Digital Video Recorders. *ICCE*. 43-44.
10. Doulamis, A.D. and Doulamis, N.D. (2004). Optimal Content-based Video Decomposition for Interactive Video Navigation. *IEEE CSVT*. 757-775.
11. Grossman, T., Matejka, J., and Fitzmaurice, G. (2010). Chronicle: Capture, Exploration, and Playback of Document Workflow Histories. *UIST*. 143-152.
12. Holthe, O. and Ronningen, L.A. (2006). Video Browsing Techniques for Web Interfaces. *IEEE CCNC*. 1224-1228.
13. Hürst, W. (2006). Interactive Audio-Visual Video Browsing. *ACM MM*. 675-678.
14. Hürst, W. and Darzentas, D. (2012). Quantity versus quality: The role of layout and interaction complexity in thumbnail-based video retrieval interfaces. *ICMR*. 45.
15. Janin, A., Gottlieb, L., and Friedland, G. (2010). Joke-o-Mat HD: Browsing Sitcoms with Human Derived Transcripts. *ACM MM*. 1591-1594.
16. Matejka, J., Grossman, T., and Fitzmaurice G. (2012). Swift: Reducing the Effects of Latency in Online Video Scrubbing. *ACM CHI*. 637-646.
17. Pongnumkul, S., Wang, J., Ramos, G., and Cohen, M. (2010). Content-Aware dynamic timeline for video browsing. *ACM UIST*. 139-142.
18. Ramos, G. and Balakrishnan, R. (2003). Fluid Interaction Techniques for the Control and Annotation of Digital Video. *ACM UIST*. 105-114.
19. Schoeffmann, K., Taschwer, M., and Boeszoermenyi, L. (2010). The Video Explorer – A Tool for Navigation and Searching within a Single Video based on Fast Content Analysis. *ACM SIGMM*. 247-258.
20. Torralba, A., Fregus, R., and Freeman, W. T. (2006) 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Trans. PAMI*. 11, p. 1958-1970.
21. Truong, B.T. and Venkatesh, S. (2007). Video Abstraction: A Systematic Review and Classification. *ACM TOMCCAP*. Appl. 3, 1, Article 3.
22. Tsukayama, H. (2012). YouTube: The future of entertainment is on the Web. *The Washington Post*.