Data-Driven Interaction Techniques for Improving Navigation of Educational Videos

Juho Kim[1] **Philip J. Guo**[1,2] **Carrie J. Cai**[1] **Shang-Wen (Daniel) Li**[1]
Krzysztof Z. Gajos[3] **Robert C. Miller**[1]

[1]MIT CSAIL
Cambridge, MA USA
{juhokim, cjcai, swli, rcm}@mit.edu

[2]University of Rochester
Rochester, NY USA
pg@cs.rochester.edu

[3]Harvard SEAS
Cambridge, MA USA
kgajos@eecs.harvard.edu

ABSTRACT

With an unprecedented scale of learners watching educational videos on online platforms such as MOOCs and YouTube, there is an opportunity to incorporate data generated from their interactions into the design of novel video interaction techniques. Interaction data has the potential to help not only instructors to improve their videos, but also to enrich the learning experience of educational video watchers. This paper explores the design space of data-driven interaction techniques for educational video navigation. We introduce a set of techniques that augment existing video interface widgets, including: a 2D video timeline with an embedded visualization of collective navigation traces; dynamic and non-linear timeline scrubbing; data-enhanced transcript search and keyword summary; automatic display of relevant still frames next to the video; and a visual summary representing points with high learner activity. To evaluate the feasibility of the techniques, we ran a laboratory user study with simulated learning tasks. Participants rated watching lecture videos with interaction data to be efficient and useful in completing the tasks. However, no significant differences were found in task performance, suggesting that interaction data may not always align with moment-by-moment information needs during the tasks.

Author Keywords

Video learning; Interaction peaks; Video summarization; MOOCs; Multimedia learning; Video content analysis.

ACM Classification Keywords

H.5.1. Multimedia Information Systems: Video

INTRODUCTION

Millions of people watch free educational videos online on platforms such as Khan Academy, Coursera, edX, Udacity, MIT OpenCourseWare, and YouTube. For example, the "Education" channel on YouTube currently has over 10.5 million subscribers, and a typical MOOC has thousands of video-watching learners. In addition, learners also take paid video-centric courses on commercial platforms such as Lynda, Udemy, and numerous university e-learning initiatives.

The server logs of these platforms contain fine-grained, second-by-second data of learners' interactions with videos, which we refer to as *interaction traces*. This data is now being used for real-time analytics to optimize business metrics such as viewer engagement time. Researchers have also used this data to perform retrospective empirical analyses. For example, video analytics studies on MOOCs have compared the effects of video production methods on learner engagement [13] and identified common causes of peaks in learner activity within videos [20].

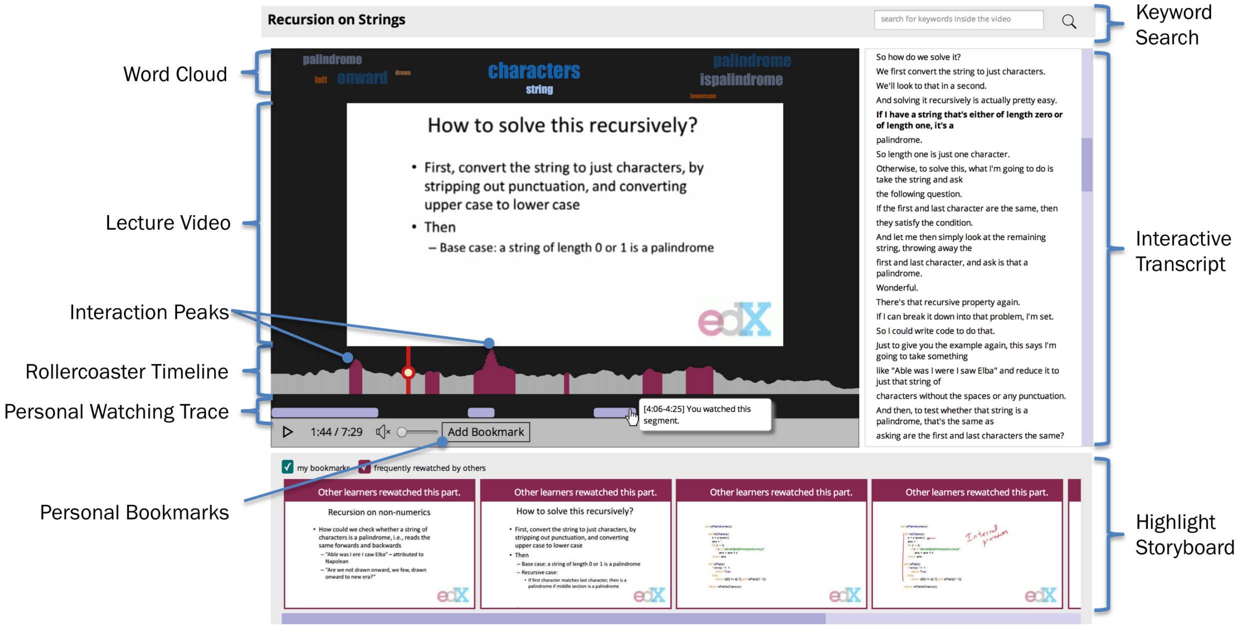
Interaction data provides a unique opportunity to understand collective video watching patterns, which might indicate points of learner interest, confusion, or boredom in videos. However, to our knowledge, researchers have not yet attempted to feed these patterns back into the video navigation interface to support learners. While learners might have diverse goals in navigating through a video, existing video interfaces do not provide customized navigation support beyond scrubbing on a linear timeline slider with thumbnail previews and synchronizing with a textual transcript. Adapting to collective video watching patterns can lead to richer social navigation support [10].

This paper explores the design space of navigation techniques for educational videos that leverage interaction data. We introduce novel data-driven interaction techniques that process, visualize, and summarize interaction data generated by many learners watching the same video. For instance, in a typical MOOC, at least a few thousand learners watch each video. Based on prior findings about learner intent and typical formats of educational videos [13, 20], we have designed these techniques to support fluid and diverse video navigation patterns. Typical video watching scenarios include:

- Rewatch: "Although I understand the high-level motivation, I didn't quite get the formal definition of 'admissible heuristic' the first time I watched this lecture. So I want to rewatch the section explaining the formal definition."
- Textual search: "I want to jump to where the instructor first mentioned the phrase 'alpha-beta pruning.'"

UIST'14, October 5–8, 2014, Honolulu, HI, USA.

ACM 978-1-4503-3069-5/14/10 $15.00.
http://dx.doi.org/10.1145/2642918.2647389



How to solve this recursively?

- First, convert the string to just characters, by stripping out punctuation, and converting upper case to lower case
- Then
  - Base case: a string of length 0 or 1 is a palindrome

[4:06-4:25] You watched this segment.

So how do we solve it?
We first convert the string to just characters.
We'll look to that in a second.
And solving it recursively is actually pretty easy.
If I have a string that's either of length zero or of length one, it's a
palindrome.
So length one is just one character.
Otherwise, to solve this, what I'm going to do is take the string and ask
the following question.
If the first and last character are the same, then
they satisfy the condition.
And let me then simply look at the remaining string, throwing away the
first and last character, and ask is that a palindrome.
Wonderful.
There's that recursive property again.
If I can break it down into that problem, I'm set.
So I could write code to do that.
Just to give you the example again, this says I'm going to take something
like "Able was I were I saw Elba" and reduce it to just that string of
characters without the spaces or any punctuation.
And then, to test whether that string is a palindrome, that's the same as
asking are the first and last characters the same?

Recursion on non-numerics

- How could we check whether a string of characters is a palindrome, i.e., reads the same forwards and backwards
  - "Able was I ere I saw Elba" – attributed to Napolean
  - "Are we not drawn onward, we few, drawn onward to new era?"

How to solve this recursively?

- First, convert the string to just characters, by stripping out punctuation, and converting upper case to lower case
- Then
  - Base case: a string of length 0 or 1 is a palindrome
  - Recursive case:
    - If first character matches last character, then is a palindrome if middle section is a palindrome

Figure 1. This paper presents three sets of novel interaction techniques to improve navigation of educational videos. 1) Dynamic timelines (Rollercoaster Timeline, Interaction Peaks, and Personal Watching Trace), 2) Enhanced in-video search (Keyword Search and Interactive Transcript), 3) Highlights (Word Cloud, Personal Bookmarks, Highlight Storyboard). All techniques are powered by interaction data aggregated over all video watchers.

- Visual search: "I remember seeing this code example in a diagram somewhere in the video. I want to find it again."
- Return: "Hey, that was annoying! I don't want to see the instructor's talking head. I'm not done looking at this PowerPoint slide yet. I want the slide back!"
- Skim: "This lecture seems somewhat trivial. I'll skim to see if there's something I probably shouldn't miss."

Specifically, we developed interaction techniques to augment a traditional Web video player with 1) a Rollercoaster timeline that expands the video timeline to a 2D space, visualizes collective interaction traces of all learners, and dynamically applies non-linear scrolling to emphasize interaction peaks, 2) enhanced in-video search that visualizes and ranks occurrences on the timeline and recommends salient keywords for each video section, and 3) a video summarization method that captures frames that are frequently viewed by other learners. These techniques combine learners' collective interaction traces with text and visual content analysis.

We package all of these techniques together in LectureScape, a prototype Web-based video interface shown in Figure 1. In a laboratory study with simulated search and skimming tasks, we observed that participants employ diverse video navigation patterns enabled by our techniques. Specifically, they noted that LectureScape helped them to quickly scan the video and efficiently narrow down to parts to direct their focus. They also found interaction data to be useful in identifying important or confusing pedagogical points within videos. However, no significant differences were found in task performance, suggesting that interaction data may not always align with moment-by-moment information needs participants had for the study tasks.

This paper makes the following contributions:

- a conceptual design approach for interaction techniques that leverages information about other learners' behavior to improve the video learning experience,
- a set of novel video interaction techniques powered by real log data from learners in a MOOC platform, introducing 1) a 2D, non-linear timeline, 2) enhanced in-video search, and 3) a data-driven video summarization method,
- and an empirical evaluation of the techniques with learners, which enabled fluid and diverse video navigation.

RELATED WORK

We review previous research in leveraging interaction history to improve user interfaces and video navigation.

Leveraging Interaction History

There is a rich thread of research in using interaction history data to analyze usage patterns and improve users' task performance. Interaction history data is automatically collected by applications during normal usage. Examples include Web browsers logging Web page visit history, search engines capturing query history, and video players storing video interaction clickstreams such as play and pause events. Read Wear [14] presented a visionary idea in this space to visualize users' read and edit history data in the scrollbar. Chronicle [12] captured and provided playback for rich, contextual user interaction history inside a graphical application. Dirty Desktops [16] applied magnetic forces to each interaction trace, which improved target selection for commonly used widgets. Patina [26] separated individual and collective history and added overlays on top of the GUI, to help people find commonly used menu items and discover new ways of completing desktop-related tasks. Causality [30] introduced an application-independent conceptual model for working with interaction history. This paper uses video interaction history to support common navigation tasks in video-based learning.

To model user interest in video watching, researchers have proposed features such as viewership [38], scrubbing [39], zooming and panning [7], and replaying and skipping [9] activities. SocialSkip [9] applied signal processing to replaying activity data in order to infer interesting video segments. Other researchers have used more explicit input from video watchers, including user ratings [33], annotations [38], and the "this part is important" button [36]. Most existing approaches introduce a modeling technique or data visualization. We take this data further to build new interaction techniques for video navigation, which prior work has not done. Also, we combine both implicit user history data and explicit user bookmarks to support diverse learning tasks, which extends prior work on supporting social navigation for lecture videos [28].

Video Navigation Techniques

To improve video navigation with interaction data, we designed novel techniques to 1) add richer interactions to the video timeline, 2) support enhanced in-video search, and 3) automatically summarize video content. We now review related work for each of the three techniques.

To improve video scrubbing, YouTube displays thumbnail previews for quickly skimming local frames, and Swift [25] overlays low-resolution thumbnails to avoid network latency delays. The content-aware timeline [34] extracts keyframes with content analysis and plays a video snippet around these points when the user scrubs the timeline. Elastic interfaces [24] use the rubber band analogy to control scrubbing speed and support precise navigation, as seen in the PV Slider [35] and Apple's iOS video interface. We extend this line of research by asking: "What if the scrubbing behavior adapts to learners' watching patterns, as collected from interaction history data?" To our knowledge, no video scrubbing technique has leveraged interaction history data.

Another thread of research introduced techniques to support navigation of how-to videos, a sub-genre of educational video that includes procedural, step-by-step instructions about completing a specific task. Existing systems reveal step-by-step structure by adding rich signals to the video timeline, such as tool usage and intermediate results in graphical applications [8, 21]. Classroom lecture videos tend to be less structured than how-to videos, which makes capturing clear structural signals harder. We instead turn to interaction data that is automatically logged for learners as they watch the video.

Popular GUI applications such as Web browsers and text editors have incremental search features where the scrollbar and text visually highlight locations of search term occurrences. Also, video players on educational platforms such as edX show a synchronized transcript alongside the currently playing video. Learners can search for text in the transcript and then click to jump to the corresponding spot in the video. We improve these interfaces by augmenting search results with interaction data and visualizing them on the video timeline.

Existing video summarization techniques use video content analysis to extract keyframes [3], shot boundaries [22], and visual saliency [15]. To provide an overview of the entire clip at a glance and support rapid navigation, recent research has used a grid layout to display pre-cached thumbnails [27], short snippets [18] in a single clip, personal watching history for multiple clips [1], a conceptual hierarchy visualization [19], or a 3D space-time cube display [31]. For educational lecture videos, Panopticon [18] has been shown to shorten task completion time in seeking information inside videos [32]. For blackboard-style lecture videos, NoteVideo [29] reverse-engineers a rendered video to create a summary image and support spatial and temporal navigation. This paper introduces a new summarization technique that uses *interaction peaks*, points in a video with significantly high play button click activity, to generate highlight frames of a clip.

DESIGN GOALS

This work focuses on supporting video navigation patterns common in online education, which differ from watching, say, a movie or TV show in a sequential, linear manner. Our designs are informed by quantitative and qualitative findings from analyses of educational videos, which suggest that learners often re-watch and find specific information from videos. Prior work in video clickstream analysis on four edX MOOCs found many *interaction peaks*, i.e., concentrated bursts in play/pause button clicks during certain segments of a video [20]. 70% of automatically-detected peaks coincided with visual transitions (e.g., switching between an instructor's head and a slide) and topic transitions [20]. A challenge in using interaction data to support learning is that the meaning of an interaction peak can be ambiguous (e.g., interest, confusion, or importance). In this paper, we do not assume a specific meaning behind interaction peaks, but do assume they are worth emphasizing regardless of the real cause. If a peak indicates importance, it would make sense to highlight it for future learners. If it indicates confusion, it may still make sense to emphasize so that learners would pay more attention.

To discover unsupported needs in lecture video navigation, we also conducted multiple rounds of feedback sessions with learners using our initial prototypes. The data analysis and interviews led to three high-level goals that informed our design of data-driven video interaction techniques.

Provide easy access to what other learners frequently watched. Our observations suggest that learners find it hard to identify and navigate to important parts of information-dense educational videos. To help a learner make more informed decisions about which part of the video to review, we leverage other learners' interaction traces, especially interaction peaks. We designed navigation techniques to emphasize these points of interest while the learner visually scans the video or physically scrubs the timeline.

Support both personal and collective video summaries. To prepare for homework assignments or exams, learners often take notes and watch videos multiple times to create a meaningful summary. Since there are often thousands of learners watching each video, we explore ways to present collective interaction traces as an alternative summary to complement each learner's personal summary. We extend prior work on social navigation in videos [28], history visualization, and revisitation mechanisms by supporting both manual bookmarking and automatic personal and collective watching traces.

Support diverse ways to search inside of a video. In our formative studies, learners described different ways they look for specific information inside a video. They would rely on both textual cues (e.g., topic and concept names) and visual cues (e.g., an image or a slide layout) to remember parts of the video. A more challenging case is when they cannot remember what the cue was for their particular information need. This observation inspired us to support both active search (e.g., when the learner has a clear search term), and ambient recommendations (e.g., when the learner does not know exactly what to search for). We designed techniques to enhance existing search mechanisms with interaction data, which provide social cues to serve both search scenarios.

DATA-DRIVEN VIDEO NAVIGATION TECHNIQUES

We introduce three interaction techniques to improve navigation of educational videos: an alternative timeline, search interface, and summarization method. Our main insight is to use the non-uniform distribution of learner activity within a video to better support common navigation patterns. Although the prototypes shown in this paper use videos on edX, a MOOC (Massive Open Online Course) platform, the techniques can be implemented for other video platforms such as YouTube because they use only standard Web technologies.

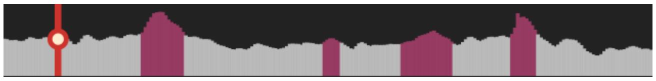


Figure 2. The 2D Rollercoaster timeline that appears below each video instead of a traditional 1D timeline. The height of the timeline at each point shows the amount of navigation activity by learners at that point. The magenta sections are automatically-detected interaction peaks.

The Rollercoaster Timeline: 2D, Non-Linear Timeline

To help learners identify and navigate to important parts of the video, we introduce the rollercoaster timeline. Unlike a traditional 1D timeline, the rollercoaster timeline is 2D with an embedded visualization of second-by-second learner interaction data (Figure 2). It visualizes the navigation frequency as a proxy of importance, as revealed by the behavior of other learners, and modifies the timeline scrubbing behavior to make precise navigation in important regions easier.

Navigation events are logged when the learner pauses and resumes the video, or navigates to a specific point. The Rollercoaster timeline uses navigation event counts as the vertical dimension. This visualization can also show other kinds of interaction events, including the number of viewers, rewatchers, unique viewers, or play or pause button clicks.

2D timeline

If the learner wants to jump to a specific point in the video, he can click on any point in the 2D timeline, which will capture the x coordinate of the click and update the playhead. The embedded peak visualization shows the intensity and range of each peak, and the overall distribution of the peaks within a video. Since interaction peaks are highlighted in magenta and span a wider region than other points, the learner can visually review and navigate to the commonly revisited parts in the video. We use the Twitinfo [23] peak detection algorithm to detect peaks in the server log data.

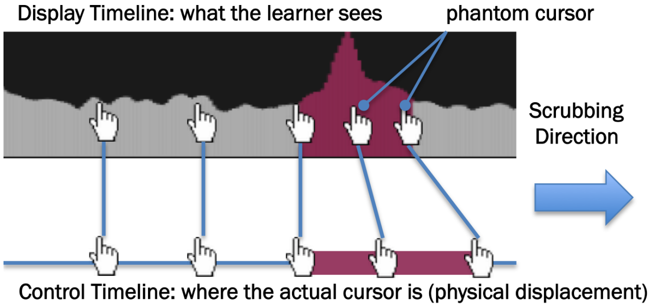

Display Timeline: what the learner sees
phantom cursor
Scrubbing Direction
Control Timeline: where the actual cursor is (physical displacement)

Figure 3. Non-linear scrubbing in the Rollercoaster timeline. To draw the learner's attention to content around interaction peaks, the phantom cursor decelerates scrubbing speed when the cursor enters a peak range.

Non-linear scrubbing with the phantom cursor

This timeline also enables dynamic, non-linear scrubbing, which takes advantage of interaction peaks. The basic idea is to apply friction while scrubbing around peaks, which leads to prolonged exposure so that learners can get a more comprehensive view of the frames near the peaks even when scrubbing quickly. Friction also makes it easier to precisely select specific frames within the range, since it lowers the frame update rate. It is an example of control-display ratio adaptation [5, 16, 37], dynamically changing the ratio between physical cursor movement and on-screen cursor movement.

Previous techniques have applied elastic, rubber band-like interactions to scrubbing [17, 24, 34, 35]. Our technique differs in that 1) it uses interaction data instead of content-driven keyframes, 2) elasticity is selectively applied to parts of the timeline, and 3) the playhead and the cursor are always synchronized, which reduced user confusion in our pilot studies.

When the mouse cursor enters a peak region while dragging, the dragging speed slows down relative to the dragging force, creating the sense of friction. The faster the dragging, the weaker the friction. We achieve this effect by temporarily hiding the real cursor, and replacing it with a *phantom cursor* that moves slower than the real cursor within peak ranges (Figure 3). The idea of enlarging the motor space around targets is inspired by Snap-and-Go [4].

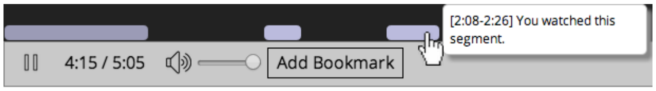

[2:08-2:26] You watched this segment.

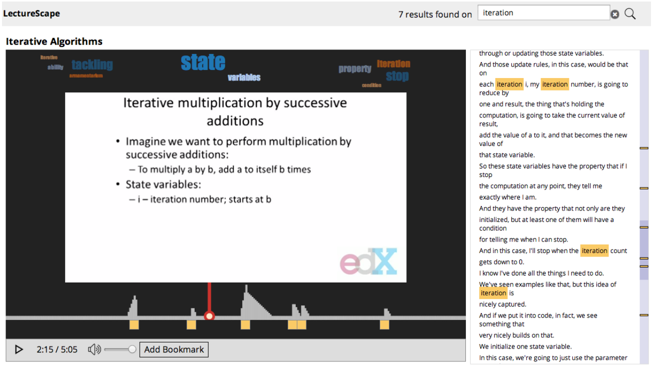
Figure 4. The real-time personal watching traces visualize segments of the video that the learner has watched.

Personal watching trace visualization

When we observed pilot study users navigating videos with our timeline, a common desire was to keep track of which parts of the video they personally watched, which might not align with the aggregate interaction peaks collected over all learners. Thus, we added another stream under the timeline to visualize each learner's personal watching traces. Previous research has separated personal and collective history traces to support GUI command selection [26], and added history indicators to a document scrollbar [2], which improved task performance in information finding. We extend these approaches to video navigation by using personal watching traces to support revisitation. Once the learner pauses the video or jumps to a new point, the current watching segment is visualized on a separate track below the timeline (Figure 4). Clicking on a generated segment replays the segment. More recent segments are displayed with higher opacity to further emphasize them over older ones. These traces can be stored on a per-user basis to help learners quickly find points of interest when they return to re-watch a video at a later date.

Keyword Search and Visualization

To better support searching for relevant information inside of a video, we use interaction data to power keyword search and transcript analysis. Instead of weighing all occurrences equally, our search technique rewards results in sections of the video where more learners watched. Since key concepts often appear dozens of times in a video, this feature helps the learner prioritize which parts of the video to review. Furthermore, to support novice learners who do not necessarily have the vocabulary to translate their information needs into a direct search query, we suggest major topics discussed in each section of the video in a word cloud. These topics serve as a keyword summary that can help learners recognize and remember the main topics discussed in each video.


LectureScape

7 results found on iteration

Iterative Algorithms

Iterative multiplication by successive additions

- Imagine we want to perform multiplication by successive additions:
  - To multiply a by b, add a to itself b times
- State variables:
  - i – iteration number; starts at b

through or updating those state variables.
And those update rules, in this case, would be that on
each iteration i, my iteration number, is going to reduce by
one and result, the thing that's holding the
computation, is going to take the current value of result,
add the value of a to it, and that becomes the new value of
that state variable.
So these state variables have the property that if I stop
the computation at any point, they tell me
exactly where I am.
And they have the property that not only are they
initialized, but at least one of them will have a condition
for telling me when I can stop.
And in this case, I'll stop when the iteration count
gets down to 0.
I know I've done all the things I need to do.
We've seen examples like that, but this idea of iteration is
nicely captured.
And if we put it into code, in fact, we see something that
very nicely builds on that.
We initialize one state variable.
In this case, we're going to just use the parameter

Figure 5. Our interaction data-driven keyword search brings in more context for the learner to decide where to focus on when searching for a keyword. For instance, the learner can visually check the distribution of when the lecturer said the keyword in the current video, which is useful in seeing where it was heavily discussed versus simply introduced.

Keyword search

If the learner has a keyword to search for, she can type it in the search field (top right in Figure 5), which searches over the full transcript of a video clip, and displays results both on the timeline and in the transcript. When the learner enters a search query, the timeline dynamically displays the search results instead of the default interaction visualization (see Figure 6). In this way, the timeline serves as a dynamic space for supporting different learner tasks by changing the peak points it emphasizes. Each result renders as a pyramid-shaped distribution, whose range is the duration of the sentence the word belongs to and whose peak is where the term is spoken. Figure 7 shows how hovering over the result displays a tooltip, and clicking on the result plays the video from the beginning of the sentence that includes the search term. This sentence-level playback provides the learner with more context surrounding the term.

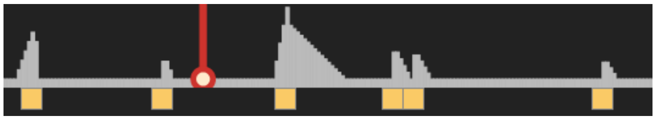


Figure 6. The search timeline appears below the video after each search. It visualizes the positions of search results, as well as the relative importance of each. Here the high peak in the middle indicates both that it contains the search term and that lots of learners watched that part.

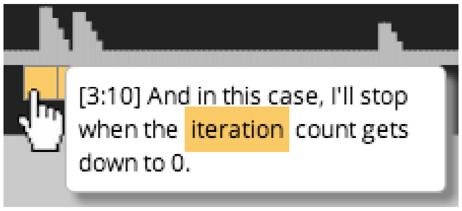

[3:10] And in this case, I'll stop when the iteration count gets down to 0.

Figure 7. Hovering on a search result displays a tooltip with the transcript sentence that contains the search term. Clicking on the result plays the video starting at the beginning of the sentence to assist the learner with comprehending the surrounding context.

Because key terms are repeated many times even during a short video, it can be hard for the learner to locate the most important search result. For example, in a 5-minute edX video on iterative algorithms, "variable" is mentioned 13 times, and "state" 12 times. We use interaction data to rank search results, with the assumption that parts of the video more frequently watched by previous learners are likely to reflect the current learner's interest. Our ranking algorithm analyzes learner activity around a search result and assigns a weight to the result, giving higher weights to sentences that were viewed by more learners. It then computes the relevance score by combining this weight with term frequency within the sentence. To support quick visual inspection of search results, we represent the computed score as the height on the timeline (gray peaks in Figure 6). If the term was mentioned multiple times in a sentence, we convolve the distributions for all occurrences and assign the maximum score to it.

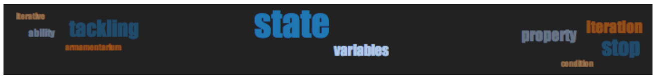


Figure 8. The word cloud displays automatically-extracted topics for the currently visible section of the video, providing a keyword-based summarization of the video. Clicking on any word triggers a search for it.

Word cloud: topic summarization and visualization

To address the low visual variation between frames in many videos and to help learners recognize and remember major topics in the clip, we use word clouds to dynamically display keywords in different segments of the video. We use TF-IDF (term frequency-inverse document frequency) scores for extracting keywords and weighing their importance. To compute the TF-IDF scores for the keywords in a transcript, we define a document as the transcription sentences between two consecutive interaction peaks, and the background corpus as the collection of all video transcripts in the entire course. This user activity-driven mechanism extracts self-contained segments from each video.

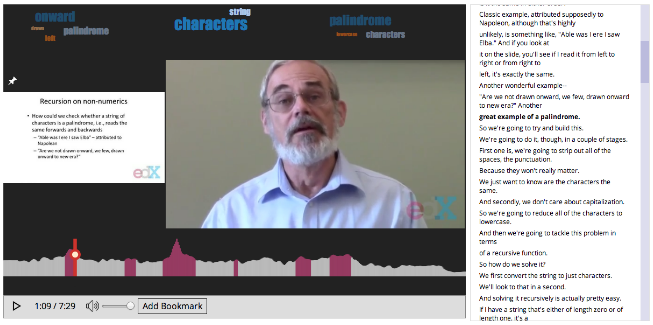
The visualization is positioned directly above the video as a panel consisting of three word clouds (see Figure 8), and gets updated at every interaction peak. The center cloud corresponds to the present segment being viewed, and two additional clouds represent the previous and upcoming segments, respectively. These displays are intended to give a sense of how lecture content in the current segment compares to that of surrounding segments. To bring focus to the current word cloud, we de-emphasize the previous and next word clouds by decreasing their opacity and word size relative to the current cloud. Clicking on any keyword in the cloud triggers a search using that term, visualizing the occurrences in the transcript as well as on the timeline.

Video Summarization with Highlights

To enable a quick overview of important points, we present a strip of visual highlights of selected video frames. Consistent with our design goal of providing access to both personal and collective interaction traces in the timeline, we support both collective and personal highlights. Collective highlights are captured by interaction peaks, while personal highlights are captured by the learner bookmarking a frame of interest.

Interaction peak highlights

We capture interaction peaks and provide one-click access to them to support common watching patterns such as jumping directly to these points. The storyboard-style display of the peak frames allows the learner to visually scan the video's progress (Figure 1). These highlights are visually-oriented, while the word cloud of Figure 8 is text-oriented.



Recursion on non-numerics

- How could we check whether a string of characters is a palindrome, i.e., reads the same forwards and backwards
  - "Able was I ere I saw Elba" – attributed to Napolean
  - "Are we not drawn onward, we few, drawn onward to new era?"

Classic example, attributed supposedly to Napoleon, although that's highly

unlikely, is something like, "Able was I ere I saw Elba." And if you look at

it on the slide, you'll see if I read it from left to right or from right to

left, it's exactly the same.

Another wonderful example--

"Are we not drawn onward, we few, drawn onward to new era?" Another

great example of a palindrome.

So we're going to try and build this.

We're going to do it, though, in a couple of stages.

First one is, we're going to strip out all of the spaces, the punctuation.

Because they won't really matter.

We just want to know are the characters the same.

And secondly, we don't care about capitalization.

So we're going to reduce all of the characters to lowercase.

And then we're going to tackle this problem in terms

of a recursive function.

So how do we solve it?

We first convert the string to just characters.

We'll look to that in a second.

And solving it recursively is actually pretty easy.

If I have a string that's either of length zero or of length one, it's a

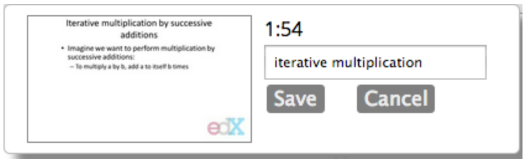
Figure 9. Our pinning algorithm analyzes interaction peaks and visual transitions in the video to display a smaller static frame (on the left) next to the video (on the right). Learners can manually pin any frame as well.

Pinning video frames

Most existing video players display only one frame at a time. This ubiquitous interface is sensible for the narrative structure of general-purpose videos such as TV shows, where a sequential flow is natural. However, educational videos are information-heavy, and active learning involves skimming and scrubbing [20]. For example, an instructor might verbally refer to the formula she described in the previous PowerPoint slide, but the formula might no longer be available on screen. A learner who wants to refer to that formula has to scrub the video timeline to go back to a frame with the relevant formula slide. To support watching patterns that are not easily supported by existing players, our video player pins a relevant frame next to the video stream for easy reference (Figure 9).

Our video player automatically determines a frame to pin. A relevant pinned frame should 1) not be identical to what is currently shown in the video, 2) include important content that is worth referencing, and 3) contain readable content such a textual slide or technical diagram, not merely a static frame of the instructor's face or students sitting in a classroom. Otherwise, juxtaposing a static frame next to the video might cause distraction and visual clutter. To meet these requirements, our pinning algorithm uses both interaction peaks and visual transitions. It automatically pins an interaction peak frame if there is a visual transition shortly after the peak. Checking for a visual transition ensures that the pinned frame is not visually identical to the frames right after the transition. Also, pinning an interaction peak frame ensures that the frame at least includes content viewed by many others.

The learner can also manually pin a frame by clicking the pin icon attached to each peak frame, which replaces the current pinned frame with the learner's. While the system attempts its best effort to show the most relevant frame at a given time, the learner also has the flexibility to control what gets displayed.



Iterative multiplication by successive additions

- Imagine we want to perform multiplication by successive additions:
  - To multiply a by b, add a to itself b times

1:54

iterative multiplication

Figure 10. The learner can add a labeled bookmark, which is added to the highlights stream below the video for visual preview and revisitation.

Personal bookmarks

While peak frames might be a reasonable summary of a video, individual learners might have summarization needs that are not captured by the collective traces. The learner can add personal bookmarks by clicking on the "Add Bookmark" button. The learner can see the captured frame at the point of click and add their own label for future reference (Figure 10). Once a bookmark is saved, it is added to the highlights stream, chronologically ordered along with other bookmarks and interaction peak highlights. This view allows the learner to choose between naturally-formed interaction peaks by other learners as well as self-generated bookmarks.

LECTURESCAPE: WEB-BASED PROTOTYPE

This section introduces LectureScape, a prototype lecture video player that combines all of the techniques in a unified interface (see Figure 1). The main goal of LectureScape is to give learners more control and flexibility in deciding how to navigate educational videos. LectureScape features the video player in the main view, along with the Rollercoaster timeline below the video and the word cloud above it. The interactive transcript is in the right sidebar, and the highlights are positioned at the bottom of the screen. The search box at the top enables keyword search. The widgets are all collapsible to reduce visual clutter and to hide unused features.

We implemented LectureScape using standard Web technologies: HTML5, CSS3, and JavaScript. The word cloud and Rollercoaster timeline are rendered with D3 [6]. Our customized HTML5 video player does not require additional re-encoding or streaming, and is independent of the encoding or streaming method used. It only requires interaction data for each video to activate the data-driven interaction techniques. Our data processing pipeline formats interaction traces and generates visual assets from an existing video.

First, three data streams are stored for each video: interaction data, TF-IDF results on the transcript, and visual transition data. Our peak detection algorithm runs on page load, which allows dynamic parameter tuning. Peak detection is run on both the interaction data and visual transition data, which returns interaction peaks and shot boundaries, respectively.

Second, the system generates thumbnail images for each second of a video. Because many of our interactions require displaying a thumbnail of a video frame on demand, low latency in image loading is crucial in supporting seamless interactions. It is especially important for educational videos whose visual changes are more subtle than in movies or TV shows, and whose on-screen information matters for learners to read and comprehend. Upon a page load, the system preloads all thumbnails for a video. When a learner drags the timeline, instead of loading the video each time a dragging event is triggered, our player pauses the video and displays an overlay screenshot. This results in much less latency and smoother dragging, similar to the benefits reported by Swift [25].

EVALUATION

To assess the feasibility of using interaction data to enhance video navigation, we conducted a user study comparing video players with and without our data-driven interaction techniques. We explored three research questions:

- **RQ1.** How do learners navigate lecture videos with LectureScape in typical kinds of learning tasks such as search and summarization?
- **RQ2.** How do learners interpret interaction data presented in LectureScape?
- **RQ3.** Are LectureScape's features useful and learnable?

Study Design

The study was a within-subjects design, where each learner used both LectureScape and a baseline interface that stripped off all interaction data-related features from LectureScape. The baseline interface still included the interactive transcript and preview thumbnails on hover, to emulate what is available in platforms such as edX or YouTube. To maintain uniformity in look and feel for our comparative study, the baseline interface had the same layout and visual design as LectureScape.

Learners performed three types of learning tasks for lecture videos: visual search, problem search, and summarization. These tasks represent realistic video watching scenarios from our observations, and match common evaluation tasks used in the literature on video navigation interfaces [11, 32].

- **Visual search** tasks involved finding a specific piece of visual content in a video. These tasks emulated situations when a learner remembers something visually and wants to find where it appeared in a video. For example, for a video about tuples in Python, a visual search task asked: *"Find a slide where the instructor displays on screen examples of the singleton operation."* Targets were slides that appeared briefly (less than 20 seconds) in the video. We mixed tasks that had targets around interaction peaks and non-peaks to investigate how LectureScape fares even when the target is not near a peak. For all visual search tasks, we provided learners with only the video timeline (linear timeline in baseline, 2D timeline in LectureScape) and removed all other features (e.g., transcript, word cloud) to restrict video navigation to timelines only. Learners were told to pause the video as soon as they navigated to the answer.
- **Problem search** tasks involved finding an answer to a given problem. These tasks emulated a learner rewatching a relevant video to answer a discussion forum question or to solve a homework problem. For example, for a video about approximation methods, a problem search task asked: *"If the step size in an approximation method decreases, does the code run faster or slower?"* Learners were asked to find the part in the video that discussed the answer, and then state their answer.
- **Summarization** tasks required learners to write down the main points of a video while skimming through it. We gave learners only three minutes to summarize videos that were seven to eight minutes long, with the intent of motivating learners to be selective about what parts to watch.

All videos used in the study were from an introductory computer science course on edX. Interaction data was collected from server logs during the first offering of the course in fall 2012. The course has been recurring every semester since then. Each of the eight tasks in the study used different videos to minimize learning effects. We also chose videos of similar length, difficulty, and style within each task type to control for differences across videos.

Participants

We recruited 12 participants (5 male, mean age 25.6, stdev=11.0, max=49, min=18) via a recruitment flyer on the course discussion forum on edX and the on-campus course website, both with consent from instructors. We recruited only learners who were currently enrolled in the introductory CS course (either on edX or on campus) to which the videos belong. The on-campus version of the course shares the same curriculum but is taught by different instructors. Furthermore, we picked videos for lessons given earlier in the semester, so that participants were likely to have already been exposed to that material before coming to our study, as is often the case in video re-watching scenarios. Four participants were enrolled in a current or previous edX offering of the course, while six were taking the on-campus version. Two had previously registered in the online offering but were currently taking the on-campus course. Participants received $30 for their time.

Procedure

A 75-minute study session started with 15-minute tutorials on both interfaces. Next, participants performed eight learning tasks: four visual search tasks, two problem search tasks, and two summarization tasks. After each task, they answered questions about confidence in their answer and prior exposure to the video. After each task type, we interviewed them about their task strategy. For each task type, we counterbalanced the order of the interfaces and the assignment of videos to tasks. After completing all the tasks, participants completed a questionnaire on the usability of each interface and their experience and opinions about interaction data. All click-level interactions were logged by the server for analysis.

Results

RQ1. Navigation Patterns for Search and Summarization

In **visual search**, most participants in the baseline 1D timeline sequentially scanned the video using thumbnail previews or dragging. In contrast, participants using the 2D Rollercoaster timeline often jumped between interaction peaks to reach the general area of the answer. But the latter strategy did not help in many cases because interaction data represents collective interests in a video, not results for a search query.

For the two out of four tasks where the search targets were located near interaction peaks, it took participants in both conditions similar amounts of time (LectureScape: μ=85 seconds, σ=51, baseline: μ=80, σ=73). This difference was not statistically significant with the Mann-Whitney U (MWU) test ($p>0.4$, Z=-0.9). For the other two tasks where the search targets were outside of an interaction peak range, it took participants in the LectureScape condition longer to complete (μ=117, σ=44) than in the baseline condition (μ=90, σ=50), although the MWU test showed no statistical significance ($p>0.1$, Z=-1.5) The longer time might be due to the fact that many participants navigated by clicking on the peaks to see if the answer existed around peaks. Nonetheless, results show that LectureScape did not adversely affect task completion times even when the answer was not in peak ranges.

Because **problem search** tasks required some understanding of the context to provide an answer, participants often tackled the problem by narrowing down to a section of the video that mentioned relevant concepts and then watching the section until they found the answer. In the process of narrowing down to a video section, most participants in the baseline condition relied on searching for keywords (10 out of 12) and clicking on transcript text (11 / 12), while participants with LectureScape used search (6 / 12) and clicking on transcript text (6 / 12) less frequently, and additionally clicked on interaction peaks on the timeline (6 / 12) or highlights below the video (6 / 12). Over all problem search tasks, participants in the LectureScape condition completed them slightly faster (μ=96, σ=58) than participants in the baseline condition (μ=106, σ=58), although the difference was not significant (MWU test, p=0.8, Z=0.3).

In comparison to the other tasks, participants in **summarization** tasks did not rely on keyword search (1 / 12 in both conditions), because the task required them to quickly scan the entire video for the main points. Many participants with LectureScape scanned the peak visualization, word cloud, and highlights for an overview, and clicked on interaction peaks (9 / 12) or highlights (6 / 12) for a detailed review. In one video, all six participants with LectureScape visited an interaction peak almost at the end of the video (located at 6:35 in the 7:00 clip). This slide summarized main ideas of variable binding, which was the topic of the video. In contrast, in the baseline condition, only one learner navigated to this section of the video. Most participants spent majority of their time in the earlier part of the video in the baseline condition.

Despite inconclusive evidence on quantitative differences in task completion time, participants believed that they were able to complete the tasks faster and more efficiently with LectureScape than with the baseline interface. Answers to 7-point Likert scale questions on the overall task experience revealed significant differences between the two interfaces in participants' belief in speed (LectureScape: μ=5.8, Baseline: μ=4.8) and efficiency (LectureScape: μ=6.1, Baseline: μ=4.8). The MWU test shows that both differences were significant at $p<0.05$ for these questions.

RQ2. Perception of Interaction Data

Generally, participants' comments about watching videos augmented by others' interaction data were positive. Participants noted that *"It's not like cold-watching. It feels like watching with other students."*, and *"[interaction data] makes it seem more classroom-y, as in you can compare yourself to what how other students are learning and what they need to repeat."*

In response to 7-point Likert scale questions about the experience of seeing interaction data, participants indicated that they found such data to be "easy to understand" (μ=5.9), "useful" (5.3), "enjoyable" (5.2), that interaction peaks affected their navigation (5), and that interaction peaks matched their personal points of interest in the video (4.4).

In an open-ended question, we asked participants why they thought interaction peaks occurred. Common reasons provided were that these parts were "confusing" (mentioned by 8 / 12), "important" (6 / 12), and "complex" (4 / 12). Identifying the cause of a peak might be useful because they can enable more customized navigation support. While most participants mentioned that highlighting confusing and important parts would be useful, some noted that personal context may not match the collective patterns. One said, *"If it were a topic where I was not very confident in my own knowledge, I would find it very helpful to emphasize where others have re-watched the video. If however it was a topic I was comfortable with and was watching just to review, I would find it frustrating to have the physical scrolling be slowed down due to others' behavior while watching the video."*

RQ3. Perceived Usability of LectureScape

Many participants preferred having more options in navigating lecture videos. As one learner noted, *"I like all the extra features! I was sad when they got taken away [for the baseline condition]."* Also, when asked if the interface had all the functions and capabilities they expected, participants rated LectureScape (6.4) significantly higher than the baseline interface (4.3) ($p<0.001$, Z=-3.2 with the MWU test).

However, some expressed that LectureScape was visually complex, and that they would have liked to hide some widgets not in use at the moment. They found it more difficult to use than the baseline (ease of use: 4.7 vs. 6.3, $p<0.001$, $Z=2.7$ with the MWU test). This perception leaves room for improvement in the learnability of the system. A participant commented: *"[LectureScape] is fairly complex and has a lot of different elements so I think it may take a bit of time for users to fully adjust to using the interface to its full potential."* These comments are consistent with our design decision to support collapsible widgets to reduce visual clutter, although the version of LectureScape used in the study had all features activated for the purpose of usability testing.

Due to the limitations of a single-session lab study, few participants actively used personal bookmarks or personal history traces. A longitudinal deployment might be required to evaluate the usefulness of these features.

Navigation Pattern Analysis

Now we provide a detailed analysis of how participants navigated videos during the study. In all tasks, most participants' strategy was to start with an overview, and then focus on some parts in detail. Participants alternated between the overview and focus stages until they found what they were looking for, or covered all major points in the video for summarization.

While the high-level strategy was similar in the two video interfaces, participants' navigation patterns within each of the two stages differed noticeably. With LectureScape, participants used more diverse options for overview and focused navigation, making more directed jumps to important points in the video. With the baseline interface, most participants sequentially scanned the video for overview and clicked on the timeline or transcript for focusing. Another common pattern in the baseline condition was to make short and conservative jumps on the timeline from the beginning of the video, in order not to miss anything important while moving quickly.

In the **overview** stage, most participants tried to scan the video to grasp the general flow and select a few points to review further. One learner described her strategy with LectureScape in this stage: *"having this idea of 'here's where other people have gone back and rewatched, being able to visually skim through very quickly and see titles, main bullet points, and following along with the transcript a little bit as well was definitely helpful."* Although visual scanning did not result in click log entries, our interviews with participants confirm that it was a common pattern. They pointed out three main features in LectureScape that supported overview:

- the 2D timeline with an overall learner activity visualization: *"I could use the 2D overlay to scroll through... I think I took a quick scan through and saw the general overview of what was on the slides."*
- highlight summaries with a strip of screenshots: *"They would get me close to the information I needed. They also made it easier to quickly summarize."*
- the word cloud with main keywords for sections in the video: *"I just looked at the top keywords, then I watched the video to see how [the instructor] uses those keywords."*

After scanning for an overview, participants chose a point in the video to watch further. All of the methods described above provide a single-click mechanism to directly jump to an "important" part of the video. Participants reviewed data-driven suggestions from LectureScape to make informed decisions. The log analysis reveals that participants using LectureScape made direct jumps such as clicking on a specific point in the timeline or a highlight 8.4 times on average per task, in contrast to 5.6 times in the baseline condition.

In the **focus** stage, participants watched a segment in the video and reviewed if content is relevant to the task at hand. In this stage they relied on methods for precise navigation: scrubbing the timeline a few pixels for a second-by-second review, and re-watching video snippets multiple times until they fully comprehend the content. With LectureScape, participants had options to use the slowed-down scrubbing around peaks in the rollercoaster timeline, automatic pinning of the previous slide, and sentence-level playback in search. To navigate back to the previously examined point, participants frequently used timestamped anchors attached to search results, interaction peaks, and highlights.

In summary, with LectureScape, participants used more navigation options in both the overview and focus stages. A learner commented that *"[LectureScape] gives you more options. It personalizes the strategy I can use in the task."* They had more control in which part of the video to watch, which might have led them to believe that they completed the tasks faster and more efficiently.

DISCUSSION AND FUTURE WORK

Availability of interaction data: Discussion throughout this paper assumes the availability of large-scale interaction data. With modern Web technologies, clickstream logging can be easily added with APIs for video event handling.

There remain unanswered questions around interaction data, such as "Will using the data-driven techniques bias the data so that it reinforces premature peak signals and ignores other potentially important ones?", and "How many data points are required until salient peaks and patterns emerge?" Our future work will address these questions through a live deployment on a MOOC platform such as edX. We will also explore other types of interaction data such as active bookmarking and content streams such as voice to enrich video-based learning.

Adaptive video UI: The data-driven techniques introduced in this paper open opportunities for more adaptive and personalized video learning experiences. In this paper, we demonstrated how collective viewership data can change the video interface dynamically, influencing the physical scrubbing behavior, search ranking algorithm, and side-by-side frame display. We envision future video UIs that adapt to collective usage. Also, incorporating interaction data can lead to personalized video learning. Because interaction data is likely to represent an average learner, comparing personal history traces against collective traces may help model the current user more accurately and improve personalization.

Beyond MOOC-style lecture videos: While this paper used MOOC-style lecture videos for demonstration, we believe our techniques can generalize to other types of educational videos such as programming tutorial screencasts, how-to demonstration videos, and health education videos. We expect to apply the techniques introduced in this paper to these videos.

CONCLUSION

This paper introduces a novel concept of designing video interaction techniques by leveraging large-scale interaction data. We present three sets of data-driven techniques to demonstrate the capability of the concept: 2D, non-linear timeline, enhanced in-video search, and a visual summarization method. In a lab study, participants found interaction data to draw attention to points of importance and confusion, and navigated lecture videos with more control and flexibility.

Ultimately, the design techniques we have presented provide enriched alternatives to conventional video navigation. We envision engaging a community of learners in creating a social, interactive, and collaborative video learning environment powered by rich community data.

ACKNOWLEDGMENTS

The authors thank the members of the UID Group at MIT CSAIL for their feedback, and the 6.01x course staff for study advertisement. This work was funded in part by Quanta Computer. The Samsung Fellowship has supported Juho Kim.

REFERENCES

1. Al-Hajri, A., Miller, G., Fong, M., and Fels, S. S. Visualization of personal history for video navigation. In *CHI '14* (2014).
2. Alexander, J., Cockburn, A., Fitchett, S., Gutwin, C., and Greenberg, S. Revisiting read wear: Analysis, design, and evaluation of a footprints scrollbar. In *CHI '09* (2009), 1665–1674.
3. Arman, F., Depommier, R., Hsu, A., and Chiu, M.-Y. Content-based browsing of video sequences. In *MULTIMEDIA '94* (1994), 97–103.
4. Baudisch, P., Cutrell, E., Hinckley, K., and Eversole, A. Snap-and-go: Helping users align objects without the modality of traditional snapping. In *CHI '05* (2005), 301–310.
5. Blanch, R., Guiard, Y., and Beaudouin-Lafon, M. Semantic pointing: Improving target acquisition with control-display ratio adaptation. In *CHI '04* (2004), 519–526.
6. Bostock, M., Ogievetsky, V., and Heer, J. D^3 data-driven documents. *Visualization and Computer Graphics, IEEE Transactions on 17*, 12 (2011), 2301–2309.
7. Carlier, A., Charvillat, V., Ooi, W. T., Grigoras, R., and Morin, G. Crowdsourced automatic zoom and scroll for video retargeting. In *Multimedia '10* (2010), 201–210.
8. Chi, P.-Y. P., Liu, J., Linder, J., Dontcheva, M., Li, W., and Hartmann, B. Democut: generating concise instructional videos for physical demonstrations. In *UIST '13*, ACM (2013).
9. Chorianopoulos, K. Collective intelligence within web video. *Human-centric Computing and Information Sciences 3*, 1 (2013), 10.
10. Dieberger, A., Dourish, P., Höök, K., Resnick, P., and Wexelblat, A. Social navigation: Techniques for building more usable systems. *Interactions 7*, 6 (Nov. 2000), 36–45.
11. Ding, W., and Marchionini, G. A study on video browsing strategies. Tech. rep., College Park, MD, USA, 1997.
12. Grossman, T., Matejka, J., and Fitzmaurice, G. Chronicle: capture, exploration, and playback of document workflow histories. In *UIST '10* (2010).
13. Guo, P. J., Kim, J., and Rubin, R. How video production affects student engagement: An empirical study of mooc videos. In *L@S '14* (2014), 41–50.
14. Hill, W. C., Hollan, J. D., Wroblewski, D., and McCandless, T. Edit wear and read wear. In *CHI '92* (1992), 3–9.
15. Hou, X., and Zhang, L. Saliency detection: A spectral residual approach. In *CVPR '07* (2007), 1–8.
16. Hurst, A., Mankoff, J., Dey, A. K., and Hudson, S. E. Dirty desktops: Using a patina of magnetic mouse dust to make common interactor targets easier to select. In *UIST '07* (2007), 183–186.
17. Hürst, W., Götz, G., and Jarvers, P. Advanced user interfaces for dynamic video browsing. In *MULTIMEDIA '04* (2004), 742–743.
18. Jackson, D., Nicholson, J., Stoeckigt, G., Wrobel, R., Thieme, A., and Olivier, P. Panopticon: A parallel video overview system. In *UIST '13* (2013), 123–130.
19. Jansen, M., Heeren, W., and van Dijk, B. Videotrees: Improving video surrogate presentation using hierarchy. In *CBMI 2008*, IEEE (June 2008), 560–567.
20. Kim, J., Guo, P. J., Seaton, D. T., Mitros, P., Gajos, K. Z., and Miller, R. C. Understanding in-video dropouts and interaction peaks in online lecture videos. In *L@S '14* (2014), 31–40.
21. Kim, J., Nguyen, P., Weir, S., Guo, P., Gajos, K., and Miller, R. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *CHI '14* (2014).
22. Li, F. C., Gupta, A., Sanocki, E., He, L.-w., and Rui, Y. Browsing digital video. In *CHI '00* (2000), 169–176.
23. Marcus, A., Bernstein, M. S., Badar, O., Karger, D. R., Madden, S., and Miller, R. C. Twitinfo: aggregating and visualizing microblogs for event exploration. In *CHI '11* (2011), 227–236.
24. Masui, T., Kashiwagi, K., and Borden, IV, G. R. Elastic graphical interfaces to precise data manipulation. In *CHI '95* (1995), 143–144.
25. Matejka, J., Grossman, T., and Fitzmaurice, G. Swift: Reducing the effects of latency in online video scrubbing. In *CHI '12* (2012), 637–646.
26. Matejka, J., Grossman, T., and Fitzmaurice, G. Patina: Dynamic heatmaps for visualizing application usage. In *CHI '13* (2013), 3227–3236.
27. Matejka, J., Grossman, T., and Fitzmaurice, G. Swifter: Improved online video scrubbing. In *CHI '13* (2013), 1159–1168.
28. Mertens, R., Farzan, R., and Brusilovsky, P. Social navigation in web lectures. In *HYPERTEXT '06* (2006), 41–44.
29. Monserrat, T.-J. K. P., Zhao, S., McGee, K., and Pandey, A. V. Notevideo: Facilitating navigation of blackboard-style lecture videos. In *CHI '13* (2013), 1139–1148.
30. Nancel, M., and Cockburn, A. Causality: A conceptual model of interaction history. In *CHI '14* (2014).
31. Nguyen, C., Niu, Y., and Liu, F. Video summagator: An interface for video summarization and navigation. In *CHI '12* (2012), 647–650.
32. Nicholson, J., Huber, M., Jackson, D., and Olivier, P. Panopticon as an elearning support search tool. In *CHI '14* (2014).
33. Olsen, D. R., and Moon, B. Video summarization based on user interaction. In *EuroITV '11* (2011), 115–122.
34. Pongnumkul, S., Wang, J., Ramos, G., and Cohen, M. Content-aware dynamic timeline for video browsing. In *UIST '10* (2010), 139–142.
35. Ramos, G., and Balakrishnan, R. Fluid interaction techniques for the control and annotation of digital video. In *UIST '03* (2003), 105–114.
36. Risko, E., Foulsham, T., Dawson, S., and Kingstone, A. The collaborative lecture annotation system (clas): A new tool for distributed learning. *Learning Technologies, IEEE Transactions on 6*, 1 (2013), 4–13.
37. Scott MacKenzie, I., and Riddersma, S. Effects of output display and control–display gain on human performance in interactive systems. *Behaviour & Information Technology 13*, 5 (1994), 328–337.
38. Shaw, R., and Davis, M. Toward emergent representations for video. In *Multimedia '05* (2005), 431–434.
39. Yew, J., Shamma, D. A., and Churchill, E. F. Knowing funny: genre perception and categorization in social video sharing. In *CHI '11* (2011), 297–306.