Video Metadata Viewer

Mackenzie Leake
Stanford University
Stanford, CA USA
mleake@cs.stanford.edu

ABSTRACT
We present Video Metadata Viewer, a tool for visualizing features of scripted video sequences. Our system takes as input time-aligned videos, transcripts, and a script and aggregates information about the videos. This additional metadata includes information about the timing and content of the script lines as well as information about the zoom type of the shot. The system then provides users with interfaces at two different levels of abstraction (lines and frames) for exploring the metadata associated with these inputs. The line level visualization allows users to compare actor performance and line timing across different takes of the same scene. The frame level visualization supports users in assessing the accuracy of the shot type labels for each frame. Together these visualizations allow the user to learn more about their video data in order to help them make faster, more informed video editing decisions.

ACM Classification Keywords
H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

Author Keywords
Video; face detection; media visualization

INTRODUCTION
The world is increasingly turning to audio-visual media to entertain viewers and convey important news information. Many traditional news organizations, such as *The New York Times*, are increasing the size of their video production teams to meet the world's growing demand for audio-visual media. Home moviemakers, too, are faced with large amounts of captured video footage. But video production is a complex, slow task. Editing video for professionals and amateurs alike is challenging for many reasons.

One reason editing videos can be difficult is that the amount of raw video footage is often overwhelming. Editing video can require rewatching hours of video in order produce just a few minutes of meaningful content. Conventional video editing systems require users to scrub through the video timeline to find relevant pieces to include in the final edited sequence. Our system facilitates the shot selection process by time-aligning videos to their corresponding scripts and transcripts. This allows users to jump to specific content without having to scrub through the timeline.

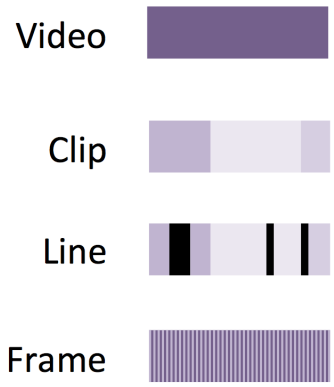

Figure 1. Video can be considered at a variety of levels of abstraction. Our system emphasizes the line and frame level abstractions.

Working with video can also be challenging because it requires thinking about the content of the narrative at various levels of abstraction. In commercial video editing software it is common to consider the audio and video separately. While this division allows for many sophisticated editing techniques, there are other levels of division that can support a closer connection between the audio-visual media and the narrative the director intends to tell.

The visualizations we present support users in exploring the narrative elements at various levels of abstraction. Videos can be broken into clips, shots, lines, or even frames (Fig. 1). Some edits require the semantic knowledge only available at the line level, while other edits require the fine granularity afforded by the frame-level information. Our system supports users in thinking about these features at the line and frame level through two separate visualizations that provide information about the videos at each of these levels of abstraction.

The goal of this visualization system is to help users understand their video data in order to make more informed editing decisions. In order to edit videos, visual and audio elements

must be combined into a coherent narrative. Our visualization tool is intended to support the video editing process by helping people understand the many attributes of their video data, including video, audio, and text inputs. This visualization system is part of a larger, ongoing research project called RoughCut, in which we incorporate principles of cinematography to guide the automatic sequencing of video clips.

RELATED WORK

Browsing and summarizing videos

To support browsing content within videos, many commercial video editing and playback interfaces require users to scrub through the timeline to find relevant sections. Goldman et al., 2006 present semantic storyboards as an alternative to the timeline scrubbing commercial nonlinear video editing software requires [8]. Using a static image and the visual language of storyboards, the system provides users with an overview of the video content. Barnes et al., 2010 [3] present another alternative to video scrubbing through the creation of video tapestries, which provide multiscale image summaries of videos. Zhang, Wang, and Altunbasak, 1995 [17] use 2-D keyframe images to support users in browsing videos. Instead of using static 2-D images to represent video content, Panopticon, a video alternative system, presents an overview of the full video content to users by displaying an animated grid of video subsequences [9]. Like these systems, our visualization tool reduces the need for scrubbing through videos to find relevant content. Our system also provides additional mechanisms for exploring a video's underlying metadata.

Truong and Venkatesh, 2007 [15] discuss the importance of abstraction for video browsing and summarization. Silver, a system designed by Casares et al., 2002, provides multiple views, including storyboard, transcript, and timeline views, to support the editing process [5]. Video digests, presented by Pavel et al., 2014, support the browsing and skimming of instructional videos by pairing video segments with their summaries [12]. The video digest system allows users to create browsable video digests using transcripts, chapter summaries, and keyframes. While Silver and the video digest system consider video at various levels of abstraction, including the chapter, clip, shot, and frame levels, the visualization system described in this paper provides information at an additional level of abstraction: the line level. This level of abstraction places added emphasis on the topics discussed during the video segment.

SceneSkim is a system that supports the alignment of the script, plot summary, and captions to videos for browsing content in feature films [11]. This system utilizes the alignment between videos and text elements for studying existing films. While our system uses many of the same techniques for aligning the text, audio, and visual elements, our system uses these techniques for generating new edited sequences rather than exploring existing movies.

Video editing

The IMPACT system proposed by Ueda, Miyatake, and Yoshizawa, 1991 [16] addresses some of the difficulties of editing video by presenting the use of image processing techniques to support multimedia editing. Hitchcock, the automatic home editing system proposed by Girgensohn et al., 2000, utilizes camera motion and video editing conventions regarding clip length and brightness to identify suitable clips for inclusion in the final edited sequence [7]. Our system also makes use of video metadata for supporting the automatic editing process. It makes an additional connection between videos and the spoken narrative by presenting to the user the video clips time-aligned with their corresponding transcripts.

DemoCut is a system for semi-automatically editing instructional videos for physical demonstrations [6]. Users identify important steps in the video, and then the system automatically sequences the video and audio segments. Arev et al., 2014 present methods for automatically editing videos taken from multiple social camera based upon cinematographic principles and important actions in the scene [2]. Our system too, encourages users to be mindful of incorporating a variety of shot types into their edited sequences, but it focuses more on conversational video footage and considers the alignment of a video with its transcript to guide the editing process.

Berthouzoz, Li, & Agrawala, 2012 present a set of tools for placing smooth transitions in interview videos [4]. These tools make use of the alignment between the video and its transcript in order to guide automatic edits. Our system is similar in its emphasis on conversation video and use of transcript-video alignment. Our system, however, also presents the user with additional information, such as identifying the emotional content in the script, to support the user in understanding the available video content for editing.

DEFINITIONS

Our system collects video, audio, and text data and helps the user visualize this information at various levels of abstraction. In this section we define terminology that will appear throughout the rest of the paper to describe various aspects of the visualization tool.

- A *line* is a single turn an actor takes speaking. A single line might be a word, sentence, or multiple sentences. A line ends when the other actor begins speaking.

- The *script* is a written document that lists in order the words that each of the actors will read and perform. A script includes the name of the character intended to read each line.

- The *transcript* is a verbatim account of what the actors said. *Captions* provide this verbatim account as well as timestamps indicating when the actors speak each of the lines. In this paper we use the terms *transcript* and *captions* interchangeably.

- *Shot type* is the zoom-level of a particular video. We classify shot types based upon the facial landmark points provided by our face tracking software. We show the six shot types our system considers in Fig. 4.

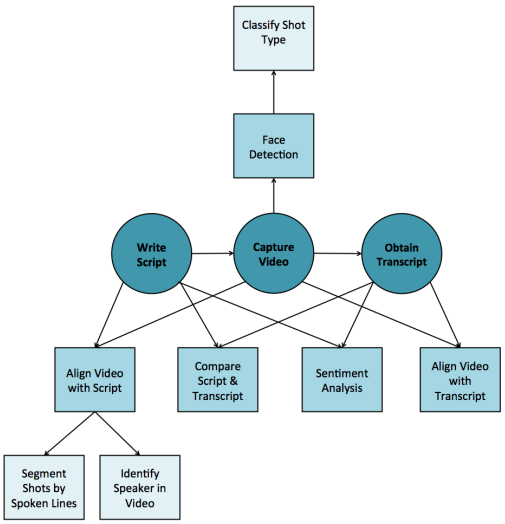

Figure 2. Our system takes as input data generated through a process of aligning and labeling videos, scripts, and transcripts.

METHODS

Video-Script and Video-Transcript Alignment

The Video Metadata Viewer supports users in understanding their video data. Essential inputs to the system are a script outlining what the actors will say and videos showing the actors performing the lines. Using the implementation of the Penn Phonetics Lab Forced Aligner (P2FA) provided by Rubin et al., 2013 [13], we align each word of the script to the video. This alignment tool maps a list of phonemes to audio elements in the video using a Hidden Markov Model. We then split each of these aligned sections into lines based upon when the actors switch turns speaking.

Frequently, when performing, actors deviate from the script in the words they speak. In order to account for these differences, we obtain captions for each of the videos using an online transcription service, `rev.com`. We upload our videos to this website and pay $1 per minute to obtain time-stamped transcripts for the content in the video. We compare the script and transcript lines by computing the edit distance between each of the lines in the script and the transcript using the Levenshtein distance metric [10]. Once we identify the corresponding script and transcript line pairs with the shortest distance between one another, we can use these pairs of lines as inputs to the P2FA alignment tool [13] to obtain the start and end times of each spoken line.

Face Detection

To indicate shot type levels we use face detection algorithms to provide facial landmark points on faces in the videos. To identify the faces in the videos, our system uses Jason Saragih's face tracking software [14]. This face tracking software provides 66 facial points for each frame (Fig. 3). We use these facial landmark points to draw bounding boxes around each of the faces. We use the points corresponding to the widest points of the left and right side of the face and the top of the face and bottom of the chin to calculate the width and height of the face, respectively. This provides us with a bounding boxes for the face in each frame.

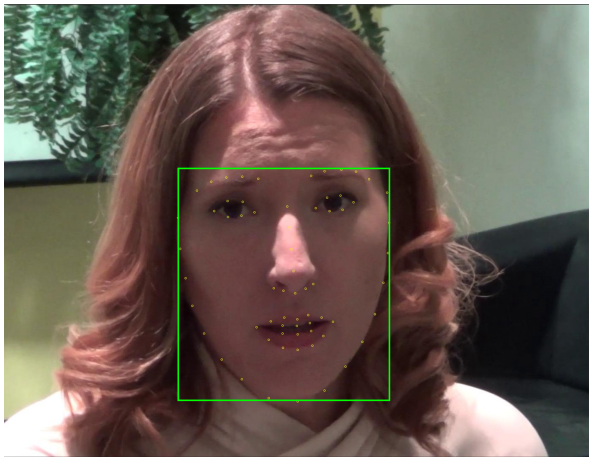

Figure 3. The face detector provides facial landmark points to identify the size of the face in the frame.

The size of each face in the video relative to the size of the frame provides an approximation of the the focal length of the lens setting on the camera's zoom lens. We then use a set of thresholds to classify these ratios into six shot types: extreme close up (ECU), close up (CU), medium close up (MCU), medium (MS), medium long (MLS), and long (LS), as shown in Fig. 4. We obtain a shot type label for each frame in each video. We can also label the shot type at the line level by obtaining the mode of the shot type labels for all frames corresponding to a given line in the video.

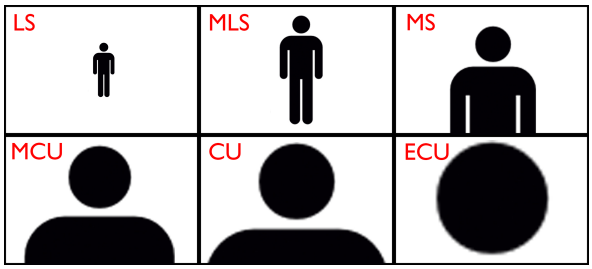

Figure 4. Our system provides shot type labels based upon the size of faces in each frame.

Speaker Label and Sentiment Analysis

To obtain additional labels for our input videos, our system also takes as input speaker labels and emotion scores for each line in the script. The speaker labels correspond to the labels provided within the script. The emotion scores of each line are calculated using the Python TextBlob package [1]. TextBlob computes a sentiment score between -1 and 1 based upon a Naive Bayes model trained on a movie review dataset. In our system we ignore the valence of the emotion and use the absolute value of the emotion score in our analysis.

RESULTS

Preprocessing

The majority of the time that the user must spend to use the system is invested in capturing the footage and obtaining the transcripts. Our system so far has been tested on short video segments (1920x1080 pixels) that are between 40 and 75 MB in size and have durations between 45 seconds and two minutes. The face tracking algorithms run at a rate between 15 and 30 fps. The turnaround time for our transcripts from rev.com is often between 8 and 24 hours.

Once the transcripts and face tracking information have been collected, the rest of the pipeline takes approximately $3\times$ the length of the video to align the video segment with the corresponding script and transcript, obtain speaker labels, and generate shot type classifications and emotion scores. Therefore, in total, after obtaining transcripts from a third-party source, the system takes approximately $5\times$ the length of the short input video to generate the data. The data for each video is collected independently, so the data processing pipeline could be performed in parallel for each video input.

The output data for each video is stored as a single JSON file, which is read by the web-based visualization system. The system is intended to be interactive, although sometimes a time lag is present from the server processing the video files.

Line Level Interface

Our visualization system features a video player as well as script and transcript viewers in the line level interface shown in Fig. 5. The script and transcript viewer allows for a number of user interactions, including selecting files to load, hovering over a script or transcript section to see the line, and clicking on a script or transcript section to jump to the corresponding section of video. Each sub-rectangle of the script and transcript viewer bars corresponds to a line in the script. The width of each colored rectangle in the visualization corresponds to the length of that line as reported by the script or transcript alignment. This feature of being able to click to jump to the correct video section allows the user to avoid scrubbing through the video to find the relevant content.

As shown in the close up of the script and transcript viewer and legend in Fig. 6, the hue of the rectangle indicates which actor is speaking, with green corresponding to Andrew's lines and magenta corresponding to the Doctor's lines in the example footage provided. The opacity of the rectangle indicates the emotion score of the line, with stronger colors reflecting more emotional lines and weaker colors reflecting less emotional lines. Together these visual encodings provide information about the line's length, content, speaker, and emotion.

Having separate script and transcript bars allows the user to compare the script and the transcript content. This enables the user to identify places where the actor deviated from the script. Seeing the transcript text for each line as well as being able to play the line easily allows the user to compare an actor's performance across the different takes.

Frame Level Interface

The frame level visualization as shown in Fig. 7 provides a frame selector, file selector, speaker indicator, and shot type indicator. Users can select a file to view using the dropdown menu and adjust the slider to scroll through the frames to view the face markers and face bounding box for each frame. By supporting the user in sliding between frames, the system allows the user to understand how well the face detections are working. The user can also assess the efficacy of the shot type classifier based upon comparing the highlighted shot type in the shot type indicator to the size of the actor in the frame.

We highlight different areas of the shot type indicator to show the user which shot type has been detected for the given frame. As shown in Fig. 8, when the shot type is known, the shot type label is highlighted in yellow in the grid, and when the shot type is unknown the full shot type indicator becomes red so the user can take note of that particular frame to assess why the detector failed at that instance.

DISCUSSION

While it is common in commercial video editing software to consider the audio and video separately, this visualization supports users in exploring their data in a different way: at the line and frame level. In designing this visualization, one of the major questions is how to reveal to users as much of the video metadata as possible without producing a cluttered design. We intentionally separate the system into two interfaces at different levels of abstraction. While this separation is intended to reduce confusion, it is critical that we support users in transitioning between these two views to think about their data at both levels of abstraction.

Line level visualization

This set of visualization tools aids users in understanding their video content. The line level interface allows for selections of multiple takes to enable users to compare shot types and actor performances across takes. Clicking on the line to jump to the corresponding section supports this comparison across different takes at the line level. Without these tools users would be required to open each video and scrub through the timeline to find the appropriate sections of the video to compare. This scrubbing method would be far more time consuming.

Being able to compare the script and transcript level alignments also supports the user in understanding how the alignment tools are performing. The varying widths of the rectangles transcript and script viewer sections shows where the script and transcript are in disagreement. Without visualization tools to support this comparison, it can be challenging to determine where the actors are going off script and how this is affecting the alignment results. Since our editing system segments videos at the line level, it is critical that the user ensure that the alignment has been performed accurately.

Frame level visualization

The frame level visualization supports users in understanding how their face detection software is performing. It allows users to find specific frames where the shot type is not predicted or is predicted incorrectly. This capability also supports

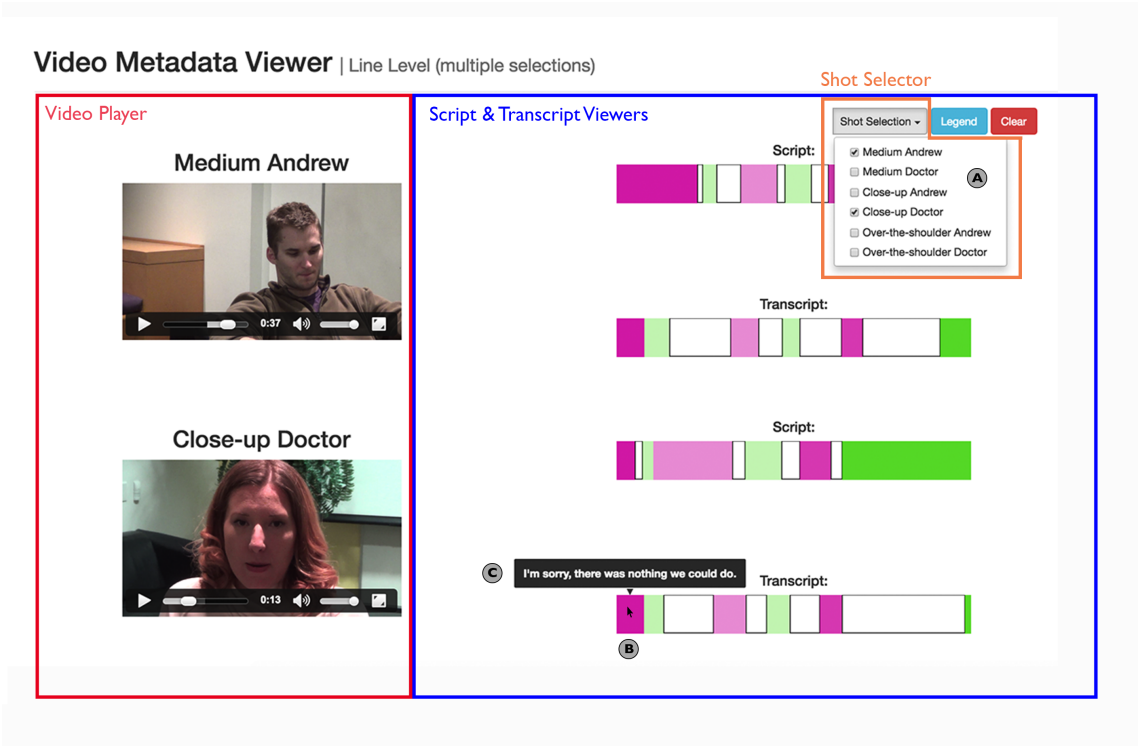

Figure 5. Our visualization system features a video player as well as script and transcript viewers. The script and transcript viewer allows for a number of user interactions, including A) selecting files to load, B) hovering over a script/transcript section to see the line, and C) clicking on a script/transcript section to jump to the corresponding piece of video.

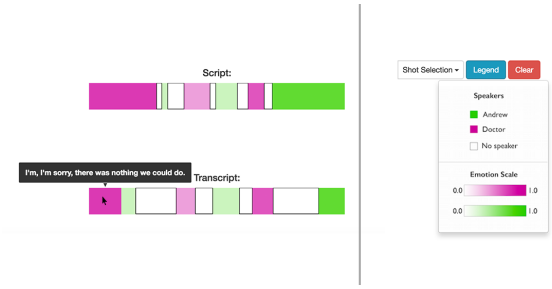

Figure 6. We provide a legend for helping the user understand the line level data concerning the content, speaker, and emotion of the line.

the debugging of the shot type classifier and has helped us determine the threshold values for classifying the bounding boxes into the different zoom levels. Since the editing system that we are building alongside these visualization tools relies heavily upon the accuracy of the shot type labels, being able to visualize where the face detections are failing is critical. In our informal evaluation this capability has proved useful in identifying problems with our videos, including extreme profile camera angles, distracting backgrounds, and poor lighting.

Informal user testing

Throughout the design process we asked several potential users of our system to provide feedback on our interfaces. Our tool will be used within our lab for understanding our video data and helping us improve our video editing system. In showing the interface to other members of the lab, two people commented on the helpfulness of the ability to click on a section of the script or transcript in the line level interface and jump to that corresponding section of the video. Another user who is not a lab member but has professional video experience said that the highlighting in the shot type indicator in the frame level visualization helped him understand how our system classifies shot zooms. Evaluating which of these features are most effective will be helpful in our designing the editing interface.

FUTURE WORK

The visualization tools described in this paper are part of a larger video editing research project called RoughCut. These tools are helpful for aiding our design of this larger system. In addition to helping our own design process, it is likely that many of these techniques for interacting with the available data will be presented to users of our editing system.

Our system currently relies on full coverage, i.e., that each of the videos has a full set of labels to analyze. In realistic video production settings full coverage is often not available. An important area of future work is helping users visualize the available material and identify pieces they are missing.

Another area of future design for this project is visualizing the audio data that corresponds with the video. The audio data can provide important insights about the actors' performance quality and would likely be beneficial for helping the editor chose video segments among different takes of the same line.

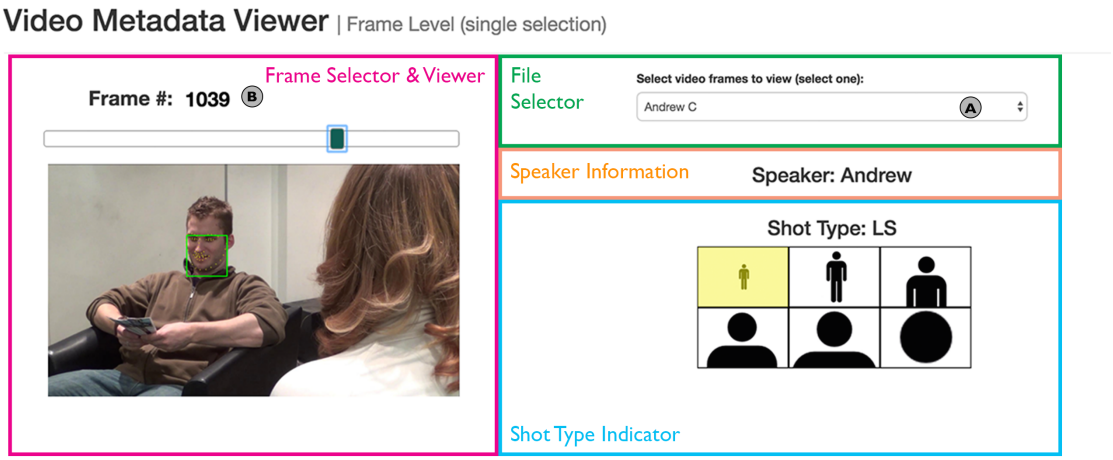

Figure 7. The frame level visualization provides a frame selector, file selector, speaker indicator, and shot type indicator. Users can A) select a file to view using the dropdown menu and B) slide the slider to scroll through the frames to view the face markers for each frame.

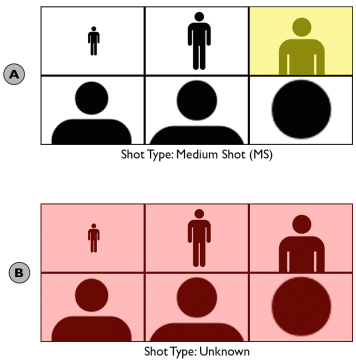

Figure 8. The shot type indicator highlights the shot type in A) yellow when the shot type is known and in B) red when the shot type is unknown.

In the area of video and multimedia visualization, much work remains in understanding which methods are most effective for helping users understand their video data in a way that eases the editing process. An important consideration is providing users with the appropriate amount of information to understand their videos at various levels of abstraction while not overwhelming them. Future work should aim to understand a) which additional video labels would be most helpful for understanding video content and b) which labels would be most informative for editing. Developing editing tools that can generate sequences rapidly is critical for supporting video production teams as they race to meet the world's growing demand for audio-visual media.

ACKNOWLEDGMENTS

We would like to thank Matthias Niessner for his help with the face detector, Alex Hall and Shannon Davis for lending their acting talents, and Katherine Breeden for sharing her insights about visualizing video metadata. We would also like to thank the CS448b students and course staff for their feedback throughout the quarter.

REFERENCES

1. 2013–2016. TextBlob: Simplified Text Processing (version 0.12.0). http://textblob.readthedocs.io/en/dev/. (2013–2016).
2. Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. 2014. Automatic editing of footage from multiple social cameras. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 81.
3. Connelly Barnes, Dan B Goldman, Eli Shechtman, and Adam Finkelstein. 2010. Video tapestries with continuous temporal zoom. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 89.
4. Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2012. Tools for placing cuts and transitions in interview video. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 67.
5. Juan Casares, A Chris Long, Brad A Myers, Rishi Bhatnagar, Scott M Stevens, Laura Dabbish, Dan Yocum, and Albert Corbett. 2002. Simplifying video editing using metadata. In *Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM, 157–166.
6. Pei-Yu Chi, Joyce Liu, Jason Linder, Mira Dontcheva, Wilmot Li, and Bjoern Hartmann. 2013. Democut: generating concise instructional videos for physical demonstrations. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 141–150.
7. Andreas Girgensohn, John Boreczky, Patrick Chiu, John Doherty, Jonathan Foote, Gene Golovchinsky, Shingo

Uchihashi, and Lynn Wilcox. 2000. A semi-automatic approach to home video editing. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*. ACM, 81–89.

8. Dan B Goldman, Brian Curless, David Salesin, and Steven M Seitz. 2006. Schematic storyboarding for video visualization and editing. In *ACM Transactions on Graphics (TOG)*, Vol. 25. ACM, 862–871.
9. Dan Jackson, James Nicholson, Gerrit Stoeckigt, Rebecca Wrobel, Anja Thieme, and Patrick Olivier. 2013. Panopticon: A parallel video overview system. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 123–130.
10. VI Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In *Soviet Physics Doklady*, Vol. 10. 707.
11. Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. SceneSkim: Searching and Browsing Movies Using Synchronized Captions, Scripts and Plot Summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM, 181–190.
12. Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. 2014. Video digests: a browsable, skimmable format for informational lecture videos.. In *UIST*. Citeseer, 573–582.
13. Steve Rubin, Floraine Berthouzoz, Gautham J Mysore, Wilmot Li, and Maneesh Agrawala. 2013. Content-based tools for editing audio stories. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, 113–122.
14. Jason M Saragih, Simon Lucey, and Jeffrey F Cohn. 2009. Face alignment through subspace constrained mean-shifts. In *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 1034–1041.
15. Ba Tu Truong and Svetha Venkatesh. 2007. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 3, 1 (2007), 3.
16. Hirotada Ueda, Takafumi Miyatake, and Satoshi Yoshizawa. 1991. IMPACT: an interactive natural-motion-picture dedicated multimedia authoring system. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 343–350.
17. HongJiang Zhang, Chien Yong Low, Stephen W Smoliar, and Jian Hua Wu. 1995. Video parsing, retrieval and browsing: an integrated and content-based solution. In *Proceedings of the third ACM international conference on Multimedia*. ACM, 15–24.