Designing Effective Interfaces for Audiovisual Media

by

Hijung Valentina Shin

B.S.E., Princeton University (2011)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Signature of Author .... Signature redacted

Department of Electrical Engineering and Computer Science
August 31, 2017

Certified by .... Signature redacted

Frédo Durand
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by.. Signature redacted

Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

ABSTRACT

Audiovisual media is an attractive and rich source of information, widely used in education, entertainment and industry. Unfortunately, audiovisual media can take painfully long hours to author, edit and navigate. This dissertation looks at ways to make creation, navigation and manipulation of audiovisual media easier and more convenient. In particular, we explore novel interaction techniques applied to three specific domains: speech recordings, lecture videos and presentations. For each domain, we propose a design solution to facilitate user interaction with the medium and implement the solution in a prototype interface. In doing so, we propose several design principles for effective audiovisual media interfaces that generalize to a wide range of applications and user tasks.

Dedicated to the loving memory of Msgr. Javier Echevarria.

1932 – 2016

PUBLICATIONS

Some of the research leading to this thesis has appeared previously in the following publications.

- Hijung Shin, Floraine Berthouzoz, Wilmot Li, Fredo Durand. **Visual Transcripts: Lecture Notes for Blackboard Style Lecture Videos.** – *ACM Transactions on Graphics (TOG) 34.6 (2015): 240.*
- Hijung Shin, Wilmot Li, Fredo Durand. **Voice Script: Dynamic Authoring of Audio with Linked Scripts.** – *Proceedings of the 29th Annual Symposium on User Interface Software and Technology. ACM, 2016.*

CONTENTS

1 INTRODUCTION

Multimedia is not more media,
but the employment of various kinds of media (and hybrid media)
for what they each offer to advance the narrative.

— Fred Ritchin [82]

1.1 CHALLENGES OF MULTIMEDIA

The main subject of this dissertation is **multimedia**, content that combines multiple forms of media such as text, graphics, audio and video. In particular, we focus on **audiovisual media** that have both a sound and a visual component. For example, presentations that involve graphics and sound, audiobooks that combine text with audio, and videos that incorporate graphics, sound and text are all different types of audiovisual media.

As these everyday examples illustrate, audiovisual media has become commonplace. From the consumer's perspective, we encounter them in our normal activities, for instance, listening to a music while following its lyrics in text, watching an advertisement or sitting in on a presentation. We also frequently produce audiovisuals to communicate our ideas, for example, by creating a blog or sharing a video on YouTube. In addition, new technologies and online platforms encourage new types of audiovisual media such as GoPro videos, 360-degree videos, interactive infographics or online lectures.

Unfortunately, ease of access does not translate directly to efficiency or good quality. Simply navigating audiovisual media to search for information can be tedious. Consider the times you had to listen to a voicemail repeatedly to get the call back number, or when you had to play the video back and forth to find a specific moment. Producing compelling audiovisual media is even more time-consuming and difficult. People spend hours to produce a single slide of presentation or a few seconds of video.

This dissertation explores three specific domains of audiovisual media: navigating lecture videos, authoring speech recordings and delivering slide presentations. For each domain, we propose a design solution to facilitate user interaction with the medium and implement the solution in a prototype interface. In doing so, we flesh out several design principles for effective audiovisual media interfaces that generalize to a wide range of applications and user tasks.

A major part of the difficulty in dealing with audiovisual media lies in the nature of multimedia itself. For one, as its name implies, multimedia blend multiple modalities, each of which have different characteristic strengths and weaknesses. Let's consider some of the basic components of multimedia.

TEXT: As a static representation of language, text is one of the most common forms of communication that most people are familiar with. It is easy to author and edit digitally, and many algorithms exist to process text, for instance, for text summarization or comparison. Text is also easy to navigate by skimming or searching. It can be organized spatially to emphasize structure and further facilitate navigation. In addition, different visual attributes such as typeface, font size or color can be used to convey extra information such as emphasis.
On the other hand, some types of information is less suited for textual representation. For example, describing a complicated diagram or a piece of music would be difficult to achieve using text alone. There is also a limit to conveying tonal nuances or voice.

AUDIO: Audio is a rich source of information that can convey not only speech but any other sound that may not have an accurate textual description, for example, the sound of a heartbeat or the sound of ocean waves. It is an effective media for evoking emotion or reflecting mood.

Figure 1: Audio represented in waveform.

However, most audio lack appropriate visual representation, which makes it difficult to navigate or edit. The waveform is the most generic representation used in many applications, but it is ambiguous and hard to manipulate (Figure 1). For instance, detecting or separating the sound of one instrument from a music recording is a challenging research problem.

IMAGE: A picture is worth a thousand words. Human perception is visually oriented, so images can be a powerful tool to communicate rich information intuitively in a small amount of space. Images are especially useful for conveying spatial relationships, structure, or detailed shape. Consider describing the layout of a building or the features of a face only using words versus showing a picture of the floor plan or the face.

On the other hand, it is difficult to convey information about movement or sequence using a still image. Oftentimes, text, labels or animation effects are attached to a still image to focus the viewers' attention to a specific part of the image or to clarify the intended message.

ANIMATION/VIDEO: Animations are created from a set of static frames whereas videos record a continuous event which is then broken up into a series of frames. Both media are especially useful for illustrating concepts that involve motion or sequence.
As with audio, it takes time to navigate through the content of an animation or video and it is difficult to skim or search.

TIME: In addition to having multiple modalities, audiovisual media is also intricately linked with time. Time imposes a linear structure on the media and is expressed, for example, with timelines on videos, scrollbars on websites and page numbers on slides. Whereas time provides a natural order to the media, it can also make non-linear interactions more cumbersome. For instance, watching a video from beginning to end is easy, but searching for specific scenes is harder.
Time also imposes a certain speed or pace on the media, and makes each *moment* within the media transient. Video frames, parts of webpages or slides are displayed for a limited amount of time and replaced with different, successive information. The abundance of concurrent and transitory information makes audiovisual media difficult to digest and manipulate compared to static media.
Finally, synchronization between multiple modalities is an essential part of audiovisual media. Effective navigation and manipulation of the media both require that the concurrence between multiple modes is appreciated and maintained. For instance, editing the audio track of a video normally requires editing the corresponding visual footage as well.

In sum, the complex interplay of multimodal information with each other and with time makes audiovisual media difficult to author, edit or navigate efficiently. Workflows around audiovisual media usually involve nonlinear navigation and iteration between different modalities. Unfortunately, conventional media browsers are best suited for linear navigation. Prevalent editing tools also handle each modality independently, leaving users to manage their interconnection manually. Both consumers and producers of audiovisual media have a right to be frustrated at the inefficiency caused by the lack of *user-friendly* and *media-friendly* interfaces that takes into account the users' workflows and characteristics of the media.

1.2 THE GOAL: EFFECTIVE INTERFACES FOR MULTIMEDIA

Just as effective multimedia carefully blends different types of media to advance the narrative, effective *interfaces* for multimedia must capitalize on the characteristics of each medium to expose the media to the users and provide tools to interact with it efficiently. Well-designed interfaces facilitate the users' workflow, be it authoring, editing or browsing, and whether it is linear or nonlinear. With the help of such interfaces, working with audiovisual media should be as simple and natural as working with text documents.

Text documents are easy to navigate. One can read them carefully or skim through them. There are efficient techniques to search for specific information. For example, a table of contents, index, alphabetically organized dictionaries and simple text search functions are well-established systems that are easy to use. Similarly, we create and edit text documents every day seemingly effortlessly, whenever we write an e-mail, a business report or a letter.

As natural as it seems today, text documents did not start out as an easy medium. Consider the 16-feet long scroll containing the text of the Diamond Sutra printed in 9th century China, or the Tripitaka Koreana, a collection of Buddhist scriptures carved onto 81,258 wooden printing blocks in 13th century Korea (Figure 3). These text documents were neither painless to author nor easy to navigate. Even after the advent of digital word processors in the 1960s, it took several decades until the *what-you-see-is-what-you-get* (WYSIWYG) form of word processors as we know them today became commonplace. Now, with online applications such as Google Docs, users can even collaborate on a single text document synchronously or asynchronously with ease. Wikipedia is an example of a new type of collaborative document that became possible through the development of online, collaborative text editors.

Figure 3: Before convenient interfaces were developed, text documents were neither easy to author nor simple to navigate. (Above) Tripitaka Koreana, a collection of Buddhist scriptures carved onto 81,258 wooden printing blocks in 13th century Korea. (Below) Each wood block measures 24cm x 70cm. The blocks are stored in Haeinsa, a Buddhist temple in South Korea.

Similar to text documents, better interfaces can make it easier for users to author, edit, navigate and collaborate on audiovisual media. Improved efficiency can also lead to greater expressivity and even new types of media. This dissertation takes a step towards this goal by exploring several approaches to designing effective interfaces for audiovisual media.

1.3 OPPORTUNITIES FOR MULTIMEDIA INTERFACES

Previously, we argued that the complex combination of multiple modalities make multimedia especially difficult to navigate and manipulate. Yet, the same multimodal quality can be turned into opportunities for designing effective user interfaces for multimedia. The specific design strategy for an interface depends on the media as well as the interaction that the interface is trying to support (e.g., authoring, editing or browsing). Here, we outline several broad principles that emerged through our work on several different applications.

Principle 1. Harness spatial and temporal structure:

The different components of multimedia are tightly associated with each other in time and space. For instance, video frames that are adjacent in time are likely to have similar content. Simultaneous audio and visuals are also closely related to each other. Even within a single image or a slide, spatial layout can reveal meaningful patterns about the content.

Inferring the spatial and temporal structure embedded in the media can help design effective interactions. For one, explicitly visualizing the structure can help viewers navigate the media efficiently. In addition, interfaces can facilitate editing by keeping track of and maintaining meaningful structure, for instance, the synchronization between different components.

Principle 2. Exploit different modalities to facilitate various types of interaction:

Each of the modalities that composes audiovisual media lends itself more naturally to different types of user interaction or user tasks. For example, during navigation, text is easier to skim or search, whereas audio is more effective in conveying tonal nuances or flow. During authoring, speaking is faster than writing, but editing audio is generally more time consuming than editing text.

Interfaces for audiovisual media can take advantage of different modalities or representations to facilitate different parts of user tasks. For example, in Chapter 3, we propose an interface for authoring voice recordings, which supports both authoring and editing via speech and text. A key challenge in designing such multimodal interfaces is to enable seamless interaction between the different modalities. That is, edits in one modality should be translatable into meaningful operations in other modalities.

Principle 3. Support direct manipulation with automation:

Hybrid approaches that combine direct manipulation with automatic algorithms have been proposed and successfully implemented in many application domains, including authoring of 3D models [45, 101], animations [12], and illustrations [26]. Automatic algorithms can greatly simplify tedious tasks and allow users to focus on the creative part of the workflow.

In light of the previous two principles, automatic algorithms can also take advantage of the inherent structure of the media or characteristics of different modalities. For example, in Chapter 2, we use an automatic algorithm to extract a set of static images from a video by analyzing the spatial and temporal structure of the visuals drawn continuously in the video. In Chapter 3, we take advantage of text–a discrete modality that is easier to process automatically–in order to compare and align audio. Finally, in Chapter 4, we use pre-authored digital slides to achieve, among other things, real-time beautification effects of hand-drawn strokes.

As the following chapters will illustrate, these principles are instantiated for each application through a combination of interface design, visualization, algorithms and data structures.

1.4 OVERVIEW

This dissertation is about designing effective interfaces to support the authoring and navigation of multimedia. We explore a range of applications – navigating lecture videos, authoring speech recordings and delivering slide presentations. We originally considered these applications in separate publications, presented to different communities in computer graphics and human computer interaction. The goal of this text is to distill the common themes across these applications and present them in a unified way, along with insights gained over the course of our work. Subsequent chapters are organized by application domain:

Navigating Lecture Videos (Chapter 2):

Lecture videos are widely used for learning, but existing video player interfaces are poorly equipped to support frequent learner tasks. For example, it is difficult to search or skim the content, or view the lecture material at one's own pace. For these and other reasons, some people prefer to read lecture notes than to watch videos.

To facilitate learning with videos, we propose **VisualTranscript**, a readable and skimmable interface for lecture videos. VisualTranscript transforms blackboard-style lecture videos into interactive lecture notes, interleaving static figures with hierarchically organized paragraphs of text. The interface serves as a short summary of the video as well as a convenient interface to browse or search through the lecture.

We describe automatic algorithms to (1) extract a discrete set of key illustrations from the video frames, and (2) to analyze the audio content and classify sentences into two categorizes: depictive sentences that verbalize what the illustrations depict, and explanatory sentences that provide additional information not directly represented by the illustrations. Both algorithms take advantage of spatial and temporal structures inferred from the video.

We compare VisualTranscript with a standard video player, and a state-of-the-art interface designed specifically for blackboard-style lecture videos. User evaluation suggests that users prefer VisualTranscript for learning and that VisualTranscript is effective in helping them browse or search through lecture videos.

Authoring Speech Recordings (Chapter 3):

Speech recordings are central to modern media from podcasts to audio books to e-lectures and voice-overs. Authoring these recordings involves an iterative back-and-forth process between script writing/editing and audio recording/editing. Unfortunately, most existing tools treat the script and the audio separately, making the iterative workflow very tedious.

We present **VoiceScript**, an interface to support a dynamic workflow for script writing, audio recording and audio editing. VoiceScript integrates the script with the audio such that, as the user writes the script or records speech, edits to the script are translated to the audio and vice versa.

Through informal user studies, we demonstrate that VoiceScript greatly facilitates the audio authoring process in various scenarios.

Delivering Slide Presentations(Chapter 4):

Presentations are an important component of both classroom and online instruction, and presentation tools have a significant impact on how we teach and learn. Electronic slides, the dominant type of presentation technology today, have several shortcomings. Pre-authored slides take a long time to prepare and restrict how their contents are presented. There is little flexibility on the order and granularity of how information is displayed during the presentation.

We introduce **Aparecium**, a presentation interface that helps presenters deliver flexible and engaging presentations by combining the aesthetics and organization of electronic slides with the spontaneity of inking. In Aparecium, presenters use inking interactions to deliver slide presentations. With inking, presenters can (1) reveal pre-authored content to the audience, (2) make annotations on top of the slide, and (3) adjust the slide layout to create blank space. Pre-authored slides help improve the visual aesthetics and organization of the slides, while inking enables presenters to have flexibility and fine-grained control over the content and pace of the presentation.

In a user study comparing Aparecium with baseline presentation tools, we found that our interface generally improves presentation quality without increasing the burden on the presenter at authoring or presentation time. Especially for text-centered or step-by-step process-driven content, both audiences and presenters preferred presentations delivered using Aparecium.

Finally, Chapter 5 reviews our contributions and discusses insights gained from our work. The remainder of this chapter offers a brief overview of related work on multimedia interfaces.

1.5 RELATED WORK

Not surprisingly, **interfaces for multimedia** is a very broad subject that touches many fields of research, including human-computer interaction, computer vision, and computer graphics. Instead of attempting to compile a comprehensive summary of the subject, we broadly categorize previous research according to different phases of users' workflows–browsing, authoring and delivery–and briefly review work that is closely related to the applications we present in this dissertation. The works cited are not intended to be an exhaustive list, but rather serve as an illustrative guide to various topics.

1.5.1 *Browsing*

A main purpose of media browsing interfaces is to improve the user's efficiency by making it easier to find and absorb useful information–in short, to reduce the browsing time. Although specific techniques vary by media type and content, there are common strategies that apply across different media.

Displaying Compact Visual Representation:

Skimming and browsing are inherently visual tasks, and we perform them instinctively, for example, while we read or window-shop. Compact visual representations of media can facilitate navigation by taking advantage of our natural visual capacity.

Visual abstractions of a video can take many forms such as shorter video skims [32, 81], keyframes [67, 68], montages [114], collages [119] and panoramas [28, 80]. They can combine static images or videos with text [96]. Static representations can also take advantage of spatial layout to expose the structure of the video. For example, Boreczky et al. arrange selected keyframes into a comic style layout, representing important segments with larger frames [17] . Interactive techniques can enhance the summary, for example by allowing users to explore the video by zooming in to expose fine-scale temporal details [13].

Unlike video, audio is a medium that does not have a natural visual representation. Nevertheless, from early on graphical representations have been used to facilitate visual scanning and searching of audio, and in particular, speech [37, 109]. It has been shown that transcript texts are highly effective as an interface to support browsing [109, 122]. In some applications, accompanying visual media such as handwritten notes [113, 121] or video [55, 65] are also used to index audio.

Analyzing and Exposing Structure within Media:

In order to construct an effective visual representation, it is usually necessary to infer some type of structure from the media. For example, videos are summarized by extracting highlight scenes [23, 43], topical themes [125], or repetitive segments [31, 129]. Similarly, speech browsing interfaces take advantage of inferred speaker turns [124], emphasis [8] or pauses [9]. This information can be inferred from the main modality itself (e.g., extracting repetitive structure in music from its signal [33]), from accompanying modalities (e.g., extracting highlights of baseball games from audio [107]), or from additional sources (e.g., using the screenplay to infer the scene hierarchy of a movie [34, 98]). They can be computed automatically [33, 98, 107] or with the help of manual annotation [27, 97].

1.5.2 *Authoring*

Authoring covers a wide range of tasks from capturing and browsing the source to drafting and editing the final product. An important objective of authoring interfaces is to make this workflow from the initial source to the final output as fluid as possible. There are many considerations that go into supporting this goal. Here, we consider two themes that arise commonly across applications.

Translating between Modalities:

The modality of the final output is not necessarily the most convenient modality to interact with during the authoring process. Many studies attempt to facilitate authoring by enabling easier, less formal input modalities such as text, pen-interactions or gestures, and translating them into appropriate operations on the final product. For example, transcript texts are used to edit videos [15, 21] or audio recordings [105], hand-drawn sketches are translated into informal presentations [76] or website designs [79], and physical gestures are captured to create cartoons [90] or 3D animations [57].

The increasing availability of various types of interactive devices is also encouraging research in multimodal interfaces that support a combination of input modalities. For example, MultiPoint uses speech and pen to author presentation slides [112], while PixelTone combines speech and direct manipulation to edit photos [73]. These interfaces aim to support a more *human* way of interacting with digital media similar to our day-to-day communication methods. Recent survey papers [39, 117] provide a comprehensive review of this subject.

Analyzing and Exploiting Relationship between Media:

It is common to combine multiple raw sources (e.g., video footage, audio tracks) to compose a single output media. Similar to browsing that takes advantage of inherent structures within a single medium, editing can exploit inherent relationships between multiple streams of media. For example, Arev et al. infer the 3D spatial relationship between multiple cameras capturing a social event in order to produce a coherent cut video of the event [7]. Bryan et al. synchronize multiple-camera videos of a same event using audio features [19]. QuickCut facilitates editing of narrated videos by aligning audio narration with semantically relevant video segments by taking advantage of user annotations and the narration transcript [115]. Li et al. present a method to synchronize slideshows of paintings to accompanying music by analyzing their common emotional features [75]. Unlike work on video or audio summarization, these works deal with relational structures across multiple input sources.

1.5.3 *Delivery*

Once authored, most media is delivered *as-is* to the consumer in a static format. However, some media such as presentations involve a dynamic aspect in which the media can be delivered interactively in a live setting. Improving flexibility for live performance is a key focus in this area.

Supporting Flexible Modes of Interaction:

One strategy to improve flexibility during performance is to support flexible modes of interaction to control the media. For example, instead of predetermined linear presentations, Palette provides random access to slides using physical cards [92]. Similarly, zooming user interfaces allow flexible presentation paths through spatial navigation of slides [49, 77]. Baudel and Beaudouin-Lafon propose hand gestures as an intuitive mode to control slides [14]. Other work enable presenters to add improvised content by inking [6, 48]. A common concern across these work is providing an intuitive interaction mode that does not further burden the presenter at performance time.

Each of our works focuses on a different medium and a different part of the users' workflow. However, they are inspired by many of the same principles that we observe in previous work. In each chapter, we also include a review of work that is more directly related to the application being discussed.

2 NAVIGATING LECTURE VIDEOS

The day is coming when the work done by correspondence will be greater in amount than that done in the classroom of our academies and colleges.

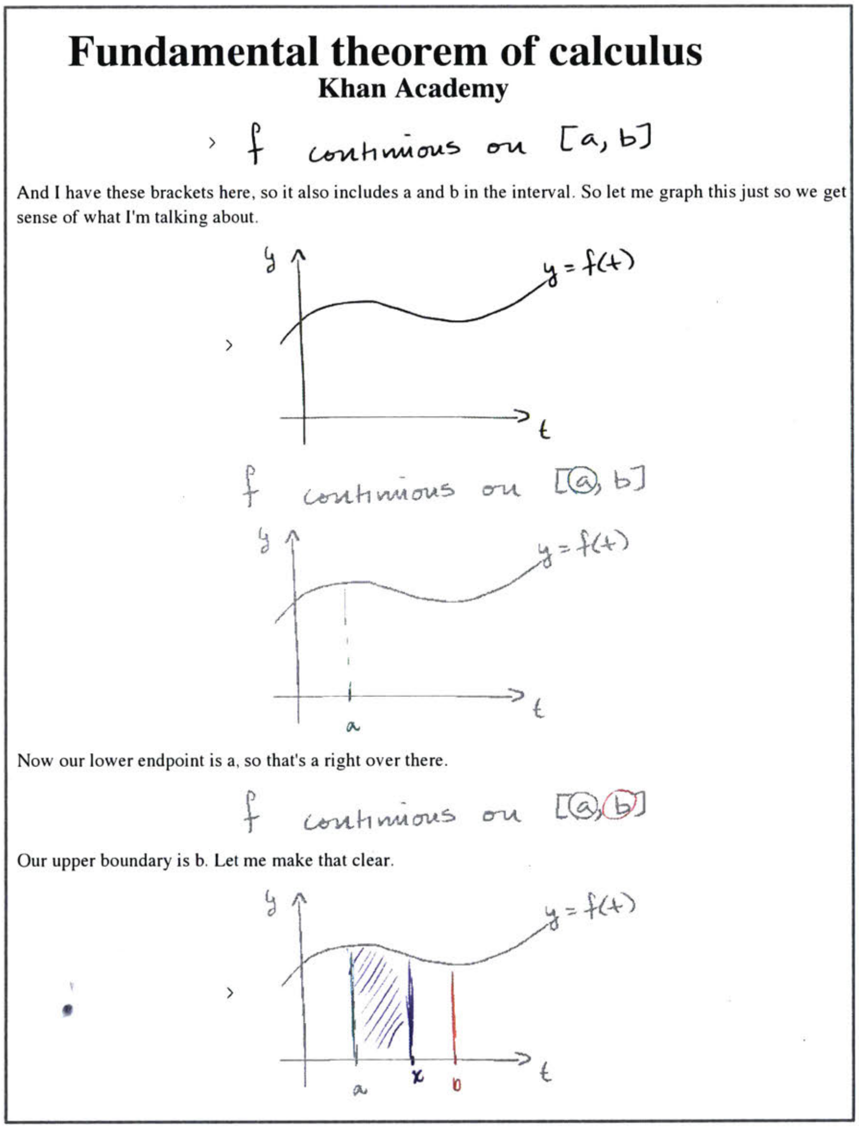
— William Rainey Harper [54]

How viewers watch videos depends on the content of the video and the viewers' needs. For example, watching a film in a theater is a very different experience from watching a video tutorial on YouTube about how to cook brussel sprouts. Even for the same video, someone who is watching it for the first time may have a different approach from someone who has seen the video previously and is watching it again only to review. In this chapter, we focus on the scenario of watching lecture videos.

Lecture videos are growing in popularity, especially through Massive Open Online Courses (MOOCs) and flipped classroom models. However, learning with these videos using existing video player interfaces can be challenging. Viewers cannot digest the lecture material at their own pace, and it is also difficult to search or skim the content. For these and other reasons, some viewers prefer lecture notes or textbooks to videos.

This chapter introduces **VisualTranscript**, a readable interface for lecture videos, that was designed to address some of these limitations. VisualTranscript resembles lecture notes combining figures and text. To generate VisualTranscripts, we take advantage of the spatial and temporal structures embedded in lecture videos. First, we segment the visual content of a lecture into a set of discrete illustrations that correspond to equations, figures, or lines of text. Then, we analyze the temporal correspondence between the transcript and the visuals to determine their relationships. Finally, based on the inferred relationships, we arrange the text and figures into a hierarchical and linear layout.

We compare VisualTranscript with a standard video player, and a state-of-the-art interface designed specifically for lecture videos. User evaluation suggests that users prefer VisualTranscript for the task of learning and that VisualTranscript facilitates browsing and searching in lecture videos.



Fundamental theorem of calculus

Khan Academy

› f continuous on [a, b]

And I have these brackets here, so it also includes a and b in the interval. So let me graph this just so we get sense of what I'm talking about.

f continuous on [a, b]

Now our lower endpoint is a, so that's a right over there.

f continuous on [a, b]

Our upper boundary is b. Let me make that clear.

Figure 5: Example of a VisualTranscript output from the Khan Academy video on the *'Fundamental theorem of calculus.'* Visual Transcript interleaves figures with short paragraphs of text, making it easy to skim or search through the lecture.

2.1 INTRODUCTION

Despite the increasingly important and broad role of lecture videos in education, learning from such videos poses some challenges. It is difficult for viewers to consume video content at their own pace [26]. To skip quickly through familiar concepts or slowly review more difficult material, the viewer must interrupt playback and scrub back-and-forth in the timeline. It is also difficult to find specific information in a video. While scrubbing allows users to browse the visual information in the lecture, it is not effective for skimming the audio content, which often includes critical explanations and context that accompany the visuals. As an alternative, some platforms (e.g., Khan Academy and YouTube) provide synchronized transcripts that allow users to click on a phrase and play the video at that location. However, skimming the transcript for relevant content can also be challenging since the text is not structured, and viewers must click on various parts of the text to see the corresponding visuals. Finally, it is hard to get a quick overview of the lecture content without watching the entire video. For these and other reasons, some people prefer static learning materials such as textbooks or printed lecture notes over videos.

Inspired by lecture notes, we present VisualTranscript, a readable interface for both the visual and audio content of a lecture video that facilitates reviewing, browsing and navigation. We focus on blackboard-style lectures. Here, we use the term *blackboard-style* to refer to lectures that are recorded on a tablet which show a possibly infinite blackboard (or whiteboard) where the instructor writes down by hand the content of the lecture. VisualTranscripts aggregate the full lecture content in a structured format where visual information is segmented and grouped with the corresponding narration text. For example, Figure 5 shows our automatically generated output for a math lecture that interleaves verbal explanations with corresponding equations written on the board. By default, VisualTranscripts hide redundant information to show a compact representation of the content that viewers can expand interactively to show relevant details. Presenting video content in this manner allows users to review the lecture at their own pace while getting both the visual and textual information in a readable, skimmable format. VisualTranscripts is also linked to the video such that clicking on the text or the visuals plays the video from the corresponding location. In this respect, VisualTranscripts offer many of the benefits of traditional static media, such as textbooks and lecture notes, while also giving viewers direct access to the video content.

There are two main challenges in transforming a video and its transcribed audio into a VisualTranscript: (1) visuals, which are drawn progressively on the board, must be discretized into a set of meaningful figures, and (2) such figures and text representing the audio content must be organized into a compact, structured format that emphasizes the relationships between the two channels of information. To segment the visuals into meaningful figures, we propose a dynamic programming approach that takes into account both the spatial layout of strokes and the time when they were drawn. We further time-align the transcript with the audio and use this alignment to establish correspondences between the visuals and the text. Finally, we use the visual-text correspondence to detect redundant information and arrange the content in a compact, sequential layout where the text is organized into readable paragraphs.

We evaluate our approach with a user study that compares VisualTranscript with a standard video player with an interactive transcript, and an existing state-of-the-art visual-based video player developed by Monserrat et al [86]. We measure performance on summarization and search tasks, and observe how the participants interact with the interfaces. Evaluation results suggest that VisualTranscript is indeed an effective interface for studying lecture videos. Specifically, users performed best using VisualTranscript for search tasks involving text. Users noted that VisualTranscript helped them to get a quick overview of the video including the details conveyed only through the text, and to efficiently focus on parts of interest. They also found the structured text easier to read and connect to relevant visuals than the baseline text-only transcript. In a post-study survey, users strongly preferred our interface for learning over the other two interfaces.

2.2 PREVIOUS WORK

2.2.1 *Video Visualization*

There is a large body of work that aims to automatically summarize videos to facilitate navigation and browsing, but most research focuses on live action footage which is very different from educational videos. Recent survey papers [18, 116] comprehensively review these techniques, which can be broadly divided into two classes according to their output: **video skims** and **still-image abstracts**.

Video skims [40, 56, 81, 93] summarize a longer video with a shorter video, usually consisting of segments extracted from the original video. These skims retain audio and motion elements and are especially useful for understanding dynamic scenes, but they are less suitable for conveying the dense, static information of blackboard-style lectures.

Still-image based methods [13, 17, 62, 118] primarily focus on conveying the visual content of a video in static form through a collection of salient images extracted from the video. [30] and [99] developed a still-image based method specific to news stories that combines text and images into summaries.

Most relevant to our work is [28], which summarizes blackboard-style lectures by creating a panoramic frame of the board. In addition to the visual content presented on the board, our interface includes the audio content and therefore maintains the sequence of the lecture and makes textual content also directly accessible.

2.2.2 *Tools for Online Lecture Videos*

Kim et al. use interaction data collected from MOOC platforms to introduce a set of techniques that augment existing video interface widgets [69]. For lecture videos based on slides, Li et al. use separate slides to automatically generate table-of-content overviews [69]. These works *annotate* the original video with useful data to facilitate navigation, but do not reformat the video content. Pavel et al. provide a tool to create *video digests*, structured summaries of informational videos organized into chapters and sections [97] . They use only the transcript to segment and summarize the video, whereas we leverage both the visual and audio content.

Most closely related to our work is Monserrat et al.'s interface [86], which presents a summary image of blackboard-style lecture videos. Their image is composed of click-able visual links to support spatial and temporal navigation. Although they provide a search box for the transcript, text is not included as part of their summary display.

2.3 DESIGN PRINCIPLES

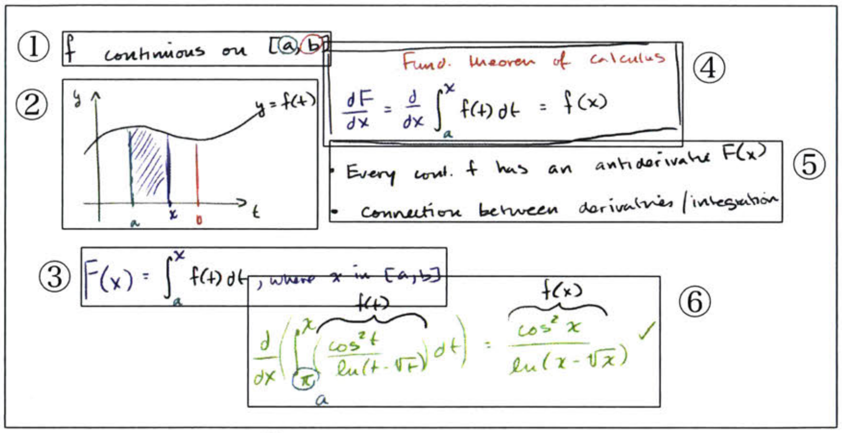
The design of VisualTranscript is informed by the following key characteristics of blackboard-style lectures:

Lectures present information progressively.

Most lectures convey concepts in a progressive manner where each new piece of information builds on the previously presented content. For example, Figure 6 (*top*) shows a panoramic image of the board for an entire lecture, where the labels show the order in which the contents were presented. Understanding the lecture often requires knowing this order. To emphasize presentation order, our VisualTranscript arranges all the content within the video in a top-to-bottom linear format.

Visuals are organized into discrete entities.

The visual content of a lecture is typically organized into well-defined entities (e.g., a line of text, an equation, an explanatory figure) that correspond to the set of presented concepts. For example, Figure 6 (*top*) shows six visual entities in a calculus lecture. Each visual en-


①
f continuous on [a, b]
②
y
y = f(t)
a
x
b
t
③
F(x) = ∫ₐˣ f(t) dt, where x in [a,b]
④
Fund. theorem of calculus
dF/dx = d/dx ∫ₐˣ f(t) dt = f(x)
⑤
• Every cont. f has an antiderivative F(x)
• Connection between derivatives / integration
⑥
f(t)
f(x)
d/dx (∫ₓᵃ (cos²t / ln(t - √t)) dt) = cos²x / ln(x - √x) ✓

- Depictive sentence for ①: *"Let's say I have some function f that is continuous on an interval between a and b."*
- Explanatory sentence between ② & ③: "*Well, how do we denote the area under the curve between two end points? Well, we just use our definite integral.*"

Figure 6: (top) Lectures convey concepts progressively. Here, the labels (1 through 6) show the order in which concepts were presented. They also organize visuals into discrete entities (outlined in this visualization with bounding boxes). (bottom) Verbal explanations during lectures are either depictive or explanatory.

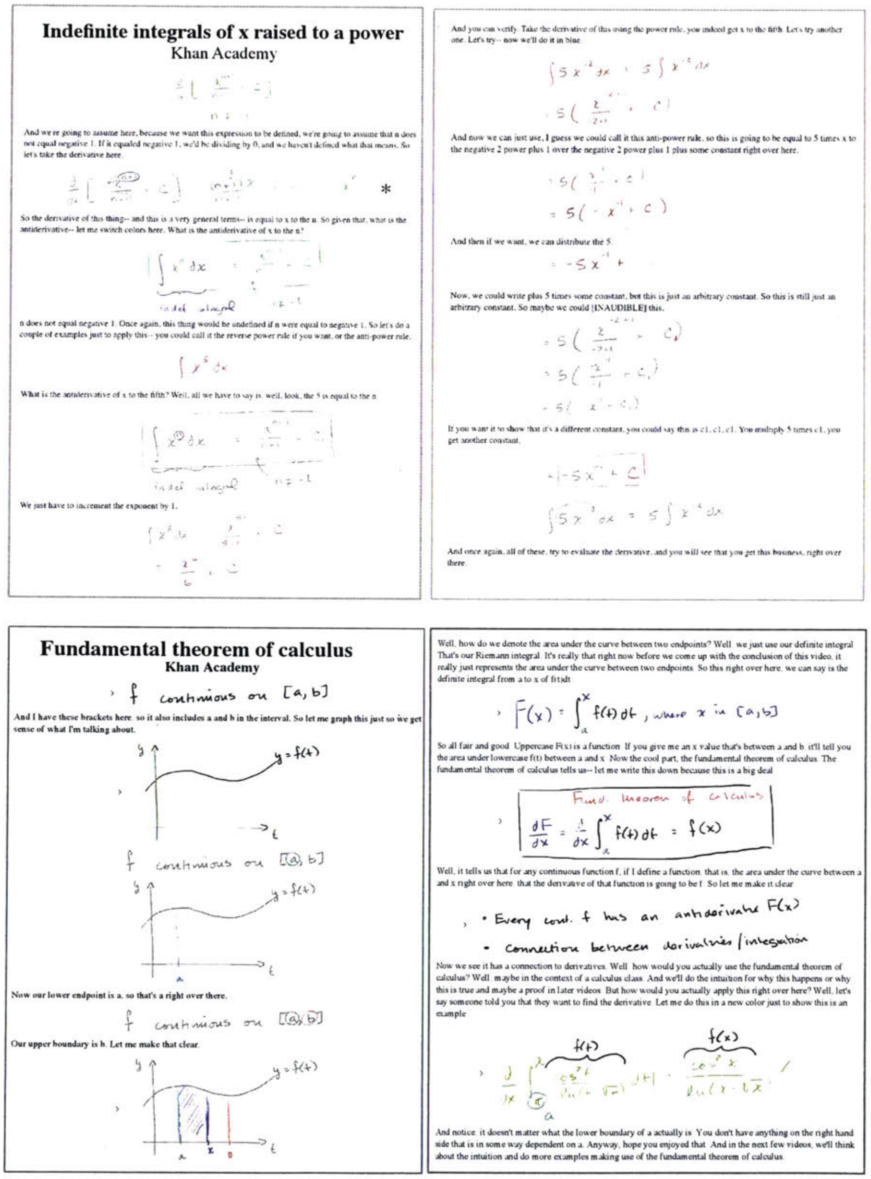
tity consists of strokes that are close together in either space or time. Moreover, since people are accustomed to parsing visual information line-by-line, from top to bottom, and left to right, visual entities are often laid out in the same manner. Building on this observation, our system segments drawings on the board into visual entities based on their spatial alignment and temporal proximity.

Audio content complements visuals.

In our analysis of lecture videos, we found that verbal explanations tend to serve one of two broad objectives. Explanations given while the instructor is *not* drawing are often *explanatory*, providing additional information not directly represented in the visuals or making connections between drawings. On the other hand, explanations given while the instructor is drawing are typically more *depictive*, repeating or reading aloud the visual information (Figure 6, *bottom*). While depictive explanations can help viewers follow along with the video, they often result in long, repetitive transcript text that is cumbersome to read or skim through. This problem is exacerbated by the fact that spoken explanations are usually colloquial.

Our interface automatically categorizes transcript text as either explanatory or depictive. In the default view, we hide depictive sentences and only show explanatory text interspersed with the set of visual entities extracted from the video. Large et al. [74] and Christel and Warmak [29] have shown that such combinations of pictures and captions aid recall and comprehension as well as navigation of video material. Our design gives the viewer relevant context for understanding the visual information without cluttering the output with redundant text. Users can click on a visual entity to see the corresponding depictive sentences.

2.4 THE VISUALTRANSCRIPT INTERFACE



Indefinite integrals of x raised to a power

Khan Academy

And we're going to assume here, because we want this expression to be defined, we're going to assume that n does not equal negative 1. If it equaled negative 1, we'd be dividing by 0, and we haven't defined what that means. So let's take the derivative here.

So the derivative of this thing-- and this is a very general terms-- is equal to x to the n. So given that, what is the antiderivative-- let me switch colors here. What is the antiderivative of x to the n?

n does not equal negative 1. Once again, this thing would be undefined if n were equal to negative 1. So let's do a couple of examples just to apply this-- you could call it the reverse power rule if you want, or the anti-power rule.

What is the antiderivative of x to the fifth? Well, all we have to say is, well, look, the 5 is equal to the n

We just have to increment the exponent by 1.

And you can verify. Take the derivative of this using the power rule, you indeed get x to the fifth. Let's try another one. Let's try-- now we'll do it in blue

And now we can just use, I guess we could call it this anti-power rule, so this is going to be equal to 5 times x to the negative 2 power plus 1 over the negative 2 power plus 1 plus some constant right over here.

And then if we want, we can distribute the 5.

Now, we could write plus 5 times some constant, but this is just an arbitrary constant. So this is still just an arbitrary constant. So maybe we could [INAUDIBLE] this.

If you want it to show that it's a different constant, you could say this is c1, c1, c1. You multiply 5 times c1, you get another constant.

And once again, all of these, try to evaluate the derivative, and you will see that you get this business, right over there.

Fundamental theorem of calculus

Khan Academy

And I have these brackets here. so it also includes a and b in the interval. So let me graph this just so we get sense of what I'm talking about.

Now our lower endpoint is a, so that's a right over there.

Our upper boundary is b. Let me make that clear.

Well, how do we denote the area under the curve between two endpoints? Well we just use our definite integral That's our Riemann integral. It's really that right now before we come up with the conclusion of this video, it really just represents the area under the curve between two endpoints So this right over here, we can say is the definite integral from a to x of f(t)dt

So all fair and good Uppercase F(x) is a function If you give me an x value that's between a and b, it'll tell you the area under lowercase f(t) between a and x Now the cool part, the fundamental theorem of calculus The fundamental theorem of calculus tells us-- let me write this down because this is a big deal

Fund. theorem of calculus
$\frac{dF}{dx} = \frac{d}{dx}\int_a^x f(t)\,dt = f(x)$

Well, it tells us that for any continuous function f, if I define a function that is, the area under the curve between a and x right over here that the derivative of that function is going to be f So let me make it clear

• Every cont. f has an antiderivative F(x)
• Connection between derivatives / integration

Now we see it has a connection to derivatives. Well how would you actually use the fundamental theorem of calculus? Well maybe in the context of a calculus class And we'll do the intuition for why this happens or why this is true and maybe a proof in later videos But how would you actually apply this right over here? Well, let's say someone told you that they want to find the derivative Let me do this in a new color just to show this is an example

And notice it doesn't matter what the lower boundary of a actually is You don't have anything on the right hand side that is in some way dependent on a. Anyway, hope you enjoyed that And in the next few videos, we'll think about the intuition and do more examples making use of the fundamental theorem of calculus

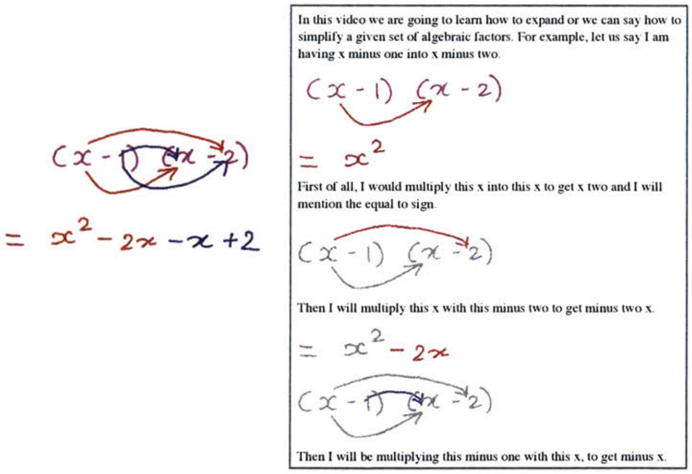
Figure 7: Examples of two VisualTranscript outputs from Khan Academy Videos. (Top)*'Indefinite integrals of x raised to a power'* and (Bottom)*'Fundamental theorem of calculus.* Users can click on a figure to expand step-by-step details. For example, the equation marked by '*' is expanded in Figure 10.

Based on these observations, we designed **VisualTranscript**, a readable and printable interface that presents both the visual and audio content of a lecture video. Since VisualTranscript contains all of the visual and audio information from the original video, it can be used by itself to study the content. Alternatively, it can be linked to the original lecture video to function as an interactive navigation aid. Similar to Monserrat et al's interface [86], clicking on a visual entity or a transcript sentence plays the original video at that point in time. As the video is playing the corresponding visual entity and transcript sentence are highlighted.

Figure 7 shows two examples of VisualTranscript created from Khan Academy videos. Please visit `https://people.csail.mit.edu/hishin/projects/visual_transcripts/abstract.html` for the interactive version of these examples. Given an input video and its transcript, we generate VisualTranscript automatically using the algorithms described in the next section. To test the robustness of our method, we generated VisualTranscripts of 20 lecture videos from 11 different instructors. Please view the appendix for all the results. Here we highlight some of the key features of VisualTranscript.

2.4.0.1 *Linear format highlights lecture progression.*

The layout of text and visual entities in VisualTranscript often emphasizes the instructor's thought process and clarifies the intermediate steps that lead to a result. Figure 8 compares equations in the final view of the blackboard at the end of the lecture with a VisualTranscript output.



$(x-1)(x-2)$

$= x^2 - 2x - x + 2$

In this video we are going to learn how to expand or we can say how to simplify a given set of algebraic factors. For example, let us say I am having x minus one into x minus two.

$(x-1)(x-2)$

$= x^2$

First of all, I would multiply this x into this x to get x two and I will mention the equal to sign.

$(x-1)(x-2)$

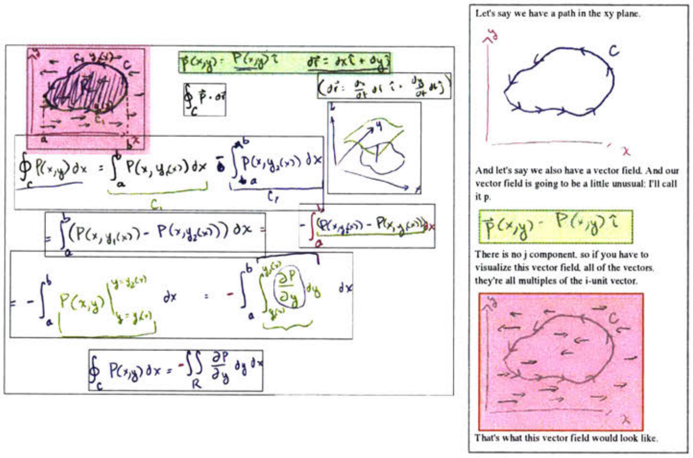
Then I will multiply this x with this minus two to get minus two x.

$= x^2 - 2x$

$(x-1)(x-2)$

Then I will be multiplying this minus one with this x, to get minus x.

Figure 8: (Right) VisualTranscript shows the step-by-step progression of an equation which is not apparent in the (left) final view of the board.



Let's say we have a path in the xy plane.

And let's say we also have a vector field. And our vector field is going to be a little unusual: I'll call it p.

There is no j component, so if you have to visualize this vector field, all of the vectors, they're all multiples of the i-unit vector.

That's what this vector field would look like.

Figure 9: (Right) Interspersing text with visuals clarifies the connection between the illustration of a vector field (pink highlight) and its equation (green highlight). Compare with the (left) final view of the original board.

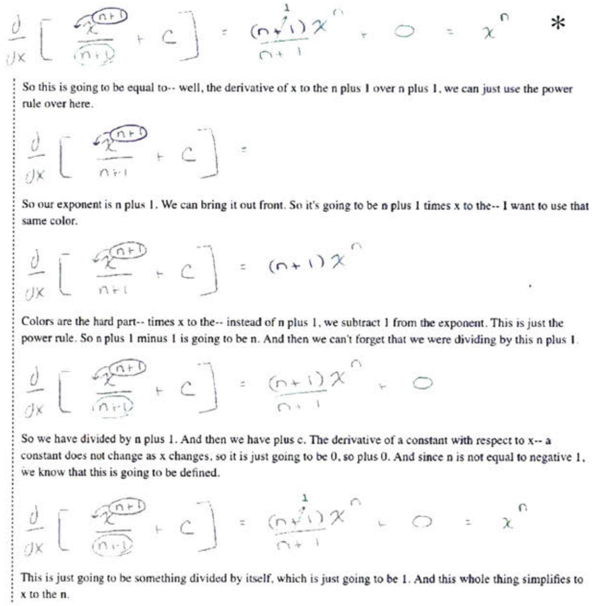
Although both views show the same set of equations, in the blackboard view (left) it is difficult to infer how the equations relate to and build upon each other. In contrast, the VisualTranscript output (right) shows a step-by-step progression of the visual content.

2.4.0.2 *Interspersing text with visuals clarifies connections.*

A purely visual summary of the video omits verbal explanations, whereas a purely textual summary (i.e., transcript) can be confusing without the corresponding visuals. Instead, VisualTranscript interleaves explanatory text and visual entities. This makes it easy to see the connection between illustrations, or the context of an illustration. For instance, compare the final view of the blackboard (Figure 9 left) and the VisualTranscript for the same video (Figure 9 right). In the former, it is difficult to see the connection between the illustration (pink highlight) and the equation to its right (green highlight) without listening to the lecture. In the latter, the text in-between explains clearly that the equation represents the vector field depicted in the illustration.

2.4.0.3 *Hierarchical organization show different levels of detail.*

By default, VisualTranscript hides redundant depictive text that just describes the corresponding visuals. Users can click on a visual entity to reveal the corresponding hidden text and read the details. In the case of a long equation or a complicated illustration, the expanded view breaks up the visual and textual information into easy-to-read blocks (Figure 10).



$$\frac{d}{dx}\left[\frac{x^{n+1}}{n+1}+c\right]=\frac{\overset{1}{\cancel{(n+1)}}x^{n}}{n+1}+0=x^{n}$$ *

So this is going to be equal to-- well, the derivative of x to the n plus 1 over n plus 1, we can just use the power rule over here.

$$\frac{d}{dx}\left[\frac{x^{n+1}}{n+1}+c\right]=$$

So our exponent is n plus 1. We can bring it out front. So it's going to be n plus 1 times x to the-- I want to use that same color.

$$\frac{d}{dx}\left[\frac{x^{n+1}}{n+1}+c\right]=(n+1)x^{n}$$

Colors are the hard part-- times x to the-- instead of n plus 1, we subtract 1 from the exponent. This is just the power rule. So n plus 1 minus 1 is going to be n. And then we can't forget that we were dividing by this n plus 1.

$$\frac{d}{dx}\left[\frac{x^{n+1}}{n+1}+c\right]=\frac{(n+1)x^{n}}{n+1}+0$$

So we have divided by n plus 1. And then we have plus c. The derivative of a constant with respect to x-- a constant does not change as x changes, so it is just going to be 0, so plus 0. And since n is not equal to negative 1, we know that this is going to be defined.

$$\frac{d}{dx}\left[\frac{x^{n+1}}{n+1}+c\right]=\frac{\overset{1}{\cancel{(n+1)}}x^{n}}{n+1}+0=x^{n}$$

This is just going to be something divided by itself, which is just going to be 1. And this whole thing simplifies to x to the n.

Figure 10: VisualTranscript is organized hierarchically. The default view hides descriptive details to present a compact summary (Figure 7*). Users can click on a higher-level figure to expand its detailed description and step-by-step derivation.

2.5 ALGORITHMS

There are three main steps to create VisualTranscript. First, we segment the visual content of a lecture into visual entities using a dynamic programming approach (2.5.2). Then, we structure the transcript content by computing temporal correspondences between visual entities and transcript sentences (2.5.3). Finally, we generate a VisualTranscript by interleaving visual entities with transcript text (2.5.4). The rest of this section describes each of these steps in detail.

2.5.1 *Pre-processing*

First, we use a simple intensity threshold to separate the foreground (ink) and background (board) pixels. The visual content in blackboard-style lectures consists of *strokes*, the set of foreground pixels generated during one continuous drawing action. In the context of a graphics tablet, a stroke corresponds to the continuous path of a pen while maintaining contact with the writing surface. Since our input is a recorded video and we do not have the vector information of the strokes, instead we extract individual strokes from the video frames using a method similar to [86]. We detect the start and end time of each drawing action by comparing the number of foreground pixels in consecutive frames. A large increase marks the start of an action, while no change marks the end. The difference image between the end and start frames gives an image of the stroke drawn during that period. The manual steps involved in this process are (1) identifying the cursor image, which is automatically removed from all frames, (2) setting a threshold for foreground/background separation, and (3) setting a smoothing window to get rid of the noise in the foreground pixel count. Depending on the instructor's writing speed, a typical stroke comprises several characters to several words, or it can also be a part of an illustration or a graph (Figure 11).

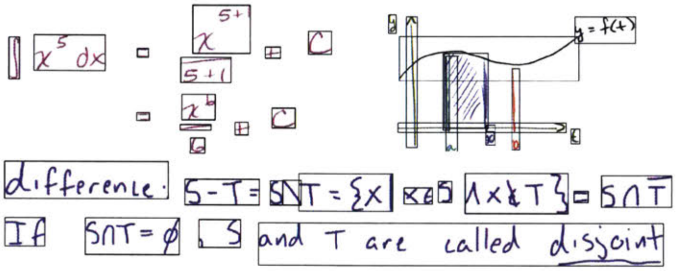


Figure 11: Examples of strokes (marked by black bounding boxes) extracted from video frames.

In addition to the visuals, lecture videos include an audio track with the instructor's spoken explanations. Several on-line video lecture platforms (e.g. Khan Academy, YouTube) provide transcripts of the audio. We assume such transcripts and we use an online audio transcription service (`castingwords.com`) if they are not available.

2.5.2 *Segmenting Visual Content*

One straightforward strategy for grouping strokes into visual entities is to process strokes in the order they are drawn and decide whether each stroke represents the start of a new visual entity or is part of an existing visual entity formed by previous strokes [88]. While this simple, greedy approach works in some cases, there are many scenarios where it leads to poor segmentations. For example, in Figure 13, there is a large space between the first stroke ($\int_{y_1(x)}^{y_2(x)}$, ①) and the second stroke (dx, ②). Without considering the semantics of these symbols, they appear to be separate equations. However, once we consider the subsequent set of red strokes(③) it becomes clear that this is not the best segmentation. In general, computing good stroke segmentations requires considering the global configuration of strokes in both space and time.

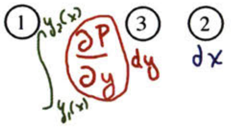


Figure 13: Without considering the semantics of these symbols and before the last stroke ($\frac{\delta p}{\delta y}$ dy) is inserted, the first two strokes ($-\int_b^a$ and dx) appear like two separate equations with a large space between them.

In this respect, the problem of segmenting strokes into visual entities is analogous to the *line-breaking* problem, i.e., arranging the words of a paragraph into lines. In both cases, we want to segment a sequence of elements (strokes or words) into an optimal set of groups (visual entities or lines) defined by some scoring function over candidate entities or lines. An important difference is that in the traditional line-breaking problem, only a contiguous set of words can be put on the same line. In our case, strokes in one visual entity can be interspersed by strokes in a different visual entity. For example, the instructor may go back and forth between two lines of equations, or between a graph and an equation (Figure 14).

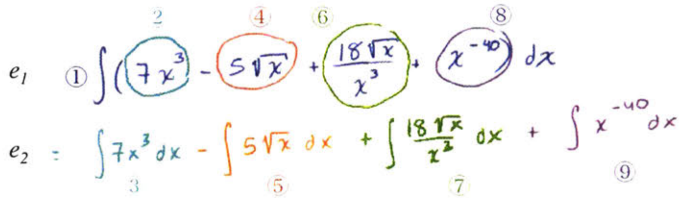


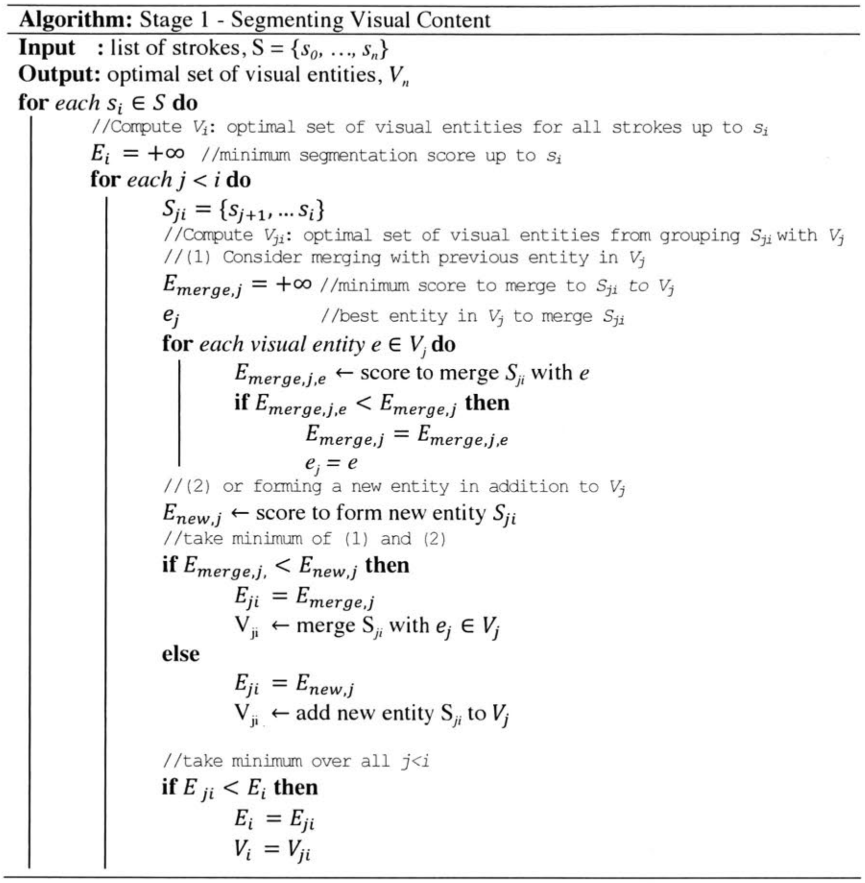
Figure 14: The instructor goes back and forth between writing two lines, e_1 and e_2. The order of strokes 1-9 is as indicated.

Given these observations, we propose a heuristic inspired by the dynamic programming approach based on the classic optimal line-breaking algorithm [70]. Unlike the classic approach, our algorithm handles non-contiguous grouping of elements. Here, we describe the high-level structure of the algorithm, and in the next section we define the scoring function in detail.

2.5.2.1 *Algorithm Overview*

Figure 15 gives a detailed pseudo-code of our segmentation algorithm.

Given a sequence of n strokes $S = \{s_0, \ldots, s_{n-1}\}$ ordered by when they appear in the video, we find the optimal set of inter-stroke boundaries that segment the strokes into visual entities. We refer to the boundary between s_i and s_{i+1} as b_i. Our algorithm processes the strokes in order and for each s_i computes and records the optimal set of visual entities V_i formed by all strokes up to b_i, along with the total score $E(V_i)$ of this partial solution. To determine the optimal partial solution for stroke s_i, we consider each previous boundary b_j where $j<i$, and evaluate two possible ways of grouping the set of strokes $S_{j,i} = \{s_{j+1}, \ldots, s_i\}$: 1) merging $S_{j,i}$ with one of the existing entities in V_j, or 2) forming a new entity with $S_{j,i}$. Allowing $S_{j,i}$ to be merged with existing entities enables our algorithm to support non-contiguous stroke groupings. We take the better (lower) of the two scores for $S_{j,i}$ and add it to $E(V_j)$ to obtain the total score for the proposed segmentation. After considering all candidate boundaries b_j, we identify the partial solution with the minimum segmentation



```
Algorithm: Stage 1 - Segmenting Visual Content
Input  : list of strokes, $S = \{s_0, \ldots, s_n\}$
Output: optimal set of visual entities, $V_n$
for each $s_i \in S$ do
    //Compute $V_i$: optimal set of visual entities for all strokes up to $s_i$
    $E_i = +\infty$ //minimum segmentation score up to $s_i$
    for each $j < i$ do
        $S_{ji} = \{s_{j+1}, \ldots s_i\}$
        //Compute $V_{ji}$: optimal set of visual entities from grouping $S_{ji}$ with $V_j$
        //(1) Consider merging with previous entity in $V_j$
        $E_{merge,j} = +\infty$ //minimum score to merge to $S_{ji}$ to $V_j$
        $e_j$            //best entity in $V_j$ to merge $S_{ji}$
        for each visual entity $e \in V_j$ do
            $E_{merge,j,e} \leftarrow$ score to merge $S_{ji}$ with $e$
            if $E_{merge,j,e} < E_{merge,j}$ then
                $E_{merge,j} = E_{merge,j,e}$
                $e_j = e$
        //(2) or forming a new entity in addition to $V_j$
        $E_{new,j} \leftarrow$ score to form new entity $S_{ji}$
        //take minimum of (1) and (2)
        if $E_{merge,j,} < E_{new,j}$ then
            $E_{ji} = E_{merge,j}$
            $V_{ji} \leftarrow$ merge $S_{ji}$ with $e_j \in V_j$
        else
            $E_{ji} = E_{new,j}$
            $V_{ji} \leftarrow$ add new entity $S_{ji}$ to $V_j$

        //take minimum over all j<i
        if $E_{ji} < E_i$ then
            $E_i = E_{ji}$
            $V_i = V_{ji}$
```

Figure 15: We use a variation of dynamic programming to segment strokes into an optimal set of visual entities. For each stroke, s_i, the algorithm considers all previous partial solutions, $V_{j<i}$ and $S_{j,i} = \{s_{j+1}, \ldots, s_i\}$. For each V_j, it considers two possibilities: merging $S_{j,i}$ with an existing entity or forming a new entity.

score and record the corresponding set of entities as V_i and the score as $E(V_i)$. Once the algorithm iterates through all strokes, V_{n-1} gives the optimal set of visual entities for the entire lecture.

Note that unlike the classic line-breaking algorithm, our problem does not have a optimal substructure property. We merely use overlapping subproblems to approximate the solution. Our algorithm does not guarantee an optimal solution, but in practice we found the segmentation outputs to be reasonably effective across different videos.

2.5.2.2 *Scoring Function*

The heuristic algorithm described above requires a scoring function that evaluates the goodness of candidate visual entities formed by sets of strokes. We define this scoring function based on several observations: Strokes within a visual entity are (1) compactly arranged and (2) horizontally aligned. In addition, (3) separate visual entities are spatio-temporally distant from each other.

(1) VISUAL ENTITIES ARE COMPACT. Strokes that belong together in the same visual entity are typically arranged in a compact way. We consider two measures of compactness for a visual entity: horizontal and vertical.

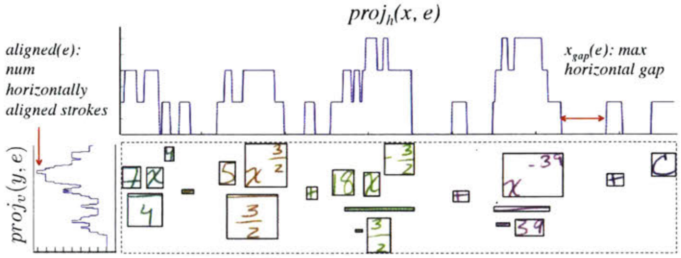

$proj_h(x, e)$
aligned(e): num horizontally aligned strokes
$x_{gap}(e)$: max horizontal gap
$proj_v(y, e)$

Figure 16: Horizontal (proj_h) and vertical (proj_v) projection functions of strokes in a line. In this example, $y_{gap}(e) = 0$.

- **Horizontal Compactness:** Intuitively, horizontal compactness is related to the horizontal *gap* between the strokes within a visual entity. Figure 16 shows an illustration of how gaps between strokes are measured. First, we define a horizontal projection function for the set of strokes in a visual entity, e, as

$$\mathrm{proj}_h(x, e) = \left|\{s \in e | x_{min}(s) \leqslant x \leqslant x_{max}(s)\}\right| \tag{1}$$

where $x_{min}(s)$, and $x_{max}(s)$ are the minimum and maximum x-coordinates of the bounding box of stroke s respectively. Intuitively, $\mathrm{proj}_h(x, e)$ counts the number of strokes in e which crosses the vertical line defined by x. Then, the maximum horizontal gap of a visual entity e is

$$x_{gap}(e) = \underset{x_i, x_{i+1}}{\mathrm{argmax}}(x_{i+1} - x_i) \tag{2}$$

where x_i and x_{i+1} are distinct consecutive elements in the ordered set $X = \{x \mid \mathrm{proj}_h(x, e) \neq 0\}$. We observed that the horizontal gap between different visual entities is usually around 100 pixels or more, so we define a horizontal compactness term C_h that imposes harsher penalties when the maximum horizontal gap exceeds this distance.

$$C_h(e) = \left(\frac{x_{gap}(e)}{100}\right)^2 \tag{3}$$

- **Vertical Compactness:** Vertical compactness is defined similarly in terms of a vertical projection function, $\mathrm{proj}_{v(y,e)}$, the maximum vertical gap, $y_{gap}(e)$, and a typical vertical gap of 40 pixels between different visual entities.

$$C_v(e) = (\frac{y_{gap}(e)}{40})^2 \tag{4}$$

(2) STROKES WITHIN A VISUAL ENTITY ARE ALIGNED HORIZONTALLY. With the exception of certain illustrations such as graphs, the strokes in most visual entities are horizontally aligned (e.g., equations, lines of text). Thus, we prefer to group horizontally aligned strokes into a single entity. The number of horizontally aligned strokes in each visual entity is computed by taking the maximum of its vertical projection function (Figure 16).

$$\mathrm{aligned}(e) = \underset{y_{min}(e) \leqslant y \leqslant y_{max}(e)}{\operatorname{argmax}} \quad \mathrm{proj}_v(y, e) \tag{5}$$

We then define an alignment term C_a whose contribution gradually diminishes with the total number of aligned strokes.

$$C_a(e) = \mathrm{aligned}(e) - \frac{1}{\mathrm{aligned}(e) + 1} \tag{6}$$

(3) DISTINCT VISUAL ENTITIES ARE SPATIO-TEMPORALLY DISTANT FROM EACH OTHER. This observation is complementary to the first observation, i.e., visual entities are compact. Whereas strokes that belong together are written close together, instructors usually leave some space on the board, for example, between lines of equations or separate illustrations. We express this property by penalizing any overlap between distinct visual entities, measured by the overlapping area between their bounding boxes. In particular, we define the overlap penalty term

$$P_o(V) = \sum_{e_i, e_j \in V, i \neq j} (\frac{\mathrm{area}(e_i \cap e_j)}{\min(\mathrm{area}(e_i), \mathrm{area}(e_j))}) \tag{7}$$

A similar property holds in the temporal domain. For example, after writing a single line of an equation and before going on to the next line, there is a brief pause while the instructor moves the cursor to the next position or provides some verbal explanation. We compute the temporal distance between two consecutive strokes across visual entity boundaries.

$$t_{dist}(s_i, s_{i+1}) = \begin{cases} 0, \text{if } s_i, s_{i+1} \text{ belong to the same visual entity} \\ \mathrm{start}(s_{i+1}) - \mathrm{end}(s_i), \text{otherwise} \end{cases}$$

where $\mathrm{start}(\cdot)$ and $\mathrm{end}(\cdot)$ are the start and end times of when a stroke is drawn in the video. We penalize visual entity boundaries with a small temporal gap.

$$P_t(V) = \sum_{i=0}^{n-1} \frac{1}{t_{dist}(s_i, s_{i+1})} \tag{8}$$

where n is the total number of strokes.

COMBINING SCORING TERMS. So far, we have defined terms that measure the compactness (C_h, C_v) and horizontal alignment (C_a) of an individual visual entity e, as well as the spatio-temporal distance (P_o, P_t) between a set of candidate entities V. We combine all these terms into a single scoring function F as follows.

$$F(V) = \sum_{e \in V} [C_v(e) + 0.5C_h(e) - C_a(e)] \tag{9}$$

$$+ P_o(V) + P_t(V) \tag{10}$$

The factor of 0.5 puts a smaller weight on horizontal versus vertical gaps. Higher values of C_a indicate more horizontally aligned strokes and better segmentation, so we put a minus in front.

The final output of our algorithm is a grouping of all the strokes on the board into a set of meaningful visual entities (Figure 17). To test the robustness of our segmentation algorithm, we applied it to 20 video lectures from 10 different authors, using the same set of parameters as described above. The lectures included nonlinear layouts of visual content and examples of complex diagrams with several layers of information. In all cases, the algorithm produced reasonable segmentations which generated comprehensible VisualTranscripts. There were few cases (≈ 5%) where the output segmentation was less than ideal, but these did not affect the overall quality of the VisualTranscripts. Please see the discussion on section 2.7 for more details. The full set of results are included in the appendix.

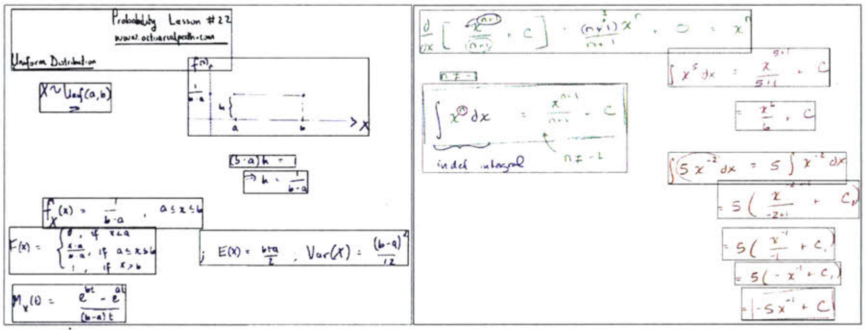


Figure 17: Examples of visual entities output from our line-breaking algorithm. Our algorithm successfully identifies meaningful groups even from complex layouts with a mix of equations, figures and graphs.

2.5.3 *Structuring Transcript Content*

Once we have segmented the visual content of the lecture, the next step is to organize the transcript text with respect to the extracted visual entities. We leverage temporal correspondences between the transcript and visuals to distinguish between explanatory and depictive sentences (section 2.3) and to break long text descriptions into shorter, more readable paragraphs.

2.5.3.1 *Aligning transcript to video.*

To obtain the temporal alignment between the transcript and video, we use an automatic algorithm by [105] which extends the Penn Phonetics Lab Forced Aligner (P2FA) built on the HTK speech recognition software. This aligner takes a verbatim transcript and an audio file as inputs and outputs a time-stamped transcript, where each word is annotated with a start and end time.

2.5.3.2 *Detecting explanatory versus depictive sentences.*

As discussed in section 2.3 about design principles, depictive sentences typically coincide with drawing actions while explanatory sentences do not. Using the time-aligned transcript, we compute correspondences between transcript sentences and visual entities. A sentence is matched to a single visual entity if most of its utterance time ($\geqslant$ 75%) overlaps with the drawing time of the visual entity. If a sentence does not coincide with any entity, we refer to it as an unmatched sentence. We classify all matched sentences as depictive text (associated with the corresponding visual entities) and all unmatched sentences as explanatory text. Note that while this is a heuristic, we found it to work well in practice. We use this information in the layout stage to reduce clutter and make the text more readable.

2.5.3.3 *Breaking up long text descriptions.*

In some cases, complex visual entities that contain a lot of information may get matched with large blocks of depictive text. When reading such text blocks, it can be hard to identify and follow all the correspondences between the individual sentences and the relevant parts of the figure. We address this problem by breaking up complex visual entities into sub-entities, each of which has a shorter, more readable block of depictive text.

In particular, we use a variant of the stroke segmentation algorithm described in the previous section to further segment a complex visual entity e. In this case, we use the following scoring function F_{sub} to evaluate a set of candidate sub-entities, V_{sub}:

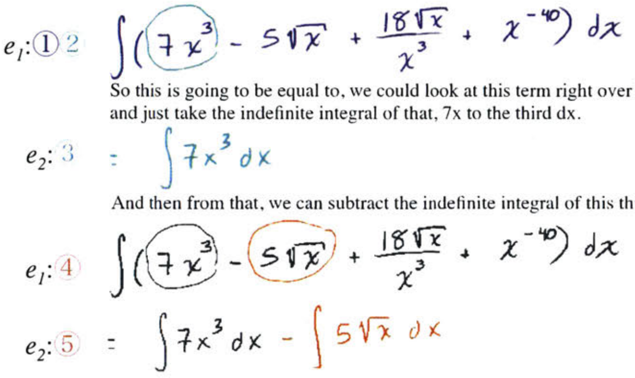
$$F_{\text{sub}}(V_{\text{sub}}) = \sum_{e_{\text{sub}} \in V_{\text{sub}}} \lambda_1 |n_{\text{words}}(e_{\text{sub}}) - w| + P_o(V_{\text{sub}}) \qquad (11)$$

where $n_{\text{words}}(e_{\text{sub}})$ is the number of words in the depictive text associated with sub-entity, e_{sub}; P_o is the overlap between bounding boxes of sub-entities in V_{sub} (defined in Equation 7); w is the target number of words in the depictive text for each sub-entity; and λ_1 determines the relative importance of the word count and overlap terms. We set $w = 50$ (about 2-4 sentences) and $\lambda_1 = 1/25$.

Using this scoring function, we apply the same dynamic programming procedure described in section 2.5.2 to segment e into sub-entities. In this variant, we only allow consecutive strokes to be grouped together since our goal is to obtain temporally sequential sub-entities. Figure 10 shows an example output from this optimization.

2.5.4 *Layout and Formatting*

We organize the visual and audio content into a static, sequential format by interleaving visual entities with blocks of transcript text in the order of their appearance in the video. As we point out in section 2.5.2, a single visual entity can be composed of non-contiguous groups of strokes. For example, in Figure 14, e_1 and e_2 each consist of 4 separate groups of strokes, (1&2, 4, 6, 8) and (3, 5, 7, 9) respectively. In this case, we show each contiguous group of strokes at its associated time, together with previous strokes in the same visual entity which are shown for context. So Figure 14 would be presented as: 1&2, 3, (1&2)&4, (3)&5 etc., where the parentheses indicate previous strokes. The new group of strokes is highlighted with color on top of the previous strokes (Figure 18).



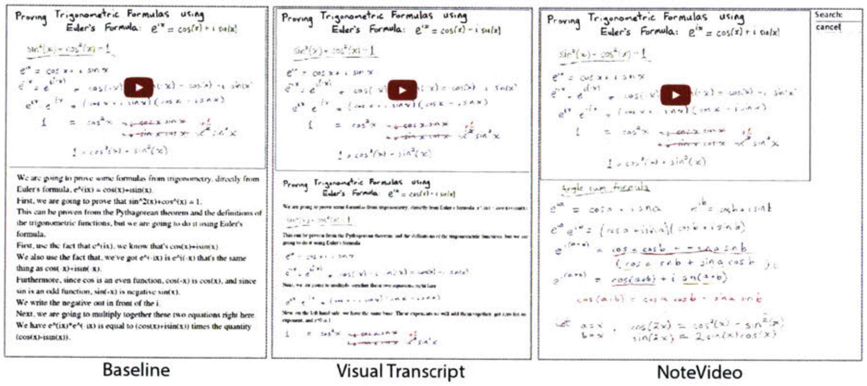
So this is going to be equal to, we could look at this term right over here, and just take the indefinite integral of that, 7x to the third dx.

And then from that, we can subtract the indefinite integral of this thing.

Figure 18: VisualTranscript presentation of strokes 1-5 of Figure 14. Each contiguous group of strokes is shown together with previous strokes in the same visual entity.

By default, all visual entities and explanatory sentences are shown; the depictive text associated with visual entities is hidden to reduce clutter. Users can click on the *expand* buttons next to individual visual entities to display the corresponding depictive sentences. For complex visual entities, the expanded view shows the decomposed sub-entities with their associated depictive sentences (Figure 10).

2.6 USER EVALUATION


Proving Trigonometric Formulas using Euler's Formula: $e^{ix} = \cos(x) + i\sin(x)$
Search:
cancel
Baseline
Visual Transcript
NoteVideo

We are going to prove some formulas from trigonometry, directly from Euler's formula, e^(ix) = cos(x)+isin(x).
First, we are going to prove that sin^2(x)+cos^(x) = 1.
This can be proven from the Pythagorean theorem and the definitions of the trigonometric functions, but we are going to do it using Euler's formula.
First, use the fact that e^(ix), we know that's cos(x)+isin(x).
We also use the fact that, we've got e^(-ix) is e^i(-x) that's the same thing as cos(-x)+isin(-x).
Furthermore, since cos is an even function, cos(-x) is cos(x), and since sin is an odd function, sin(-x) is negative sin(x).
We write the negative out in front of the i.
Next, we are going to multiply together these two equations right here.
We have e^(ix)*e^(-ix) is equal to (cos(x)+isin(x)) times the quantity (cos(x)-isin(x)).

Figure 19: We compared three interfaces to study video lectures: A standard YouTube player with an interactive transcript, our VisualTranscript, and NoteVideo.

We performed a comparative study to test the hypothesis that VisualTranscript facilitates learning. We compared three interfaces to study video lectures: a standard YouTube player with an interactive transcript (Baseline), Monserrat et al.'s interface [86] (hereafter referred to as **NoteVideo**), and our VisualTranscript interface linked to the video (Figure 19). The YouTube video player is currently the most common viewing interface for online lectures, and NoteVideo while less established, was specifically designed to facilitate navigation of blackboard-style lecture videos. In NoteVideo, a panoramic image of the board with strokes from the entire lecture serves as an in-scene navigation interface. Users can click on any stroke to play the video at that point in time.

2.6.1 *User Tasks*

Our study includes two tasks: (1) summarization, to get a quick and comprehensive overview of the lecture without watching the entire video, and (2) search, to quickly locate specific information. Although not a direct measure of learning, these tasks are inherent activities in learning and also match common evaluation tasks used in the literature on tools for lecture videos [69, 86, 97].

2.6.1.1 *Summarization Task*

Users were asked to quickly provide an overview of the lecture without watching the entire video. We gave users only 3 minutes to view and summarize 7-8 minute long lectures. We purposely did not give enough time to watch the entire video so as to motivate the users to quickly scan through its content. Before the task, users watched a sample lecture video and read a sample summary comprised of main points and detail points. Users were encouraged to try to write down at least all the main points of the lecture, and as many of the detail points as possible.
A judge compared the summaries written by the users to a gold standard list of main/detail points manually created by two referees. The judge was blind as to which summaries were written using which interface. The user summaries were scored by the number of points they covered.

2.6.1.2 *Search Task*

The search task emulates scenarios when the user wants to quickly find a specific piece of information in the video (e.g. to solve a question in a problem set, or to look up a specific formula). We differentiate three different types of search problems depending on whether the information is in the visuals, the transcript or a combination of both.

The **visual search** reproduces situations when a user remembers something visually and wants to find where it appeared. (E.g., *Find the point in the lecture where the instructor strikes out part of an equation, where terms add up to eliminate each other.*) For the **textual search**, the cue is often a word or a phrase that could be found in the transcript either directly or indirectly (E.g., *Find the point in the lecture where the property that every continuous function has an antiderivative is stated.*) For the **contextual search**, the information is neither in the text nor visuals alone, but rather in the context between the two. (E.g., *Find the point in the lecture where the instructor writes an integral expression for a bounded area under some curve.*) User performance was assessed by the task completion time as well as correctness.

Nine participants (2 female, 7 males), ages 20 to 35 were recruited using an university e-mail list. All of them were familiar with the general subject matter of the lectures, although they had not seen the particular lectures before.

We chose three college-level math lectures for our study: *Fundamental Theorem of Calculus* by Salman Khan (8 minutes), *Proving Trigonometry Formulas Using Euler's Formula* by Lee Stemkoski (7.2 minutes),

and *Uniform Distribution* by Actuarialpath (8 minutes).
We used a within-participant design, where each participant performed tasks on each interface. We counter-balanced the order of the interfaces and the assignment of videos to interfaces using a Latin Square.

Before using each interface, participants were briefed about their features and given time to familiarize themselves. After each task, they answered questions about their interaction with the interface. After completing all tasks, participants completed a questionnaire on their preference and the usability of each interface. Please refer to the appendix for the full set of tasks and post-task questionnaires.

2.6.2 *Findings and Discussion*

There are several notable findings from our user study:

2.6.2.1 *Users write more comprehensive summaries with VisualTranscript.*

Users listed the most number of main and detail points using our interface, although differences across the interfaces were not statistically significant according to the one-way analysis of variance (ANOVA) (main points: $F_{2,24} = 0.23$, $p = 0.79$, detail points: $F_{2,24} = 0.48$, $p = 0.62$). Table 1 shows the percentage of main/detail points covered by user summaries with each interface.

	Baseline	**NoteVideo**	**Ours**
Main points	0.83±0.12	0.81±0.21	0.87±0.18
Detail points	0.50±0.22	0.56±0.18	0.58±0.15

Table 1: Percentage of points covered by user summaries compared to the golden standard list.

Note that while on average, there may not seem to be a significant difference between ours and the two alternatives, summary quality varied significantly depending on the video. In particular, when the sequence of lecture was not clear in the panoramic image, NoteVideo users mixed the order of points or missed a main point entirely. For example, in the lecture on *Fundamental Theorem of Calculus* (Figure 6 bottom), NoteVideo users immediately clicked on the "Fundamental Theorem" (④) skipping the first third of the lecture about the graph of a continuous function and the area under it (①-③). While users performed comparably with VisualTranscript or the baseline, when asked which interface they preferred for the summary task, they preferred NoteVideo (5/9) and ours (4/9).

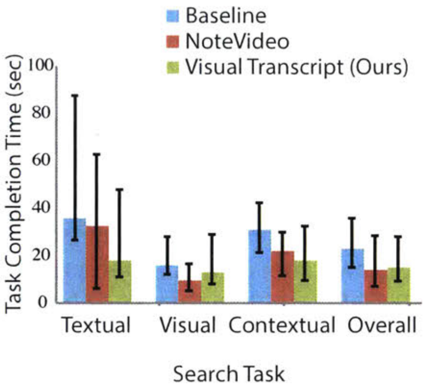


Figure 20: Graph shows median search task completion time, where error bar represents interquartile range.

2.6.2.2 *Users find information involving text faster with VisualTranscript than with NoteVideo or the baseline.*

For the *text search* and the *contextual search* users performed fastest with VisualTranscript followed by NoteVideo and then the baseline (Figure 20), although the differences across the interfaces were not statistically significant according to the non-parametric Kruskal-Wallis Test ($\chi^2 = 0.82$, $p = 0.676$).

For these tasks, users either had to find relevant text or find a visual and also look at the text around it (or listen to the audio). VisualTranscript naturally supports such tasks by interleaving text and figures. NoteVideo does not provide a text to skim through, but users could search for key words or phrases (a feature also provided in VisualTranscript and the baseline). Alternatively, they could click on a visual and listen. Interestingly, the baseline performed worst on these tasks, despite the fact that it is most text-centered and provides the exact same text as VisualTranscript. This is likely because the text in the baseline was unstructured and difficult to read (see next finding).

For the *visual search* users performed fastest with NoteVideo. For all videos we tested, NoteVideo had the advantage of presenting all the visuals in one screen, which made it easier for users to scan the entire visual content without having to scroll.

On average, participants' performance on the search task was comparable on ours and NoteVideo, which was better than the baseline. The difference between ours and the baseline was statistically significant with the Mann-Whitney U (MWU) test ($Z = -2.1$, $p = 0.02$), whereas the difference between ours and NoteVideo was not ($Z = 1.2$, $p = 0.12$). Occasionally, users missed the information in their first search attempt and then tried to scan the entire lecture, contributing to a large variance.

In terms of accuracy, users were most successful in locating the correct information with VisualTranscript (average error rate $e = 0.06$) compared to NoteVideo ($e = 0.07$) or the baseline (0.15), although the differences were not statistically significant (ANOVA, $F_{2,24} = 1.04$, $p = 0.15$).

2.6.2.3 *VisualTranscript makes the transcript text easy to read and skim through.*

Both the baseline and VisualTranscript include the entire transcript text. However, the usefulness of their transcripts is rated very differently. On a 1-7 usefulness scale, VisualTranscript scored 6.3 (range: 5 to 7), whereas baseline scored 4.7 (range: 1 to 7). With the baseline, participants mostly scrubbed through the timeline to complete the tasks. Several users (3/9) mentioned that the baseline transcript text was difficult to skim through or find correspondences with the video. In contrast, with VisualTranscript, users primarily relied on the text and visuals to solve the tasks rather than the video. One user commented that the layout was *"similar to a textbook"* and *"easy to read"*. Another user said that *"the paragraph structure corresponds to the main points"* which facilitates skimming.

2.6.2.4 *Users prefer VisualTranscript for learning.*

The post-task survey showed that, for learning in general, most users (7/9) preferred our interface over NoteVideo (2/9) or the baseline. The reasons for preferring VisualTranscript included *"having the whole script and equations [visuals]"* and *"a good balance between getting the overview and some more detail."* Those who preferred NoteVideo appreciated having all visual content presented *at once* without having to scroll, but also noted that the sequence was not apparent (e.g., many users asked where to click to get to the beginning of the lecture) and that *"there's a risk of missing something that's not written on the board."*

2.7 DISCUSSION

2.7.1 *Limitations*

2.7.1.1 *Generalizability*

VisualTranscript focused on blackboard-style lectures, but the key ideas behind the design of VisualTranscript (e.g., presenting discrete visual entities next to its corresponding narrative in a linear layout) are generalizable to other style of lecture videos. Different styles of lecture videos would require different or more sophisticated visual entity recognition and layout techniques. For example, a classroom recording will have human occlusion, and a slide-based presentation could have animation or other multimedia effects. Our pre-processing step for stroke extraction works especially well with digital blackboard-style lectures with constant background and minimal occlusion. However, videos with more noise (e.g., from lighting change or occlusion by hand) would require a more sophisticated method.

Although in all of our examples the segmentation algorithm outputs produce a comprehensible VisualTranscript, we also observed 2 types of failure cases: (1) *under-segmentation*, where too many strokes are grouped into a single visual entity, and (2) *over-segmentation*, where related strokes are separated into different visual entities. Our scoring function assumes a layout where distinct visual entities are more or less spatially separate from each other. A different method is required to handle videos that violate this assumption, for example a history lecture where most of the writing is superimposed on top of a map, or where figures are overlayed on top of each other (Figure 21). In both cases, our segmentation algorithm output a single large visual entity (under-segmentation).

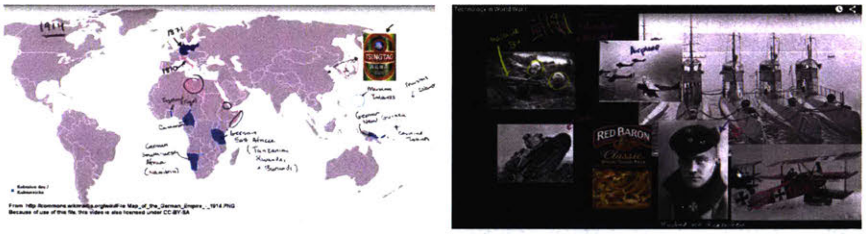
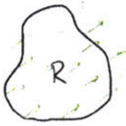

1914
1871
1870
TSINGTAO
German South-west Africa (Namibia)
German East Africa (Tanzania, Rwanda, & Burundi)
German New Guinea
RED BARON
Classic

Figure 21: Our segmentation algorithm assumes that distinct visual entities are more or less separate from each other. For example, our algorithm fails to produce reasonable segments for a history lecture where most of the writing is on top of a map, or where figures are overlayed on top of each other. In both cases, the our algorithm outputs a single large visual entity.

In general, to design the scoring function for the segmentation, we used a heuristic approach based on observations. A machine learning approach based on more data would be useful. An editing or annotation mechanism, including crowdsourcing, to aid segmentation may also be helpful and a potential area for future work.

We also do not handle special cases such as when the instructor erases a part of the board or uses copy-and-paste operations. For instance, if the instructor updates a part of visual content in order to correct a mistake, the current implementation only shows most recent stroke.

In placing temporally aligned visuals and sentences next to each other, we assume that instructors talk about what they are drawing at the same time. This assumption holds in most cases, but fails to resolve other types of references. For example, in Figure22, the pronoun 'here' is used twice, each time referring to a different part of the visual. Whereas in the video these references become clear with the cursor movement, they remain ambiguous in our static output.



Over *here* you'll get positive fluxes.
[...]
But *here*, you'll get a negative flux.

Figure 22: Layout using only temporal correspondence fails to resolve some references. In this example, 'here' in the first sentence refers to the top right portion of the boundary R, whereas the second 'here' refers to the bottom left portion. In the video, these references are clarified by pointing with a cursor.

Our user study shows preliminary results demonstrating the usefulness of VisualTranscript. A more extensive study with more participants across longer periods of time would reveal further insights. For example, in our pilot studies we noticed that users' behaviors varied significantly depending on their familiarity with existing interfaces (e.g., YouTube video player), or their pre-knowledge of the lecture subject.

2.7.2 *Conclusion*

This chapter introduced VisualTranscript, a readable and interactive representation of blackboard-style lecture videos, which interleaves visual content with corresponding text. We use a variant of the classic line-breaking algorithm to segment the visual content of a lecture video into discrete figures. Then, we leverage the temporal correspondence between the figures and transcript sentences to structure the transcript text. Finally, we interleave the figures with corresponding text in an easy-to-read format. User evaluation suggests that compared to a standard video player and a state-of-the-art interface for watching blackboard-style lectures, users prefer our interface for learning. It also suggests that VisualTranscript is effective in helping users browse or search through lecture videos.

3 AUTHORING VOICE RECORDINGS

The first sentence can't be written until the final sentence is written.

— Joyce Carol Oates [94]

Speech recordings are central to modern media such as podcasts, audio books, e-lectures, and narrated documentaries. As with any serious authoring process, producing speech recordings involves an iterative back-and-forth between planning, authoring and editing. That is, producers repeat script writing, script editing, audio recording and audio editing back and forth multiple times. Throughout this process the script and the audio are closely linked to each other.

Yet, most existing tools treat the script and the audio separately, making the iterative workflow between them very tedious. Users have to switch between different applications to work on the script versus the audio, and manually translate edits in one mode to edits in the other mode. This process is especially time-consuming when there are multiple takes involved, or when multiple people are collaborating on the same recording.

This chapter introduces **VoiceScript**, an interface to support a dynamic workflow for script writing, audio recording and audio editing. VoiceScript integrates the script with the audio such that, as the user writes the script or records speech, edits to the script are translated to the audio and vice versa.

We conduct informal user studies to demonstrate that our interface facilitates the audio authoring process in various scenarios.

3.1 INTRODUCTION

Audio recordings of speech are prevalent across a variety of media, including podcasts, audio books, e-lectures and voice-overs for narrated videos. Creating such audio recordings typically involves three main tasks: writing a script, recording the speech, and editing the recorded audio. While authors typically start by writing at least a rough script of what they plan to record, in practice, the process of creating the final audio rarely involves a simple linear progression through these steps. A more common workflow is to move back and forth between writing or editing the script, recording or improvising subsets of the speech, and editing together portions of multiple recorded takes.

For example, consider the case of recording the audio for an online lecture. After writing some notes to use as a rough script, the lecturer records a few takes and listens to the speech. She decides that one of the concepts requires a more detailed explanation, so she edits her notes, re-records the relevant speech, and merges the new recording into the final audio. Such updates may also happen in response to feedback from viewers after the lecture is published online. Similarly, when authoring a voice-over for a video, the initial recording may not align perfectly with the visual footage (e.g., some spoken explanations may be too short or too long for the corresponding video clips). In a collaborative scenario, an editor could request edits to the initial recording of a narrator. In each case, users may need to modify the script and re-record certain sections of the speech. In general, the process of recording and editing the speech together often reveals issues that require going back to edit portions of the script.

Unfortunately, most existing tools for authoring speech recordings do not facilitate this back and forth workflow. Typically, users write and edit the script in a text editing environment and then record and edit the audio in a different audio editing tool. The central issue is that the written script and the recorded audio are treated as completely separate entities. This separation introduces several sources of friction in the workflow. When the user records the speech, any deviations from the initial written text (either intentional or not) are not reflected in the script. Evaluating the recordings to decide what takes to choose or what script modifications are necessary requires careful scrubbing through the audio to find the relevant parts. In addition, once the user chooses a particular version of the speech to include, the script no longer matches the speech, which complicates any subsequent edits. Finally, if the user decides to modify a portion of the script, she must figure out what subset to re-record to ensure that the new recording can be merged in without creating audio artifacts (e.g., replacing a single word in a recorded sentence is hard to do since the word may not blend seamlessly with the adjacent words).

To address these challenges, we design VoiceScript, an interface that supports script writing, speech recording, and audio editing in a unified way. Our key idea is to maintain a so-called **master-script** that is linked to the audio and always reflects the current state of the project, including unrecorded, recorded, improvised and edited portions of the script. We use automatic speech recognition to transcribe the audio into text, and solve the task of combining together multiple recordings and syncing audio with the script like a text differencing and merging problem. To help users maintain a consistent master-script, VoiceScript provides semi-automated tools for merging recorded takes into the master-script and visualizations that indicate what portions of the script need to be recorded or re-recorded in response to edits to the script. The combination of these features enables users to move back and forth between script editing, speech recording and audio editing in a seamless fashion.

VoiceScript can be used to create audio recordings in a variety of workflows, including recording a fairly detailed script, recording without any script, and a collaborative scenario between two users. We conduct informal evaluations where users create their own audio recordings to summarize technical articles. We also compare VoiceScript to a state-of-the-art text-based audio editing tool for the task of creating an audio recording from multiple raw recordings. The results demonstrate that VoiceScript supports a wide range of workflows and enables first-time users to easily author speech recordings. User feedback suggests that the integration of script and audio through the master-script greatly facilitates the authoring process.

3.2 PREVIOUS WORK

3.2.1 *Scripting*

Adobe Story [4], FinalDraft [41] and Celtx [22] are examples of professional software dedicated to script writing. They support collaboration, automatic formatting, navigation and planning for future production, but they treat the script as a text document that is essentially separate from the recordings. In fact, in our formative interviews of lay and professional audio producers, we found that many of them use general-purpose document editors like Google Docs [50] or Microsoft Word [85] to prepare their scripts.

3.2.2 *Recording and Editing Audio*

At the recording and editing stage, many users rely on commercial digital audio workstations, like Adobe Audition [1], Avid ProTools [11], GarageBand [46] and Audacity [10]. Video editing software such as Adobe Premiere [3] or ScreenFlow [110] are also commonly used. These tools allow users to edit audio by manipulating waveforms in a multi-track timeline interface. They also provide a wide variety of low-level signal processing functions. However, since they are designed to serve as general-purpose audio production systems, they include many features that are not directly relevant for creating audio narratives whose main content is speech. Hindenburg Systems [58] develops tools that are specifically targeted for audio narratives.

Still, they are primarily concerned only with the audio and they do not deal with the script directly.

3.2.3 *Text-Based Audio Editing*

Recently, several researchers have explored using audio transcripts to support text-based navigation and editing of audio. Whittaker and Amento [120] demonstrate that users prefer editing voicemail through its transcript instead of its waveform. Inspired by similar intuition, Casares et al. [21] and Berthouzoz et al. [15] enable video navigation and editing through time-aligned transcripts. Rubin et al. [105] extend this approach to audio narratives and propagate edits in the transcript text to the corresponding speech track. These systems all focus on editing pre-recorded audio via its transcript, whereas we also consider how script edits influence the recording process and how audio edits also evolve the script. NarrationCoach developed by Rubin et al. [106] also uses automatic speech recognition to align speech recordings with an input script. However, its focus, improving speech performance at recording time, is different from VoiceScript. Here, we focus on facilitating the back-and-forth workflow between script writing and speech recording. VoiceScript also takes advantage of text-based navigation and editing, but unlike these systems, it supports a dynamic workflow where both the audio recordings and the underlying script can be continuously updated.

3.3 DESIGN PRINCIPLES

To learn about current practices and challenges for creating speech recordings, we interviewed ten professional lecturers and two video producers who regularly create audio recordings for online lectures that are published on online platforms, including YouTube, Udacity, EdX and MITx. The following are several key insights we gained from the interviews.

Scripts are prevalent.

All of the lecturers prepared written materials about what they were going to say before they started recording. The format and level-of-detail of these scripts varied. For instance, one lecturer used his lecture slides containing images and a list of bullet points as his script. Another lecturer typed a thorough word-for-word transcription of what he was going to say in a text document. Another person used handwritten notes as an outline. In all cases, while they were recording, they kept the scripts within their view and depended on them to guide their speech.

Recordings deviate from the script.

In many cases, the initial scripts were rough or incomplete. Only two out of the ten lecturers we interviewed prepared a word-for-word script before recording. The majority used lecture slides or handwritten notes containing a rough outline of what they were going to record. They used these outlines as guides and improvised most of the actual recorded speech. One of the lecturers did an initial recording from the outline, and then used that to flesh out the script before recording additional takes. Even when a word-for-word script was prepared beforehand, the recording often did not follow the script exactly. While recording, the speaker sometimes remembered and added more details, or found a more natural way of saying a written sentence. In some cases, major script changes were made long after the initial recording was created. For example, one lecturer noted that he periodically revisited and re-recorded parts of lectures to add up-to-date examples. The result is that recorded speech almost always differs either slightly or significantly from the initial written script.

While a few people edited the written script to resolve these discrepancies, in most cases the script and recorded audio end up in inconsistent states. This inconsistency makes it difficult for users to update the recording. They cannot simply read and edit the script because it may not accurately represent the recorded audio. Moreover, changing any portion of the recording requires identifying the appropriate subset of speech to re-record such that the new recording can be merged into the final track with no noticeable seams at the take boundaries.

Final track includes multiple recordings.

As mentioned above, users almost always record multiple takes of the speech. Thus, assembling the final track typically requires merging these takes together using audio editing software. Many users noted that aligning the waveforms of multiple takes, finding the best take, and then cutting and joining them seamlessly were very time consuming and tedious tasks.

3.4 THE VOICESCRIPT INTERFACE

Based on these observations, we designed VoiceScript, a speech authoring interface that supports script writing, speech recording and audio editing in a single unified workflow. Our interface is built on three key features.

3.4.0.1 *Text-based representation of audio.*

We build on previous work [15, 21, 105, 120] that demonstrates the benefits of text-based representations of spoken audio for navigation and editing. VoiceScript uses automatic speech recognition to transcribe audio recordings in realtime and represent each take with a verbatim transcript. As with previous systems, text edits to these transcripts are automatically propagated to the audio, which facilitates simple audio editing tasks.

3.4.0.2 *Master-script view.*

To help users manage the relationship between scripted text and recorded speech, we introduce the notion of a master-script that shows a unified view of both unrecorded portions of the script and recorded speech included in the final track. By representing and visualizing both recorded and unrecorded text, the master-script provides a complete, readable view of the current state of the project that evolves as the user records and adds new takes to the final track, edits recorded text, or modifies text that must be recorded.

3.4.0.3 *Merge process.*

Since recorded text typically differs from the script, VoiceScript provides an interface for merging changes into the master-script. The fact that we represent all recorded audio as text allows us to use text differencing algorithms to identify conflicts and execute merges. One key difference between our scenario and standard text merging is that recorded audio cannot simply be cut and merged into the master-script at any arbitrary word boundary. In many cases, the temporal gap between spoken words is not big enough to produce a seamless edit in the final track. Our merge interface takes this into account and helps the user execute merges that are likely to be artifact-free.

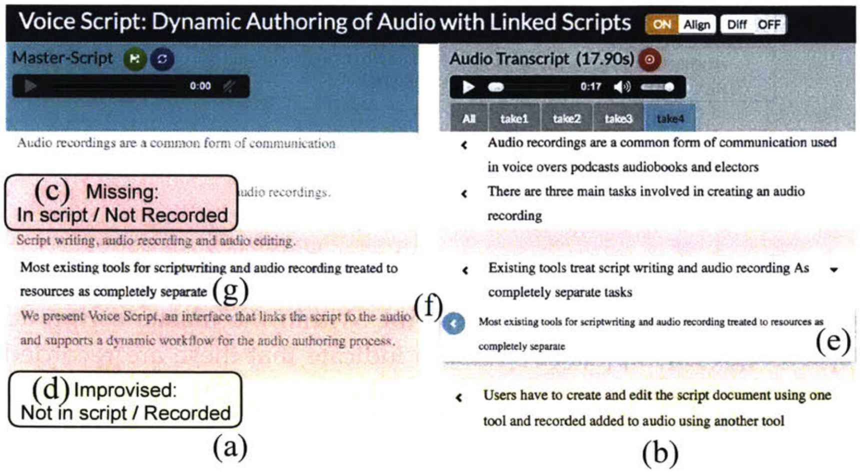
The rest of this section describes our interface through typical usage scenarios of how users might create an audio recording.

3.4.1 *Typical Usage Scenarios*

3.4.1.1 *One-pass authoring.*

Typically, the user begins by writing an outline of points to record in the master-script. The text appears in light grey to indicate that these parts have not been recorded yet (Figure 23 *left*). At this stage, the master-script is like an ordinary, editable text document.

Once the user starts recording, the audio is transcribed in real time and verbatim text corresponding to each take appears in a separate



Voice Script: Dynamic Authoring of Audio with Linked Scripts

Master-Script

Audio recordings are a common form of communication

udio recordings.

Script writing, audio recording and audio editing.

Most existing tools for scriptwriting and audio recording treated to resources as completely separate

We present Voice Script, an interface that links the script to the audio and supports a dynamic workflow for the audio authoring process.

Audio Transcript (17.90s)

Audio recordings are a common form of communication used in voice overs podcasts audiobooks and electors

There are three main tasks involved in creating an audio recording

Existing tools treat script writing and audio recording As completely separate tasks

Most existing tools for scriptwriting and audio recording treated to resources as completely separate

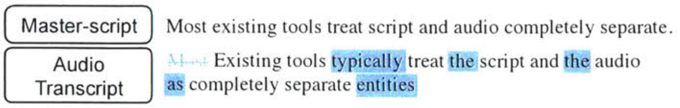
Users have to create and edit the script document using one tool and recorded added to audio using another tool

Figure 23: The VoiceScript interface. (a) Users start by writing an outline in the master-script. Unrecorded text is displayed in grey. (b) When the user records, speech is transcribed in real-time. The transcript is aligned with the master-script. Each take is displayed in individual tabs, and the 'All' tab that shows a summary of all takes. (c,d) Missing / improvised segments are color-coded. (e) Alternative takes of similar sentences are grouped. Users can compare and select. (f) User can accept an audio segment to insert it in the final track. (g) Accepted segments are displayed in dark.

transcript tab (Figure 23 *right*). Each transcript is time-aligned with the corresponding recording, so the user can quickly navigate to specific parts of the audio by clicking on a word in the transcript.

The next task is to cut and merge parts of the recording into the final track. The user needs to compare the recording to the original outline, replace parts of the outline with the corresponding recording, and possibly insert improvised speech. To this end, we provide a **compare-view** that aligns segments of the recording transcript to corresponding segments in the master-script and shows them side-by-side. To indicate improvised portions of the audio, segments of the transcript that do not correspond to any part of the master-script are highlighted in yellow. To indicate missing portions in the audio, segments of the master-script that do not correspond to any part of the transcript are highlighted in red. To view more detailed discrepancies between the script and recording, the user can enable a **diff-view** that displays per-word differences using standard track change markers, i.e., strikethroughs for missing words and highlighting for added words (Figure 24).

To add recorded audio to the final track, the user can *accept* any portion of the recording by clicking a button next to the appropriate transcript segment. If there is a corresponding segment in the master-script, the accepted transcript segment replaces it. If there is no cor-



Most existing tools treat script and audio completely separate.

~~Most~~ Existing tools typically treat the script and the audio as completely separate entities

Figure 24: In the diff-view, users can view detailed, per-word discrepancies between the master-script and audio transcript.

responding master-script segment, the accepted transcript segment is simply inserted into the master-script. Within the master-script, accepted segments appear in black to indicate that these are recorded portions of text that have been added to the final track (vs. grey for unrecorded text).

If there is more than one take, the user has to compare and select between multiple versions of the same segment. In addition to each of the transcript tabs, the **all tab** provides a summary of all of the takes. For each segment in the master script, this tab displays all the corresponding transcript segments from all of the audio takes. A drop-down button next to a transcript segment indicates that there are multiple versions or takes of the segment. Clicking on the button opens a list showing the alternative versions (Figure 23-5). The user can listen to any of these takes and select one without having to search through individual takes.

Finally, the user has to determine which parts of the outline are still missing. When the all tab is in focus, any part of the master-script that has not been recorded in any of the takes is highlighted in red. In this way, the user can tell at a glance what has already been recorded and what still needs to be recorded. All of the dark (i.e. recorded) text in the master-script represents the current state of the final audio track; all of the grey text has not been recorded or is recorded but the author has not yet accepted it into the final track.

3.4.1.2 *Iterative and collaborative authoring.*

The final recording is rarely produced in a single pass. Instead, the user often iterates back and forth between editing the master-script, recording audio takes, and merging audio segments into the final track. It is also common for multiple people to collaborate on a single voice-over. For example, a narrator who records the voice-over may work with others who write and edit the script, or several people may work on a recording with multiple voices.

During any point in the process, users can edit the master-script like a text document. For example, a user can simply insert more text to record or make changes to unrecorded text to flesh out the original outline. These edits can include verbatim script as well as comments or stage directions (e.g., "include examples" or "speak softer"). A user can also edit or delete recorded portions of the text. Deleting recorded text from the master-script will remove the corresponding portion of the audio from the final track. Altering recorded text can introduce audio artifacts (e.g., when a word is deleted mid-sentence), or it could mean that the corresponding text no longer matches the underlying audio. When the user edits a recorded word without completely deleting it, the word is flagged as *dirty* (italicized and marked blue) to remind the user to review or re-record relevant portions. Finally, the user has an option to correct the transcription of recorded words without affecting the underlying audio or flagging it as dirty.

In both iterative and collaborative editing, users need to identify (1) new content that needs to be recorded for the first time, and (2) existing content that needs to be re-recorded after the script edits. To visualize this information, VoiceScript keeps track of per-word metadata about whether a word is unrecorded (grey), recorded and unedited (black), or recorded and edited (blue italics). For collaboration, this metadata is passed between users with the script and recordings. The visualization and the text-based editing and merging interface facilitate audio editing even when different persons work on different parts of editing the script, recording the audio and re-arranging the recorded audio.

3.4.1.3 *Other workflows.*

One key benefit of our interface is that it supports a wide range of workflows for different users and scenarios. For instance, instead of starting with a written outline, the user can begin with an empty master-script, start recording, and then use the initial recording as an outline. The user can also record the entire script in a single take, or work on a single section at a time.

Please visit `https://people.csail.mit.edu/hishin/projects/voice_script/abstract.html` to see a video describing the interface. The voiceover for this video was created by two authors collaborating over VoiceScript. We also look at various workflows in our informal user evaluation in Section 3.6.

3.5 ALGORITHMS

Our authoring interface relies on audio transcription and text alignment algorithms to link the master-script to the audio recordings.

3.5.1 *Transcribing the audio recording*

We use IBM Speech-to-Text Service [63] to obtain a verbatim transcript of each audio recording in real-time. The service outputs a time stamp for each word indicating its start and end time within the audio. It also segments the transcript into *utterances* where each utterance is separated by a longer silent gap in the speech (longer than 500 ms). While automatic speech recognition is imperfect, we have found that in most cases the results were accurate enough for the purpose of alignment (described below) and for users to understand the transcript.

3.5.2 *Aligning the transcript to the master-script*

To support our side-by-side compare-view as well as the all tab view, we must identify corresponding parts of the master-script and recording transcripts. Moreover, we must partition these corresponding parts into segments that users can easily compare and merge into the master-script. Ideally, our segments should respect natural boundaries such as punctuation and line breaks in written text to aid readability. As discussed earlier, the segment boundaries should also align with longer pauses in the audio so that merge operations do not introduce obvious audio artifacts. Finally, we also want to separate parts of the transcript that generally agree with the master-script (i.e., planned speech) from parts that do not (i.e., improvised speech). We designed a scoring function that optimizes for these requirements and use an iterative algorithm to co-segment the two texts. We first explain the algorithm and then describe the scoring function in detail.

3.5.2.1 *Iterative co-segmentation*

Before running our co-segmentation algorithm, we first compute the global word-to-word alignment between each recording transcript and the master-script using the Needleman-Wunsch (NW) algorithm [91]. NW allows for insertions and deletions, which account for differences in the two texts, for example, due to rough scripts, inaccurate speech, or transcription errors.

The segmentation of the master-script depends on the segmentation of the transcript and vice versa. Our iterative algorithm alternates between optimally segmenting the master-script and the transcript independently using the result from one to segment the other. We initialize the segment boundaries at punctuation marks (.!?:;) in the unrecorded text and silent gaps (> 500ms) in the recorded text. In practice, we found that two iterations were sufficient to converge to a solution.

For each optimization step, we use the classic optimal line-breaking algorithm by Knuth and Plass [70]: The algorithm takes an input text T and a reference text R, and outputs an optimal segmentation for T.

First, let $e(T_{j,i})$ (defined in the following section) denote the score of a segment containing words j through i. We want to find a segmentation, S, which maximizes the total score over all segments in S.

Consider an optimal segmentation, S_i^* of words $1, ..., i$. For some $k <= i$, words $k, ..., i$ are in the last segment, and the remaining words $1, ..., k-1$ are in the previous segments. Let S_{k-1} denote this segmentation of words $1, ..., k-1$. Then, S_{k-1} must be the optimal segmentation of words $1, ..., k-1$. Otherwise, we could exchange S_{k-1} for the optimal segmentation and increase the total score of S_i^*, contradicting the fact that S_i^* is optimal. (This assumes that there is no interaction between the cost of the previous segments and the cost of the last segment, which is the case for our cost function defined below.) We can express the maximum score E of a segmentation as a recurrence relation:

$$E(S_i) = \begin{cases} 0, \text{ if } i = 0 \\ \max_{0 \leqslant k \leqslant i} E(S_{k-1}) + e(T_{k,i}) \end{cases} \tag{12}$$

We solve for the optimal segmentation using dynamic programming.

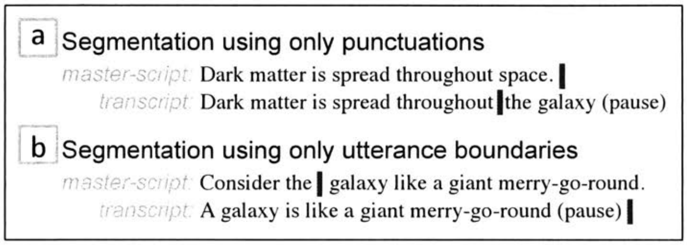
3.5.2.2 *Scoring function*

The algorithm described above requires a scoring function (E) that evaluates the goodness of a candidate segmentation. E is a sum of the scores, e, for individual segments in the candidate segmentation. We define the scoring function based on three terms:

1. *Punctuation and silent gaps*: We prefer segment boundaries after sentence punctuation marks, and in case of recorded text, where there is a longer silent gap. Placing cuts at silent gaps allows audio segments from different takes or different parts of a single take to be joined seamlessly. More precisely, we define the boundary score, e_b, for a single text segment $T_{j,i} = \{w_j \ldots, w_i\}$ as:

$$e_b(T_{j,i}) = \begin{cases} 1.0, \text{ if } w_i \text{ is unrecorded \& ends with punctuation (.!?:;)} \\ -1.0 \text{ if } w_i \text{ is unrecorded \& ends w/o punctuation} \\ t_{gap}(w_i) \text{ if } w_i \text{ is recorded} \end{cases} \tag{13}$$

where $t_{gap}(w)$ is the silence gap in seconds after a recorded word, w, and is equal to 1.0 for w that is at the end of a recording. It is important to consider both the sentence punctuation and the silent gaps. As the examples in Figure 25 illustrate, considering only punctuation can result in audio artifacts when merging recordings. Similarly, considering only utterance boundaries can produce unnatural cuts in the middle of a scripted sentence.



a Segmentation using only punctuations

master-script: Dark matter is spread throughout space. |

transcript: Dark matter is spread throughout | the galaxy (pause)

b Segmentation using only utterance boundaries

master-script: Consider the | galaxy like a giant merry-go-round.

transcript: A galaxy is like a giant merry-go-round (pause) |

Figure 25: Co-segmentation of master-script and transcript texts. (a) Segmentation using only punctuation marks results in abrupt cuts in the audio. (b) Segmentation using only utterance boundaries produces unnatural cuts in mid-sentence. Our scoring function takes into account both sentence punctuation marks and audio pauses.

2. *Global alignment*: We try to separate transcript segments that have a counterpart in the master-script (planned) from those that do not (improvised). Likewise, for the master-script, we want to separate segments that have a match in the transcript (recorded) from those that do not (unrecorded). We utilize the global alignment output from the Needleman-Wunsch (NW) algorithm. For each word w_i in the input text, T, NW outputs a mapping to the reference text, R, and vice versa. For instance, m_i is the index of the word in R that matches w_i. $m_i < 0$ if the word has no match. We prefer text segments that have the proportion of matching words close to 0 or 1. The alignment score, e_a, for a single text segment $T_{j,i}$ is:

$$e_a(T_{j,i}) = 2 \times \left| \sum_{n=j}^{i} \mathrm{match}(w_n)/(i-j+1) - 1/2 \right| \tag{14}$$

where $\mathrm{match}(w_i)$ is 1 or 0, depending on whether $m_i \geqslant 0$ or not (whether the word has a match or not).

3. *Consistency with the other text*: Since the end goal is to align the segments from both texts, we would like the segment boundaries from the input text to align with the segment boundaries in the reference text even when the punctuation and utterance boundaries do not coincide. Let $S' = \{s'_1, \ldots, s'_n\}$ be the segmentation of text R. Given this segmentation and the mapping of T

to R from NW, the consistency score, e_c, for a text segment $T_{j,i}$ is:

$$e_c(T_{j,i}) = \begin{cases} 1.0, \text{ if } s'_{m_i} \neq s'_{m_k} \text{ for the smallest } k > i, m_k \geqslant 0 \\ -1.0 \text{ otherwise} \end{cases} \tag{15}$$

s'_{m_i} is the segment index of the word in R that matches w_i (or in case, w_i does not have match, we find the closest previous match to w_i). Likewise, s'_{m_k} is the index of the word in R that matches w_k (the closest next match to w_i). In other words, w'_{m_k} is the closest word after w'_{m_i} that has a match. The segmentation score is higher if these two words belong to a separate segment of R. (Figure 26)

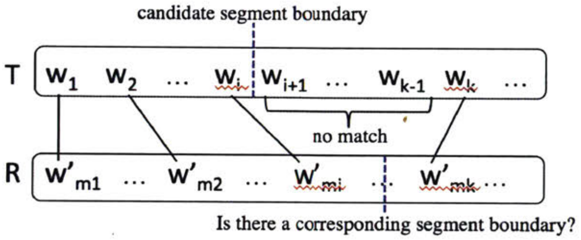

candidate segment boundary
T
W1 W2 ... Wi Wi+1 ... Wk-1 Wk ...
no match
R
W'm1 ... W'm2 ... W'mi ... W'mk ...
Is there a corresponding segment boundary?

Figure 26: We prefer consistent segmentation between the two texts. w_i is a good segment boundary for T if it has a corresponding segment boundary in the reference text R.

We combine these terms into a single scoring function e as follows.

$$e(T_{j,i}) = e_a(T_{j,i}) + e_b(T_{j,i}) + 0.5e_c(T_{j,i}) - 1 \tag{16}$$

where -1 is a normalization term to prevent single-word segments.

The total score for a set of text segments S is:

$$E(S) = \sum_{T_{j,i} \in S} e(T_{j,i}) \tag{17}$$

For notational convenience, we use $T_{j,i} \in S$ to refer to the set of contiguous words in T that are assigned to the same segment in S.

We iteratively segment the master-script and the transcript texts. In practice, we found that 2 iterations was sufficient to converge to a final co-segmentation of the two texts.

3.5.2.3 *Alignment.*

Given a co-segmentation of the master-script and the transcript text, we then compute the best matching master-script segment for each transcript segment. The match score between two text segments T_1 and T_2 is defined as the proportion of words in those segments that have a match between each other (from the NW output). Since NW outputs a linear match between the words in the two texts, the alignment between the segments also respects a linear order. The result is an alignment between the master-script and transcript segments.

We use this alignment to facilitate syncing and merging of script and audio by presenting tools similar to common text differencing and merging tools. The compare-view displays matching segments side-by-side. Our color-coded visualization indicates which portions of the master-script is missing from the transcript, and which parts of the transcript are improvised (i.e., parts that do not have matching segments). In the all tab, segments from separate transcripts that match similar portions of the master-script are grouped together so that users can quickly compare and select between one of them. Similar to text or code merging, users can select a transcript segment to overwrite the matching master-script segment.

3.6 USER EVALUATION

3.6.1 *Informal User Study*

To assess the overall usability of VoiceScript and to observe how users leverage various features of the interface, we conducted an informal evaluation with 4 users (U1-4). 4 graduate students were recruited for the study. All of them had little or no previous experience with speech recording. We started each session with a 10-minute demonstration of our interface. Then, we gave users a short article about a technical subject: *What is a Decibel?* from howstuffworks.com [60] or *How Lasers Work* from David Macaulay's illustrated book, *The Way Things Work* [83]. The users' task was to create an explanatory audio recording about the subject using our interface. Users were allowed to refer to the article during the authoring process or to take notes on the master-script, but they were discouraged from recording the article by reading it out loud. We examined the users' workflow and solicited written qualitative feedback about the authoring experience at the end of the session. Each session lasted about 40 minutes.

While the size of our user evaluation is small, the initial findings are extremely encouraging. All users successfully produced a complete audio recording summarizing the article.

3.6.1.1 *VoiceScript supports various workflows.*

Interestingly, each user adapted a very different workflow. For example, U1 started by writing a complete list of main points. For each take, U1 recorded a few points from the list, merged them into the master-script, and then continued to record the next point on a separate take. In contrast, U2 wrote part of the script, recorded that portion, and moved on to write the script of the next part. U3 did not write an initial script, but improvised the recording and used that as a starting point to edit and re-record afterwards. Similar to U1, U4 started by writing a rough outline. But, instead of recording a few points, U4 recorded the full script at each take and merged the best parts to get the final track.

Sometimes users typed verbatim script to read aloud during the recording, and other times they wrote rough outlines. For example, U2 noted, *"For the introduction, I had a pretty good idea of what I wanted to say, so it saved me time to use only bullet points. [For the second part] I wrote full sentences, as I was not familiar with all the technical details and it would have been more difficult to improvise. I enjoyed being able to use the master-script in both ways."*

The differences in the workflows could be due to personal preference, and/or to the article content. In any case, our interface was able support various workflows.

3.6.1.2 *The master-script facilitates iterative workflows.*

As the above examples also demonstrate, users took advantage of the master-script to go back and forth between scripting and audio recording. For instance, U2 initially wrote a very rough outline for the script. After recording and merging the first take based on this rough script, U2 refined the master-script, and then recorded more takes. Similarly, after recording and merging audio takes into the final track, U1 noticed a mistake in the speech (i.e., instead of saying *140 decibels*, U1 had said *40* decibels). U1 corrected the corresponding recorded text in the master-script, re-recorded the relevant portion part by reading out the edited master-script, and replaced it. During the back-and-forth iteration, users took advantage of our color-coded visualization that indicated sections of the master-script that required recording (grey) or re-recording (blue italics).

3.6.1.3 *Users found the master-script to be helpful.*

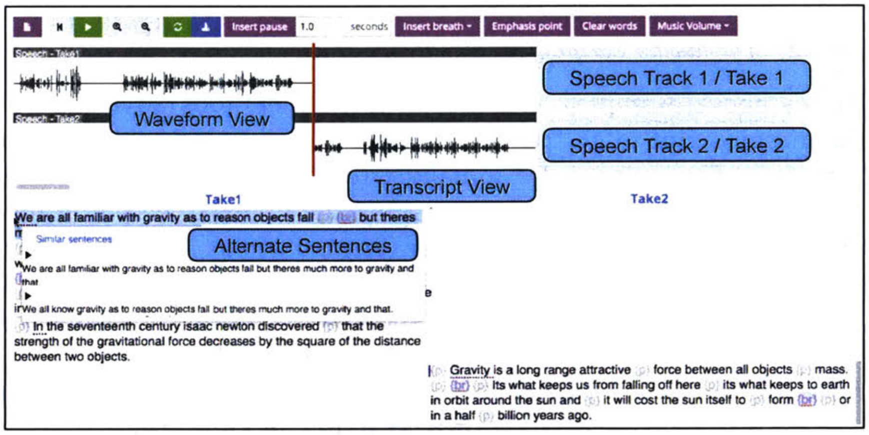
All users offered strong positive feedback about our authoring interface, and said they would use it to create speech recordings. They were most enthusiastic about the integration of the script and the recordings in the master-script document, and the ability to align the master-script to the transcripts. To quote from one user, *"Writing the script on the same interface and having that integrated with the audio was most helpful."* Another user noted that the *"compare-view helped to keep* *track of what pieces of information were already recorded and which ones were still needed."*

3.6.1.4 *Users were satisfied with the quality of the final recording.*

Participants were satisfied with the overall quality of the final recording. One user wrote, *"I was surprised how the final recording from the multiple takes was seamless."* Users noted that while speech recognition was imperfect, *"the transcriptions were accurate enough to understand and easy to check [by clicking to listen to the corresponding audio]."*

3.6.2 *Comparative study*

One of the main tasks in creating audio recordings is cutting and merging multiple audio takes. Many existing applications specifically assist this task (see Section 3.2 Previous Work). We separated out this audio editing task, and conducted a small comparative study to explore whether our master-script view facilitates the task compared to a state-of-the-art transcript-based speech editing interface [105].



We are all familiar with gravity as to reason objects fall {p} {br} but theres

We are all familiar with gravity as to reason objects fall but theres much more to gravity and that

We all know gravity as to reason objects fall but theres much more to gravity and that.

{p} In the seventeenth century isaac newton discovered {p} that the strength of the gravitational force decreases by the square of the distance between two objects.

{p} Gravity is a long range attractive {p} force between all objects {p} mass. {p} {br} {p} Its what keeps us from falling off here {p} its what keeps to earth in orbit around the sun and {p} it will cost the sun itself to {p} form {br} {p} or in a half {p} billion years ago.

Figure 27: Rubin et al.'s text-based audio editing tool (*Interface-R*). For the purpose of our comparative study, each audio take was loaded as a separate speech track.

Similar to VoiceScript, Rubin et al.'s interface (shown in Figure 27, and referred to as **Interface-R** hereafter) also uses time-aligned transcripts to support text-based editing. In both systems, users can edit the transcript like a text document using operations such as copy-and-paste, insert, or delete, and the edits are propagated to the audio. Both systems also detect alternate takes of the same sentence and group them together so that users can easily compare and select between them. However, unlike VoiceScript, Interface-R does not have a master-script that integrates multiple audio recordings with a script. In fact, Interface-R does not explicitly handle multiple recordings. To simulate multiple recorded takes, we took advantage of their multiple speech tracks, so that each take appeared in a separate speech track (i.e., separate columns).

We recruited 2 undergraduate and 2 graduate students, none of whom had prior experience using text-based audio editing systems. We gave them a script with bullet points outlining a mini lecture on a technical subject (*Gravity* and *Dark matter*) and two audio takes roughly corresponding to the script. In VoiceScript, the script was contained in the initial master-script. Since Interface-R does not have a notion of a script separate from the transcripts, we gave users a hard copy of the script. The task was to cut and merge the two prerecorded takes to produce a recording that contained all the contents listed in the script and only those contents. The two takes were similar, but each take had some missing content and some extra content. The participants had to choose parts from each take and combine them to get the final result. We encouraged the users to focus on having the complete content rather than on the details of the audio quality (e.g., tempo, diction, flow of speech).

Each participant completed the task twice, once on each interface using different scripts. The subject of the lecture and the order of the interface were counter-balanced. We examined the time users spent to complete the task, the number and type of functions they used, and the quality of the final recording. After each task, participants gave written qualitative feedback about their experience. In total, each session lasted about 1 hour.

3.6.2.1 *Users completed the task faster using Voice Script.*

All participants completed the task 25% faster using our interface (average 7.4 vs 9.9 min), and also preferred it to Interface-R. The difference may be explained by the different workflow that each interface affords. In Interface-R, users effectively started with both recordings in the final track. They applied copy-and-paste to cut and merge the two takes, and deletion to remove redundant or superfluous content. In contrast, in VoiceScript, users started with an empty final track. Then, using the compare-view, they accepted parts that matched the script from either of the takes. Although copy-and-paste and deletion were also available in VoiceScript these operations were used only rarely, for example to delete a mistakenly accepted segment, to delete individual words, or to change the ordering of accepted segments (Table 1).

Audio Editing Session

User	Time spent	Total	Accept	View	Text
	ours *(Rubin)*	cuts	*ind / all*	alternate	edit
A	5:58 *(08:10)*	3	4 / 3	-	2
B	6:40 *(11:10)*	3	3 / 4	-	1
C	9:37 *(11:05)*	2	7 / -	1	7
D	7:25 *(09:10)*	6	10 / 1	3	-

Table 2: Four participants created audio recordings from two pre-recorded takes, using our interface and Rubin et al.'s interface. Usage statistics pertain to our interface. Accept *ind / all* are number of segments accepted from the individual transcript view and the *all* tab view respectively.

3.6.2.2 *The master-script facilitates merging.*

Users appreciated having the master-script view. First, the master-script served to integrate the script outline and the recordings into a single comprehensive document. To quote from one user, *"The master view integrated the two takes into an almost seamless whole that I just had to edit, as opposed to presenting two separate drafts from which I had to generate a third and final story."* Secondly, the master-script helped users keep track of the status of the final track compared to the planned script. One user noted that *"The outline [in the master-script] made it easier to find what I had accepted to the final track and what was still missing, instead of making notes on the paper outline and going back-and-forth between it and the recordings."*

3.6.2.3 *The compare-view facilitates merging.*

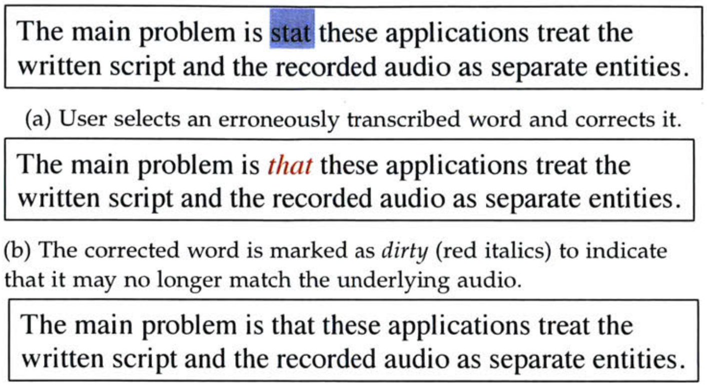
All 4 participants mentioned the align function in the compare-view as the most helpful feature in VoiceScript. First, as one user noted, the segmentation in *"the alignment view made editing much faster by visually breaking the script and the transcript into corresponding parts. I found I had to read much less."* Also users found it *"easier to click than to copy and paste"* in order to merge portions of recordings into the final track.

3.7 DISCUSSION

3.7.1 *Limitations*

We rely on automatic speech recognition (ASR) to transcribe the audio recordings in real time. Despite recent improvements in speech recognition [59], its performance varies widely. For example, ASR accuracy can fall significantly for speakers with strong accent. Such transcription errors can affect the user's performance negatively (e.g., in navigating the audio, or if the user has to spend time correcting the errors) [47].

In VoiceScript, users can click on a word to listen to its corresponding audio and manually correct the transcription without affecting the audio. (Figure 28)



The main problem is stat these applications treat the written script and the recorded audio as separate entities.

(a) User selects an erroneously transcribed word and corrects it.

The main problem is *that* these applications treat the written script and the recorded audio as separate entities.

(b) The corrected word is marked as *dirty* (red italics) to indicate that it may no longer match the underlying audio.

The main problem is that these applications treat the written script and the recorded audio as separate entities.

(c) User can mark the word as *clean*

Figure 28: In VoiceScript, users can manually correct the transcript without affecting the underlying audio.

An efficient interface to handle transcript edits is an interesting area for future work. For example, our current interface does not differentiate between transcription error correction versus content editing. When a user edits a transcribed word it is marked as *dirty* to indicate that it may no longer match the underlying audio and that it may need re-recording. If the user intended to simply correct a transcription error (without changing the audio), she can manually mark it as *clean*. Otherwise, she can re-record and replace the *dirty* portion of the audio. We also do not handle correcting multiple words at the same time or retiming the audio according to the corrected transcript. Taking advantage of the written script to fix or reduce speech recognition errors is another interesting area for future work.

Our segmentation and alignment algorithm has several limitations. First, we compute a linear alignment between the script and the transcript. We do not handle cases when the speaker repeats a segment within a single take or mixes the order of the segments.
We used a heuristic approach to design the scoring function, and hand-tuned its parameters. In practice, our algorithm output reasonable segmentation across different scenarios. However, more experiment is necessary to determine the importance of different factors in the scoring function.

VoiceScript supports asynchronous collaboration to create voice recordings. Additional features are required in order to support synchronous collaboration, in particular, conflict resolution and version control.

Our interface focuses on the content of the speech recordings but does not consider editing details for audio quality. Specifically, different takes may have a different sound quality, or transition between different takes may sound artificial. As one user mentioned in the feedback, we could integrate editing tools such as the ones in Rubin et al. [105] or [106] to fine-tune the audio quality to produce a smoother final recording.

Speech recordings often accompany visual footage or other sound effects such as music that is also closely related to the script/audio. Future work could investigate how to integrate these contents in the workflow.

3.7.2 *Conclusion*

To create speech recordings, people iterate back and forth between script writing or editing and audio recording or editing. It is also common for several people to collaborate in the authoring workflow. Unfortunately, most existing tools treat the script and the audio as completely separate entities, which makes the dynamic workflow between them very tedious. We presented VoiceScript, an authoring interface that facilitates iterative workflows for script writing and audio recording or editing. We used our system to collaborate asynchronously to create a voice-over. Through an informal user study, we demonstrated that our interface supports a wide range of workflows, and that our master-script seamlessly integrates scripting and recording. A comparative study showed that our system facilitates novice users editing speech recordings.

4 DELIVERING SLIDE PRESENTATIONS

Presentation of ideas is conversation carried on at high voltage – at once dangerous and more powerful.

— Henry Boettinger [16]

Presentations are everywhere. We give and sit in on presentations in classrooms, at business meetings, and in conferences. Not surprisingly, presentation technology plays an important role in how we communicate and learn.

Electronic slides have become especially popular with standard software PowerPoint and Keynote. While slides allow attractive presentation of visual information, they do not afford some of the basic flexibility that traditional tools such as blackboards provide for pacing, handwriting, content or layout adjustment.

In this chapter, we introduce **Aparecium**, a presentation interface that helps presenters deliver flexible and engaging presentations by combining the aesthetics and organization of electronic slides with the spontaneity of inking. In Aparecium, presenters use inking interactions to deliver slide presentations. With inking, presenters can (1) reveal pre-authored content to the audience, (2) make annotations on top of the slide, and (3) adjust the slide layout to create blank space. Pre-authored slides help improve the visual aesthetics and organization of the slides, while inking enables presenters to have flexibility and fine-grained control over the content and pace of the presentation.

Aparecium: A charm for revealing invisible ink in Harry Potter [104]

In a user study comparing Aparecium with baseline presentation tools, we found that our interface generally improves presentation quality without increasing the burden on the presenter at authoring or presentation time. Especially for text-centered or process-driven content, both audiences and presenters preferred presentations delivered using Aparecium.

4.1 INTRODUCTION

Presentations are an important component of both classroom and online instruction. They allow presenters to communicate concepts by combining visual content with spoken explanations. As a result, tools for authoring and delivering presentations have a significant influence on how people teach and learn. Today, the two dominant types of presentation technology are slides and blackboards.

Slides allow presenters to refine the appearance and organization of material in advance. As such, they are most convenient for information-rich content like images or detailed diagrams and charts. However, pre-authored slides restrict how content can be revealed during the presentation. Rather than displaying all of the slide content at once, which can make it difficult for the audience to know where to focus, presenters often set up animation effects to reveal visual elements incrementally. Since the sequence and granularity of these animations are determined ahead of time, it is difficult to add new content or change the order of reveals during the presentation, for instance, in response to the audience. Animations also display information quickly and hasten the pace of the presentation. Indeed, slide lectures tend to show more information in a shorter period of time than blackboard lectures [72], potentially making it more difficult for the audience to follow.

As an alternative, some presenters prefer to write on physical blackboards or ink on virtual displays in real-time. This presentation style is direct and intuitive, and in contrast to pre-authored slides gives presenters full control over how content is displayed. On the other hand, writing and talking at the same time is cognitively demanding. As a result, bad handwriting and poor layouts (e.g., running out of room while writing) are common in blackboard-style presentations. It is also difficult to incorporate complex visuals since they must be created from scratch during the presentation. This often results in long pauses or rambling, repetitive explanations as presenters focus on drawing or writing.

Some presenters blend the two modes of presentation, for example, by projecting slides onto a board and inking on top of them. Recently, PowerPoint and Keynote, also allow presenters to ink over electronic slides in presentation mode. However, in these approaches, the ink and underlying slides are treated as completely separate layers of content that retain their individual drawbacks. The slide content remains fixed and inflexible while real-time inking still requires the user to draw carefully albeit with the help of the slide content as a reference.

To address these limitations, we propose Aparecium, a presentation interface that combines the advantages of slides and inking. In Aparecium, presenters use pre-authored slides to prepare the presentation content. However, instead of specifying animations beforehand, presenters use inking interactions to flexibly display the content on demand. By inking, presenters can control the pace at which pre-authored material is revealed, as if writing in real-time. At the same time, they do not have to worry about writing neatly or carefully arranging all the visual elements on the screen. Our system also allows improvised annotations and enables presenters to create blank space inside the slide to insert new content during the presentation. Aparecium supports all of these functions as modeless interactions.

We evaluate our interface from the perspective of presenters and the audience and compare Aparecium against two baselines, representing conventional slide and inking tools. Presenters found Aparecium easy to use for preparing and delivering presentations. They were also generally pleased with the quality of the presentation produced using our interface. Overall preferences between presentation interfaces depended on the type of content. Both presenters and audience members clearly preferred Aparecium for text-heavy, process-driven material (e.g., explaining algorithms or mathematical derivations), and rated our system as comparable to the baseline slide interface for presenting complex, multi-part diagrams.

To summarize, this chapter presents the following contributions:

- The design and architecture of a new presentation interface, Aparecium, that combines inking with pre-authored slides.
- A set of modeless inking interactions that analyze the underlying slide content to help presenters reveal, annotate, and create extra space during a presentation.
- An evaluation from the perspective of presenters and the audience that compares Aparecium against two baseline interfaces.

4.2 PREVIOUS WORK

4.2.1 *Presentation Software.*

The vast majority of presentations today are created with WYSIWYG slide authoring software like PowerPoint [100], Keynote [66] and Google Slides [52]. While these tools provide a broad range of content creation features, including the ability to add animation effects to slide elements, they offer limited flexibility or control at presentation time. Presenters can only advance linearly through the predefined sequence of animations and slide transitions.

4.2.2 *Nonlinear Presentations.*

To address this shortcoming, research investigated how to support nonlinear paths though a presentation. Moscovich et al [87] organize slides into nested directed graphs and allow presenters to choose between multiple paths on the fly. Similarly, Drucker et al. [38] suggest a method to compare and manage multiple slide presentation paths. Fly [77] and CounterPoint [49] allow spatial navigation by embedding slides on an infinite canvas and employing zooming user interfaces (ZUIs). Prezi [102] is a commercial, online platform for authoring zoomable presentations. Whereas these work focuses on navigating between slides or the presentation as a whole, the interactions presented in this chapter provide flexibility and control within each slide.

4.2.3 *Controlling Presentations.*

Other work explores alternative techniques to control slide presentations. Palette [92] uses physical cards to provide random access to slides, Baudel and Beaudouin-Lafon [14] propose hand gestures, and Cheng and Pulo [24] use an infrared laser pointer to control presentations. Cao et al. [20] perform a systematic user study comparing different interaction techniques, including hand gestures, laser pointer and standard mouse/keyboard input. We suggest inking interactions as the main mode to present slides.

4.2.4 *Inking on Digital Documents.*

Many systems [53, 84, 128] support digital inking to make annotations on top of documents. Perhaps most similar in spirit to Aparecium are systems that integrate digital ink with electronic slides. Anderson et al. [6] propose Classroom Presenter, a distributed presentation system that allows instructors and students to share digital ink on top of electronic slides. Recently, PowerPoint and Keynote added support for presenters to ink in presentation mode as well. SMART is another commercial system that supports inking and projected material using an interactive whiteboard [108]. While all of these systems combine slides with inking, the underlying slide content remains inherently separate from the ink on top. In contrast, in our system, presenters use ink to reveal underlying slide elements in a flexible, fine-grained way at presentation time.

4.2.5 *Beautifying Ink.*

To further assist freeform digital inking, researchers have experimented with different methods to beautify the user's ink strokes. Beautification is applied to meet the requirements of specific scenarios, such as geometric diagrams [42, 61, 64], hand-drawn pictures [78, 126], handwriting [130] or mathematical diagrams [71]. Aparecium does not modify the user's ink stroke per se, but it achieves a similar effect by making the ink stroke disappear gradually and revealing the underlying pre-authored slide content instead.

4.3 DESIGN PRINCIPLES

4.3.1 *Current Practices and Needs*

To learn about current practice and unsupported needs in presentation technology, we conducted in-depth interviews with 5 university lecturers, 8 graduate student TAs, 5 undergraduate students, and 5 online lecturers, from multiple institutions and with varied experiences in giving and listening to presentations. The interviews were semi-structured and covered 1) the methods of presentations they used, 2) in what settings they prefer slides versus inking, 3) challenges of currently available technologies and what they would like to see in future presentation systems. We also consulted existing literature comparing different types of presentation software. From this analysis, we summarize the key findings that informed the design of our system.

4.3.1.1 *Flexibility in presentations is preferred for interactive or informal settings.*

For settings such as research conferences or business meetings with tight time constraints and little room for audience interaction, people prefer to give highly scripted presentations with electronic slides. However, for settings such as lectures, tutoring sessions, or brainstorming meetings, presenters like to have some flexibility and often use inking as part of their presentations. Common strategies include using a blackboard or projecting slides/transparencies onto a board and inking on top of them. Several lecturers purposefully leave blank spaces on their slides to fill in by inking during the lecture.

4.3.1.2 *Presenters want flexibility over prepared contents rather than complete improvisation.*

Even for informal settings, presenters have the bulk of the content planned and prepared beforehand, in the form of lecture notes, worksheets or slides. Thus, the type of flexibility that presenters want is the ability to make small-scale adjustments on-the-fly, such as omitting part of the content, adding minor changes such as a line of text or annotations, or changing the order of the contents. Inking is often used to this effect.

4.3.1.3 *Pacing is important and context dependent.*

The choice of tool also affects the pace of the presentation. Electronic slides are useful for displaying information quickly, which may explain why presenters prefer them for time-constrained settings. Inking takes time, but it allows presenters to have fine-grained control over the pace of the presentation. Depending on the subject matter, the pace of real time writing also makes it easier to follow for the audience. For example, when describing sequential processes like solving a math problem or explaining a complex diagram, both presenters and viewers find it more effective to write them out step-by-step in real-time. Slide animations can simulate this effect, but setting up fine-grained animations is tedious. As a result, animated presentations typically include a very coarse set of discrete steps.

4.3.1.4 *Visual aesthetics matter but are difficult to achieve with inking.*

Both as a presenter and as an audience, people frequently mentioned better visual aesthetics as an advantage of slides over inking. Presenters are often not satisfied with or even embarrassed by their own handwriting. They pointed out that it is even more difficult to write while talking at the same time. Even small operations, such as changing the pen color, seem burdensome during the lecture, as noted by Anderson[5]. From a viewing perspective, people like the legibility and organization that pre-authored slides provide. As [44] also mention, sometimes audiences even felt that lecturers are better organized when they present using electronic slides.

4.3.2 *Design Goals*

The above findings highlight the complementary attributes of electronic slides and inking. While slides are typically more organized and aesthetically pleasing, inking offers greater flexibility and fine-grained control at presentation time. Our aim is to develop a presentation interface that combines the advantages of both existing technologies without increasing the burden on the presenter at authoring and presentation time. More specifically, our system should achieve the following design goals. The first two goals are concerned with improving the presentation quality, while the last goal involves reducing the presenters' effort.

4.3.2.1 *Maximize organization and aesthetics through pre-authored contents.*

In order to improve presentation quality, we want to take full advantage of contents that presenters prepare beforehand. Pre-authored contents can help achieve visual aesthetics. It also *forces* the presenter to organize the presentation ahead of time.

4.3.2.2 *Maximize flexibility and fine-grained control during presentation delivery.*

Presenters should have fine control over what visual content to present, and also when, how much, and how fast to present them. Moreover, these decisions need not be made ahead of time; instead, presenters should be able to implement and adjust them while delivering, according to the content and audience.

4.3.2.3 *Minimize presenter effort during delivery as well as during preparation.*

We want to give presenters more control, but without increasing their burden. Interactions during delivery should be as simple and natural as possible. Similarly, preparation itself should not take more effort than, for example, authoring regular slides. Moreover, presenters should be allowed to focus more on preparing the content itself rather than, for instance, spending time to setup animations effects for delivery.

4.4 THE APARECIUM INTERFACE

Based on these design goals, we developed Aparecium, a presentation system that combines electronic slides with inking interactions. With Aparecium, presenters pre-author slides to organize and refine the visual aesthetics of their material. Unlike traditional electronic slides, presenters do not specify beforehand when, how, or in which order the visual elements in the slide will appear via scripted animation effects. Instead, they simply specify which elements will be displayed to the audience immediately (background) versus which elements will be hidden and then revealed in real time (foreground).

The key innovation of Aparecium is in how presenters deliver the pre-authored content. The main mode of interaction at presentation time is inking, but instead of just adding strokes to the slide, inking supports three functions depending on the context: (1) If the presenter inks over hidden elements, it reveals the pre-authored element to the audience. (2) If the presenter inks over empty space or over revealed elements, ink strokes are added on top of the slide. (3) Finally, if the presenter holds down the pen after drawing a stroke, the user can adjust the slide layout to create blank space.

Inking allows presenters to have flexibility and fine-grained control over when, how much, and how fast to reveal elements on the slide. At the same time, the ability to reveal pre-authored content allows presenters to not worry about writing or drawing neatly. Presenters can also add extra writing or annotations on top of pre-authored elements, and create blank space if necessary. All of these interactions are implemented as modeless pen interactions.

Here we elaborate on slide authoring and inking in our system.

4.4.1 *Slide authoring*

Slides in Aparecium can be authored using any existing slide presentation software (e.g., PowerPoint, Keynote, GoogleSlides). They can include typed text or images, as well as, hand drawn ink strokes. Instead of specifying animation effects on these slide elements, presenters separate them into two layers for each slide by annotating the foreground elements. The background layer is always visible and it is what the audience sees initially. The foreground layer is initially only visible on the presenter view; it is faded to differentiate from the background layer. Presenters can reveal parts of the foreground to the audience during delivery, at which point it becomes unfaded. Presenters also have the option of preparing a third layer, the notes layer, which is only visible on the presenter view and acts as transparent speaker notes placed on top of the slides. Layers in Aparecium are represented as bitmap images. (Figure 29)

4.4.2 *Inking during delivery*

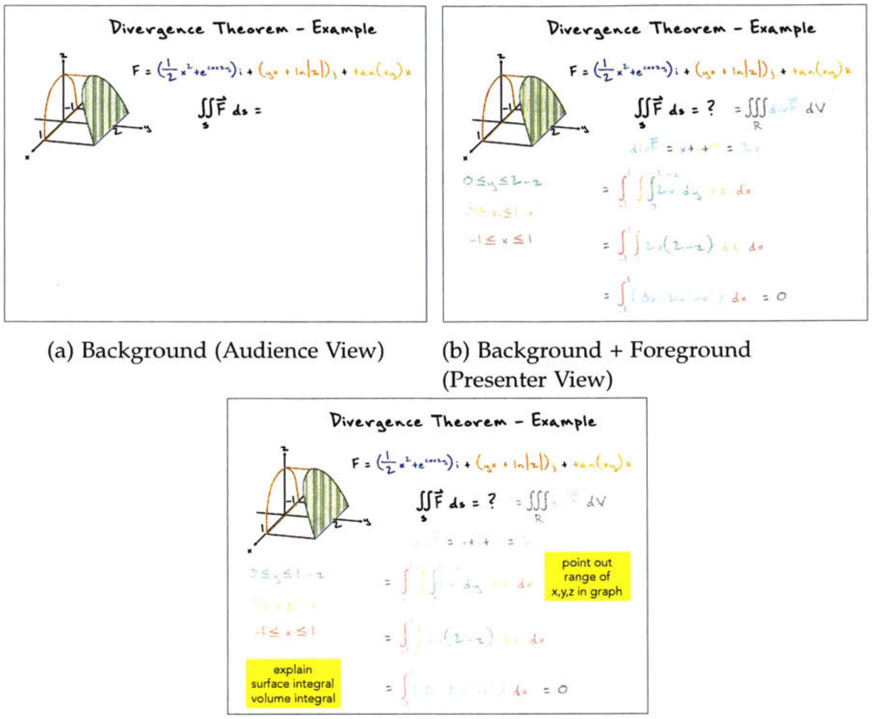
Aparecium enables robust, modeless inking operations at presentation time by leveraging the structure of the pre-authored slide content. Each of the three inking functions uses this structure in different ways.

4.4.2.1 *Reveal*

To reveal hidden foreground content to the audience, the presenter inks over the relevant portion of the foreground layer. At the end of each stroke, the system computes a subset of the surrounding foreground pixels to display. For each point, s_i on the stroke, the closest foreground pixel, p_i is computed. If the distance between p_i and s_i is within a threshold α, a flood-fill is performed starting from p_i to neighboring foreground pixels. The value of α determines how precisely the user has to ink in order to reveal the underlying content. Since presenters can be more precise when they are inking slowly versus when they are inking quickly, α is set to vary proportionally to inking velocity, v:

$$\alpha = 0.02v + 10 \tag{18}$$

where α is measured in pixels and v is measured in pixels per second.



(a) Background (Audience View)

(b) Background + Foreground (Presenter View)

point out range of x,y,z in graph

explain surface integral volume integral

(c) Background + Foreground + Notes

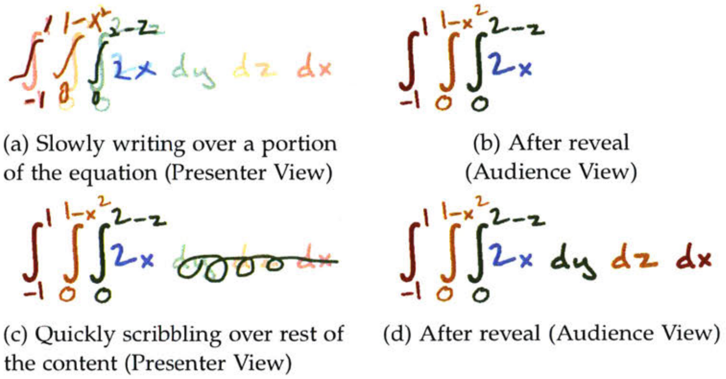
Figure 29: Slide Layers in Aparecium. Slides are separated into foreground and background layers. **(a)** The background layer is always visible and it is what the audience sees initially. **(b)** The foreground layer is initially only visible on the presenter view, and is faded to distinguish it from the background. Presenters can reveal parts of it to the audience during delivery. The revealed parts appear on the audience view and becomes unfaded on the presenter view. **(c)** Optionally, presenters can also have the notes layer, which is only visible on the presenter view and acts as transparent speaker notes placed on top of the slides.

The extent of the flood-fill is limited by two additional thresholds β and γ on (1) the distance from p_i, and (2) the color difference from p_i in CIELAB color space []. To give presenters finer-grained control over the extent of the reveal, β and γ also vary according to the velocity of the presenter's ink stroke:

$$\beta = \min(0.01v + 5,\ 40) \tag{19}$$

$$\gamma = 0.05v + 10 \tag{20}$$

Our algorithm has several important properties. Bounding the flood-fill with β and γ ensures that the revealed content is localized around the user stroke and prevents "bleeding" across regions with very different colors. In addition, modulating β and γ based on the stroke velocity allows presenters to reveal with different levels of precision. To show a small piece of content, the presenter can ink slowly over the relevant foreground region. This interaction is useful when the presenter wants to simulate writing in real-time, or needs to reveal a specific element in a dense slide. For example, in Figure 30(a), the presenter slowly writes out a part of the integral $\int_{-1}^{1}\int_{0}^{1-x^2}\int_{0}^{2-z} 2x$, while explaining each of the domains. In other situations, users may want to reveal larger pieces of content more efficiently. For example, in Figure 30(c), the presenter reveals all of $dx\,dy\,dz$ at once to complete the equation. In this case, users can ink quickly and roughly (e.g., by scribbling) over a foreground region.



(a) Slowly writing over a portion of the equation (Presenter View)

(b) After reveal (Audience View)

(c) Quickly scribbling over rest of the content (Presenter View)

(d) After reveal (Audience View)

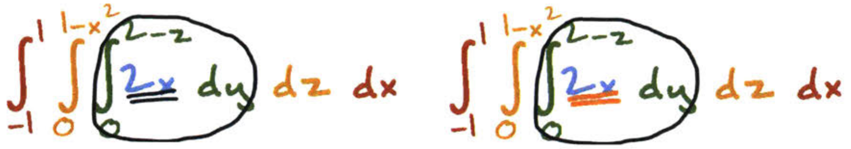
Figure 30: Inking to reveal. Presenter inks over foreground content to reveal it to the audience. Presenter can either **(a)** ink over slowly to simulate writing in real-time, or **(b)** scribble over quickly to reveal larger portions efficiently. **(c), (d)** In both cases, the relevant portion of the foreground layer is revealed to the audience and replaces the presenter's original ink strokes.

4.4.2.2 *Annotate*

Presenters sometimes add new (i.e., unauthored) information to a slide on-the-fly during a presentation. For example, a lecturer may circle an important concept for emphasis, explicitly label part of a diagram, or even add a whole new line of equation. To make such annotations in Aparecium, the presenter simply inks over empty (or already revealed) pixels on the foreground layer. If less than 10% of the s_is are within the α threshold of an unrevealed foreground pixel, the system treats the stroke as an annotation. To ensure that the annotation stands out from the surrounding slide content, Aparecium computes the average color of the slide around the stroke and sets the ink to a complementary color. (Figures 31 and 32c).

4.4.2.3 *Create Space*

In some cases, presenters may want extra space to insert new content in a slide, for example, to add an item to an existing list, a word in a sentence, or an extra line of explanation. These situations can arise as a result of a mistake in the preparation phase (e.g., the presenter forgets to list an item), as well as from presenter-audience interaction (e.g., the audience requests extra explanation). Aparecium allows pre-

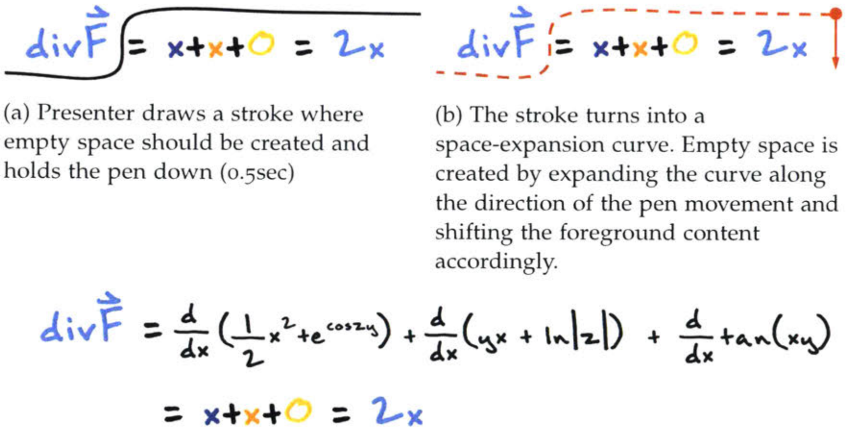

$\int_{-1}^{1}\int_{0}^{1-x^2}\int_{0}^{2-z} 2x\,dy\,dz\,dx$
$\int_{-1}^{1}\int_{0}^{1-x^2}\int_{0}^{2-z} 2x\,dy\,dz\,dx$

(a) Before auto-coloring

(b) After auto-coloring

Figure 31: Inking to annotate. **(a)** Presenter inks over already revealed content to make annotations. **(b)** To ensure that the annotation stands out from the surrounding slide content, Aparecium computes the average color of the slide around the stroke and sets the ink to a complementary color. In this case, the underlines below the cyan 2x is set to orange.

senters to create empty space from ink strokes. First, the presenter draws a curve where the empty space should be created. At the end of the curve, if the presenter holds down the pen for more than 0.5 seconds, the curve turns into a red dashed stroke, indicating that the presenter can start expanding the space around the curve. As the presenter moves the pen along one of the axis-aligned directions, empty space is created and expanded from the curve in that direction. As the space grows, the foreground content shifts accordingly. (Figure 32).


$\operatorname{div}\vec{F} = x+x+0 = 2x$
$\operatorname{div}\vec{F} = x+x+0 = 2x$
$\operatorname{div}\vec{F} = \frac{d}{dx}\left(\frac{1}{2}x^2+e^{\cos zy}\right) + \frac{d}{dx}\left(yx + \ln|z|\right) + \frac{d}{dx}\tan(xy)$
$= x+x+0 = 2x$

(a) Presenter draws a stroke where empty space should be created and holds the pen down (0.5sec)

(b) The stroke turns into a space-expansion curve. Empty space is created by expanding the curve along the direction of the pen movement and shifting the foreground content accordingly.

(c) Presenter inserts annotations in the newly created space

Figure 32: Creating space and annotating.

As space is created, the slide area grows and scrolling is automatically enabled. Users can also zoom out to fit the slide in a single view.

4.5 USER EVALUATION

We evaluate Aparecium from the perspective of presenters and the audience. We assess the ease of use of the interface from the presenters' point of view, and rate the presentation quality from both the presenters' and the audiences' perspective.

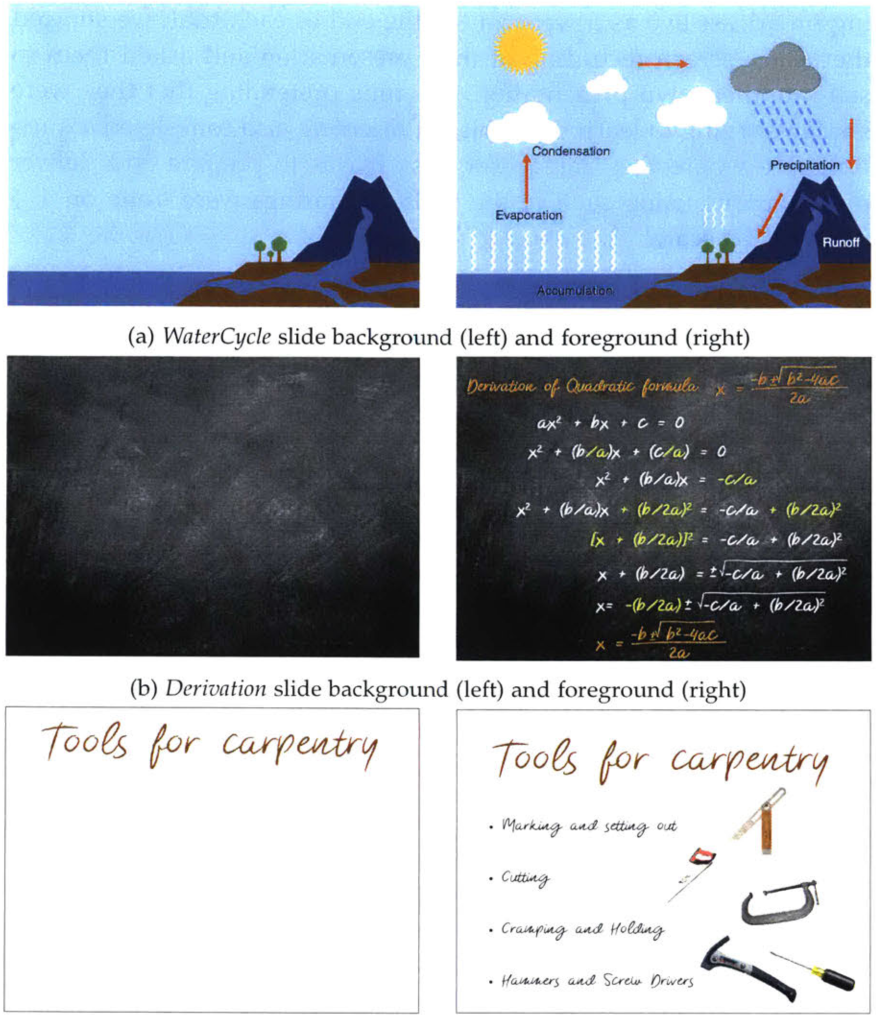
To better understand the benefits of our interface, we compared Aparecium against two baselines. The first baseline (**BaselinePPT**) represents conventional electronic slide tools. We use Microsoft PowerPoint and allow presenters to apply animation effects (but no inking) during the presentation. The second baseline (**BaselineInk**) represents standard "blackboard-style" presentation tools that allows users to ink in real-time during the presentation. For this condition, we also use Microsoft PowerPoint but only allow users to ink on top of the slides during preparation and delivery.

We hypothesized that different types of content would lend themselves to different presentation styles. We compare three different types of content: (1) a text-centered slide, explaining the derivation of the quadratic formula *(Derivation)*, (2) a diagram-centered slide, describing the hydrologic cycle *(WaterDiagram)*, and (3) a typical PowerPoint style slide with bullet points and images listing different carpentry tools *(BulletPoints)*. Using PowerPoint, we pre-authored a single page slide for each of these content types (Figure 33).

To reduce the preparation effort for all three presentation tool conditions, we separated each slide into foreground and background elements, as shown in Figure 33. For the BaselinePPT condition, we leave the individual PowerPoint elements (e.g., arrows, text boxes containing individual lines of equations, images) ungrouped to make it easier for users to author fine-grained animations. In fact, in the BaselinePPT condition users were free to animate any element including the background elements. Still, all presenters chose to keep the background elements static as in the default segmentation. We argue that this is because the content itself suggested a clear foreground-background separation, and that our default segmentation was not biased for or against any particular interface.

4.5.1 *Study 1: Presenter Perspective*

In our first study, we asked participants to present each of the three content types using the three different presentation tool conditions. For each content type, participants were given the pre-authored, pre-segmented slide. They received verbal and written explanations of the material, and a printout of the complete slide contents (both foreground and background layers). They were given time to familiarize themselves with this material and to set up the slide before the presentation.


Condensation
Precipitation
Evaporation
Runoff
Accumulation
Derivation of Quadratic formula $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$
$ax^2 + bx + c = 0$
$x^2 + (b/a)x + (c/a) = 0$
$x^2 + (b/a)x = -c/a$
$x^2 + (b/a)x + (b/2a)^2 = -c/a + (b/2a)^2$
$[x + (b/2a)]^2 = -c/a + (b/2a)^2$
$x + (b/2a) = \pm\sqrt{-c/a + (b/2a)^2}$
$x = -(b/2a) \pm \sqrt{-c/a + (b/2a)^2}$
$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$
Tools for carpentry
Tools for carpentry
• Marking and setting out
• Cutting
• Cramping and Holding
• Hammers and Screw Drivers

(a) *WaterCycle* slide background (left) and foreground (right)

(b) *Derivation* slide background (left) and foreground (right)

(c) *BulletPoints* slide background (left) and foreground(right)

Figure 33: Evaluation slides

In the BaselinePPT condition, participants could add animation effects to reveal or emphasize the foreground elements. For the BaselineInk condition, the slide only contained background elements. Participants were free to write additional content into the slide as part of their set up, which is analogous to blackboard lecturers writing on the board before class. For Aparecium, participants could also write additional content on top of the background, or they could choose to make some of the foreground elements visible (effectively moving those elements into the background layer). For simplicity, we omitted the space creating functionality in the evaluation.

After the setup phase, participants used one of the three tool conditions to deliver the presentation. To simulate a real presentation, participants were asked to pretend that their presentation was being broadcast live as a webcast. At the end of each trial, we showed the user a screen recording of their presentation and asked them to self-rate their own presentation, this time pretending that they were students trying to learn the subject. Presenters also completed a questionnaire where they rated how easy it was to prepare and deliver presentations using each of the tools. All ratings were done on a 5-point Likert scale.

We recruited 12 graduate students from 8 different universities (ages 21 to 31). All of the partcipants were familiar with the PowerPoint interface. We used a within-subject design, where each participant delivered presentations on each interface. We kept the task order fixed and counter-balanced the order of the presentation tools to obtain an equal distribution of task-tool pairs.

4.5.2 *Study 2: Audience Perspective*

In our second study, we recruited a separate set of participants to vote on presentations delivered using each interface. While we considered comparing the output presentations from the first study, we found that there was too much variation in quality between the results. Since presenters were not forced to follow fixed scripts, there were significant differences in length and detail. In addition, the presenters spoke with varying levels of enunciation and enthusiasm (not to mention different accents). Thus, we ourselves produced a new set of more comparable presentations using the slides from the first study. To make the comparison as fair as possible, we used a fixed script for each content type. We also analyzed the output presentations from the first study, and used them as a reference when creating our own versions. For example, for the BaselinePPT condition, we reproduced the granularity and order of animations that we observed in the user-created presentations. In the BaselineInk condition, we followed the typical ordering of the inked contents and chose similar ink colors. We also recorded the presentations so that the silent pauses in between inking periods were similar (or shorter) than those produced in the first study. Finally, for Aparecium, we used a similar reveal order, combination of slow tracing versus fast scribbling, and on-the-fly ink annotations as most participants. In general, presenters from the first study employed similar approaches to inking or animation so it was straightforward to extract common qualities. For the few cases, where participants varied in their approach, we selected an approach that we deemed to produce better quality (e.g., finer-grained animations, use of different ink colors).

We recruited 36 undergraduate and graduate students from multiple universities to rate the presentations. For each content type, participants watched recordings of three presentations delivered using each interface, and voted for the most engaging presentation.

4.5.3 *Findings and Discussion*

Overall, presenters found Aparecium easy to use for setting up and delivering presentations. They were satisfied with the quality of the presentations recorded using our interface, and emphasized that the appearance of real time inking had an engaging effect. Preferences between the presentation tools depended on the content type. Participants in both studies (i.e., presenters and viewers) preferred Aparecium for text-centered or process-driven content.

Given the size and exploratory nature of our study, we use descriptive statistics to summarize the Likert responses from Study 1. Below we discuss each of our findings in more detail.

4.5.3.1 *Aparecium makes it easier to prepare slides.*

Presenters found it easiest to set up the slides using our interface, followed by BaselineInk and then BaselinePPT (Figure 34). With Aparecium, most presenters did not do any extra work (revealing or writing beforehand) to set up the slides, but used them as-is. In the few cases, where they pre-revealed parts of the foreground, they expressed that the required effort was minimal.

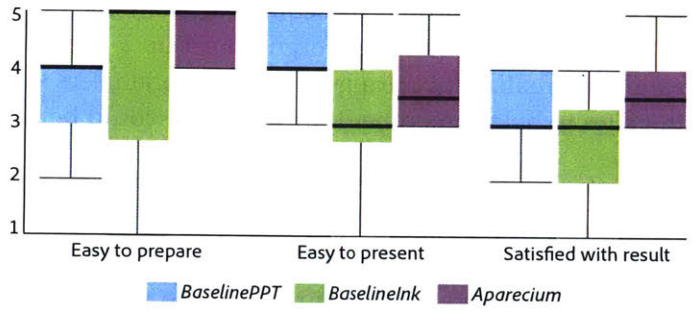


Figure 34: Summary of Likert responses from Study 1 on a scale from 1 (strongly disagree) to 5 (strongly agree).

In comparison, although our participants were familiar users of PowerPoint, they found the effort to set up animation effects tedious. To quote *U6*, *"It was cumbersome to add animations to each individual object and get the timing right... sometimes I decided I wanted to add an animation, but then had to figure out where to insert it in the existing animations sequence."*

In terms of preparation effort, the BaselineInk condition was similar to our interface; most participants used the slide as-is, without writing additional content ahead of time. However, the reasons were different. In Aparecium, presenters had the ability to reveal the pre-authored foreground elements to the audience during the presentation. In the BaselineInk condition, presenters had to manually draw the foreground content, which they could either do ahead of time (thus losing the real-time effect) or during delivery. Either way, the effort was more *"daunting" (U12)* and *"time consuming" (U8)*, and the result was aesthetically less satisfactory (*U1, U5, U8*).

4.5.3.2 *For presentation delivery, Aparecium involves comparable effort to BaselinePPT and less effort than BaselineInk.*

Regarding the ease of delivering presentations, BaselinePPT received slightly higher ratings than our interface. BaselineInk was rated the lowest by a larger margin. It is not surprising that BaselinePPT required the least effort. The only interaction presenters used was to press a single button to advance the slide animations. That said, when the animation involved more than a few steps, it was common for presenters to make mistakes. 3 out of 12 participants forgot to advance the animation at the right time at least once, and only realized it at the next animation step. They had to either repeat the verbal explanation or quickly skip through the subsequent animations. In addition, two participants reported that they forgot to set up a desired animation step and only realized it during delivery. Several users suggested that having cues to help remember the animation would be beneficial (U2, U7).

While presenters found inking in Aparecium straightforward, they noted that inking to reveal still required more effort than pushing a single button. Presenters seemed to prefer the path of minimum effort. For instance, with the exception of the math derivation content, they mostly used fast scribbling or strike-through gestures to reveal. As several presenters mentioned, this had the downside that the audience would initially see the scribbled ink strokes before the underlying pixels were revealed, which could potentially be distracting and aesthetically less pleasing. Some users suggested that they would prefer an even faster gesture such as clicking or circling to reveal large parts. On the other hand, they also expressed the idea that for the audience it could be better *"to have the word appear after the inking instead of just after clicking like in powerpoint." (U1)* We discuss this tradeoff in more detail in the Limitations and Discussion section. In addition, users appreciated the automatic color selection for annotation and the modeless switching between revealing and inking.

As expected, BaselineInk required the most effort during delivery. Participants complained that drawing took away their attention from the content delivery. Since verbal explanations tend to be faster, there were periods of silence while the presenters were still drawing. In the Derivation presentation, three out of the four presenters who used BaselineInk ran out of space and had to use the margins or eraser. The one exception was a presenter who used the setup time to layout line numbers on the slide. In addition to these challenges, some users also complained about the specific difficulty of writing on a tablet. For example, the screen interface is not as smooth as paper *(U1, U4, U6)* and operations such as switching to an eraser or a different color is tedious *(U7, U9)*.

4.5.3.3 *Presenters were most satisfied with the presentation quality of Aparecium.*

Presenters were most satisfied with the presentations produced using Aparecium. As shown in Figure 34, the distribution of ratings for BaselinePPT was slightly lower, while BaselineInk was rated the lowest. This trend held overall, as well as for each individual content type.

The feedback we gathered was consistent with our preliminary, formative interviews. Participants were accustomed to the BaselinePPT style, but in many cases they thought it could be improved with finer grained animation *(U1, U3, U4, U8, U11)*. They liked the *"handwritten in real time effect" (U4)* in Aparecium, as it made the presentation *"more interactive and seem to require engagement " (U9)*. The main complaints about BaselineInk presentations were that the drawings looked *"messy" (U7)* and *"not professional enough" (U6)*, and that the pace was too slow.

4.5.3.4 *Tool preference depends on presentation content.*

Figure 35 shows the preference data across content types from both studies. The tool preferences of both the presenters and audience depended on the content type.

In Study 1, we asked presenters to choose which interface they would prefer to use for each content type. For Derivation, presenters preferred Aparecium (8/12) and then BaselineInk (4/12). For WaterDiagram, Aparecium (6/12) and BaselinePPT(5/12) were comparable choices. Similarly for BulletPoints, BaselinePPT (7/12) and Aparecium (5/12) were preferred.

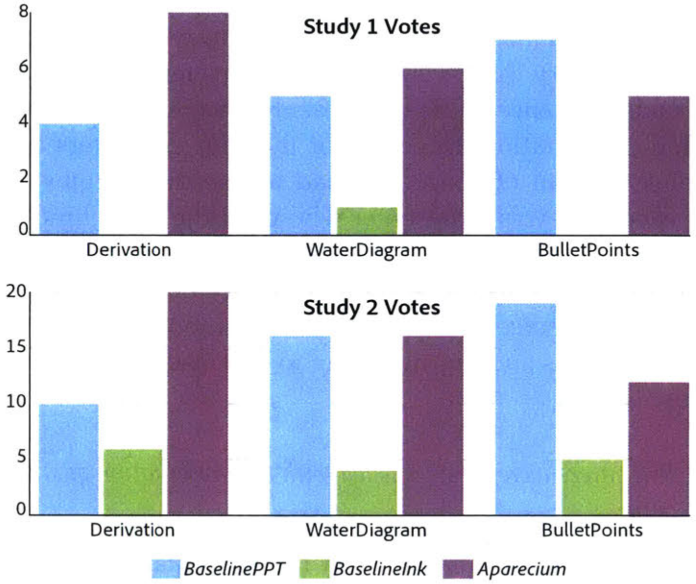


Figure 35: Preferred tool/presentation for each content type.

In Study 2, we asked audiences which presentation was most engaging. For Derivation, the majority of audiences preferred Aparecium (20/36) followed by BaselinePPT (10/36) and then BaselineInk (6/36). For WaterDiagram, Aparecium (16/36) and BaselinePPT (16/36) were comparable. For BulletPoints audiences preferred BaselinePPT (19/36) and then Aparecium (12/36).

These findings confirm our intuition that fine-grained control of pace is most important for presenting sequential processes, especially for writing out text or equations (Derivation). In the case of the WaterDiagram, it was easy to achieve similar sized steps using BaselinePPT and Aparecium. The fact that the WaterDiagram slide consists mainly of images rather than text may have affected the user's choice as well (see discussion about different ways to reveal). For simple lists and images (BulletPoint), BaselinePPT provided an appropriate pace and aesthetics.

4.6 LIMITATIONS AND DISCUSSION

Here, we consider some limitations of our design, and discuss several observations and lessons learned about designing future presentation interfaces.

4.6.0.1 *Scope of our User Study*

We conducted a focused user study to gather preliminary feedback about Aparecium. Specifically, in Study 1 (Presenter Perspective), we examine a controlled presentation scenario *without* presenter-audience interaction. Presenter-audience interaction is only one of the target scenarios that Aparecium proposes to improve, and Study 1 was designed to test other important aspects of presentation independent of audience interaction. For example, in our formative study, classroom and online lecturers expressed a desire to reduce preparation time, adjust the order of content, pace the presentation via real-time writing, and achieve neat handwriting even without audience feedback.

In order to limit the time of each user study session, Study 1 also used a single slide presentation. (Users had to prepare, deliver and evaluate 3 separate presentations.) This still allowed us to test the core features of Aparecium, which enables flexible presentation of content within a slide. Inter-slide transitions are the same as traditional tools.

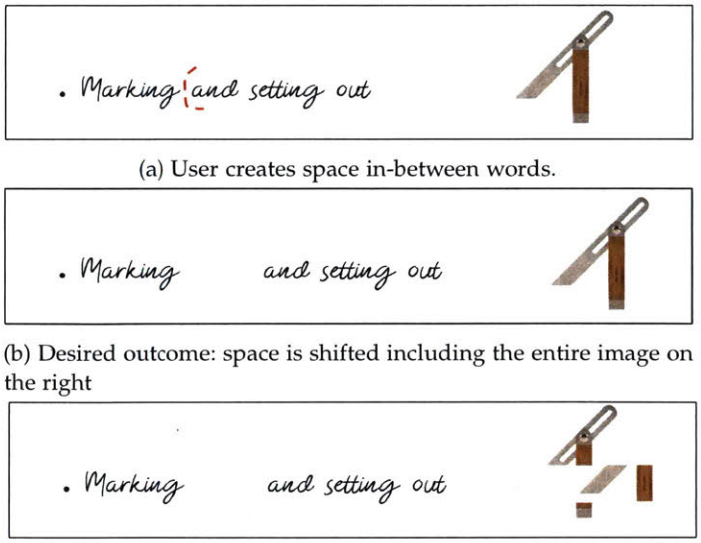
Similarly, Study 2 (Audience Perspective) is intended to be a preliminary study to gather audience assessment of presentations in a limited setting. How audience experience is affected by classroom interaction versus online lectures, by presentation content or duration are interesting research areas for future work. Note that both Study 1 and Study 2 are similar to an online lecture recording and viewing scenario respectively.

Finally, besides the limited set up stage, we do not test the authoring stage of the presentation. There are many interesting research questions related to the authoring aspect. For example, would different tools encourage presenters to author different content? In general, a longitudinal study based on a classroom deployment or an online course would be valuable to test different aspects of Aparecium.

4.6.0.2 *Space Manipulation*

We present a simple interaction to create space within the slide in a limited way, but there are many scenarios which our interaction cannot support. For example, in Figure 36, the user wants to create space to insert a word. Since the system is not content-aware, if the user is not careful the curve can shift the space in such a way as to break the image on the right.

Figure 37 illustrates a different scenario where the user wants to create space in-between the bullet points without pushing the image further down (i.e., by compressing the space between the bullet points and the image). Our space-expansion curve would cut the slide into two sides and moves the entire lower region including the image.


• Marking and setting out
• Marking and setting out
• Marking and setting out

(a) User creates space in-between words.

(b) Desired outcome: space is shifted including the entire image on the right

(c) Current outcome: the image to the right is broken by the extent of the curve.

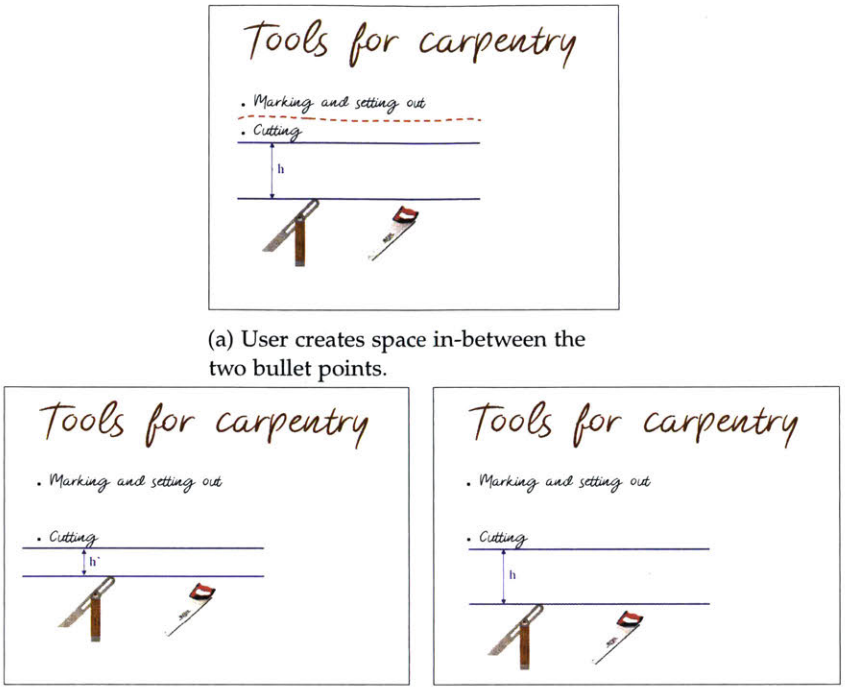
Figure 36: Limitations of our space manipulation interaction. In this scenario, user wants to create space to insert a word. Our implementation is not content aware. Only the space strictly to the right of the curve is shifted breaking the image to the right.

4.6.0.3 *Different Ways to Reveal*

Several presenters in the first user study wanted an even quicker way than scribbling to reveal content. Indeed, scribbling over large areas to reveal images or long lines of text can be tedious, especially if the presenter is not concerned about simulating the real-time hand-drawn effect.

In addition, some participants across both user studies complained that showing the scribbles before revealing the foreground elements is less aesthetically pleasing or even distracting. Conversely, other audience members commented that the scribbles served as helpful cues to draw attention to where information was about to appear.

In general, supporting different types of revealing mechanisms (e.g., clicking or lasso selection) and allowing instantaneous reveal of certain elements could improve the presentation experience. However, such a design should not increase the preparation effort or limit flexibility during presentation. For example, we considered allowing presenters to specify at setup time elements that can be revealed instantaneously upon clicking or tapping. However, this can make preparation tedious, akin to grouping elements and setting up animations in PowerPoint. Moreover, it does not allow presenters to change their mind during preparation about how to reveal elements.



(a) User creates space in-between the two bullet points.

(b) Desired outcome: space is created by compressing the extra space between the bullet points and the images.

(c) Current outcome: slide is cut at the curve and the entire lower region is shifted.

Figure 37: Limitations of our space manipulation interaction. In this scenario, user wants to create space in-between the bullet points without shifting the images. Our implementation does not support this case. Instead, the space-expansion curve cuts the slide into two and moves the entire lower region.

4.6.0.4 *Presenter Convenience vs. Presentation Quality*

The question of what revealing mechanisms to support points to a more fundamental tension with presenter interfaces. On the one hand, making the workflow more convenient for presenters is an important objective. However, it is at least as important to take into account the audience's point of view, since they are the intended consumers of the presentations themselves. For example, providing a larger set of revealing gestures may tempt presenters to always opt for the minimum-effort gesture at the cost of presentation quality. In fact, in an earlier prototype of our system, we allowed revealing either by lasso selection or slow tracing, and observed that presenters almost always opted for the lasso tool even for the *Derivation* type of content, which compromised presentation quality for the audience. Instead, we opted for a design that *forces* the presenter to go at a slower pace.

More broadly, designing presentation tools that achieve the right balance between the needs of presenters and audience members remains an interesting direction for future work. In this vein, previous research that systematically compares the effect of different presentation styles (e.g., [35, 111]) may provide valuable guidance for new interfaces that assist and *nudge* presenters towards the most effective communication techniques.

4.6.0.5 *Creating Flexible Interactivity*

Our work focuses on presenting slides with static visuals, and Aparecium supports limited interaction with the displayed content (i.e., shifting elements to create empty space). In our formative interviews, several participants expressed the desire to present interactive content that can be controlled on-the-fly, for instance, to explain the steps of a computer algorithm or a physics diagram. While there are many existing approaches to creating interactive diagrams, designing authoring and presentation tools that are both convenient and flexible is a challenging problem that bears further exploration.

4.6.1 *Conclusion*

This chapter introduced Aparecium, a novel presentation interface that combines inking interactions with pre-authored slides to help presenters deliver flexible and engaging presentations.

Our modeless inking interactions allow presenters to reveal, annotate and create extra space during a presentation. We evaluated our interface from the perspective of presenters and the audience by comparing it against two baseline interfaces, representing conventional slide and inking tools. Presenters found our interface easy to use both during preparation and delivery. In particular, for text-heavy, process-driven content, both presenters and audience preferred presentations delivered using Aparecium.

In light of these findings, we believe our inking interactions could be a valuable addition to existing slide-based presentation tools.

5 CONCLUSION

In my mind, very few people are truly literate with new media.
Would we consider someone literate with traditional media
if they could read but not write?

— Mitchel Resnick [103]

5.1 CONTRIBUTIONS

We began in the introduction with the goal of designing effective interfaces to make it easier for users to author, edit and navigate audiovisual media. We looked at three specific domains: navigating lecture videos, authoring speech recordings and delivering slide presentations. In each case, we proposed a design solution to facilitate the user task, implemented the solution in a prototype interface, and provided preliminary evaluation through focused user studies. In doing so, we fleshed out several principles for designing effective audiovisual media interfaces. These principles apply broadly to a wide range of applications and user tasks:

5.1.0.1 *Finding and Imposing Spatial & Temporal Structure in Media*

An essential aspect of audiovisual media production is iterative editing. Efficient navigation and linking between related parts are key to facilitating this process.

In a single author scenario, authors can rely on their unique knowledge of the media. For instance, the narrator may remember the timing of the specific point in the audio that needs to be re-recorded. Similarly, if the editor knows the contents of the recordings, it is easier to find an alignment between them. However, as the size of the media or the number of input stream grows, or as multiple people work together, users have to navigate and edit media which they are not familiar with. This becomes very time-consuming. Even simple operations such as alignment and synchronization can be tedious and distract the editors' attention from the content.

As we have seen in our work, inferring inherent structures from the media, and imposing meaningful structures in their representation can alleviate this problem. For example, VisualTranscripts infers the semantic relationship between individual strokes and re-groups them into meaningful static figures. VoiceScript uses a text-based representation and explicitly visualizes the alignments between the script and the audio recording spatially, side-by-side. In both instances, structure is inferred from a temporal medium and then transformed into a static, spatial representation. Static representations have the advantage that they are easy to visually navigate (including skimming and searching), spatially organize, and edit.

Similar approaches can be extended to a wide range of applications. For example, static representations of animations, similar to VisualTranscripts or motion illustrations [25], can be used to facilitate post-editing and then reverse-engineered to render the edited animations. Moreover, recent advances in audio and video analysis established new methods to extract and manipulate less obvious structures from audiovisual media. For example, [89] introduced a novel algorithm to separate sources from a combined audio signal. [36] analyzes small vibrations recorded in a video to infer sound and material properties of objects. Interfaces must capitalize and appropriately expose such structures so that users can understand and manipulate the media with ease.

5.1.0.2 *Utilizing Various Modalities for Input & Output*

Another key concern for audiovisual media interfaces is the need to support a fluid and spontaneous workflow. For instance, users often want to explore multiple alternatives, exchange feedback, make and undo changes on-the-fly, and visualize the consequences. These processes usually involve working with and combining multiple modalities, both as input and as output.

As input, users often want low-effort, less formal methods of interaction. For example, animators can produce a quick draft of a story by sketching and producers can exchange feedback about a video using e-mails. Currently, most interfaces for audiovisual media do not support such informal interactions and their output (e.g., sketch, email) must be manually translated into the actual media (e.g., animation, video).

Our work explored the idea of combining structured data with less structured interactions. VoiceScript integrates script text (structured data) with improvised speech (unstructured interaction). Aparecium proposes inking gestures (unstructured interaction) as a main modality to present pre-authored slides (structured data). Similar ideas can be applied to a range of applications, for example, integrating sketching into UX design, or using verbal annotations in audio authoring. Interfaces must seek to support natural modes of interaction in real-time and integrate the output of these interactions to the media in a meaningful way.

The principle of employing multiple modalities applies not only to user input but also to how the media is represented. VisualTranscript represents lecture videos with images, text and videos. VoiceScript represents speech recordings as both audio and text. Different modalities not only facilitate different tasks but work together to give a richer view of the media.

5.1.0.3 *Supporting Direct Manipulation with Automation*

Automatic algorithms aim to free the users from tedious tasks and allow them to focus instead on the creative part of the workflow. In VoiceScript, we use automatic alignment and segmentation so users can focus on making the content of the speech recording. In Aparecium, we take advantage of the pre-authored slide to achieve handwriting beautification effects, which relieves the users from worrying about writing beautifully on the fly.

Obviously, this is not a new principle. Hybrid approaches, combining automation with direct manipulation, have been proposed and successfully implemented in many application domains, including authoring of 3D models [45, 101], animation [12], and illustrations [25, 127].

A specific challenge for audiovisual media interfaces is to blend automation with fine-grained user control. It is important that users feel they have direct control over the media that they are authoring or navigating. Intuitive ways to explore the parameters of automatic algorithms, and manually change their outputs or part of their outputs should be an important part of interface design.

5.2 FUTURE DIRECTIONS

5.2.1 *Facilitating Collaborative Authoring of Audiovisual Media*

This dissertation looked mostly at single user scenarios. Chapter 3 briefly discussed asynchronous collaboration to author voice recordings, but still the main focus was on facilitating the workflow of a single user. To make interfaces truly effective and to empower users even further, supporting collaboration is essential.

Online collaborative editing is a growing theme across a number of domains and we are seeing a rise of new web-based editors such as OnShape for 3D modeling [95] or Prezi for presentations [102]. Cloud-based platforms such as Adobe Creative Cloud [2] or Google Drive [51] are also making it easier to collaborate and share assets among multiple users and devices.

However, major challenges remain to support web-based collaboration for audiovisual media. In particular, traditional data structures for audio or video were developed for single user, single machine interfaces. They are unwieldy even to simply load or share. We must rethink how we store and represent audiovisual data in order to effectively address issues that arise with collaboration, such as conflict resolution and tracking and visualization of edit history.

The goal is to enable novices, experts and distributed online communities to collaborate directly in real-time to author compelling audio recordings, videos, animations etc., and make the experience as smooth and easy as using Google Docs. This effort will naturally lead to the invention of new types of media and social platforms such as we have seen with Wikipedia [123] for text documents.

5.2.2 *Interactive Media*

A common theme that recurred in our work was supporting nonlinear interactions with media. We looked at searching in lecture videos, iterative editing of speech recordings and flexible presentation of slide materials. In these applications, the media themselves (video, audio and slides) were fundamentally linear and had a natural order imposed by time. It was the user tasks which required nonlinear interactions.

New types of media such as virtual reality (VR), augmented reality (AR) and 360-degree videos bring an added challenge because the media themselves are nonlinear. The superabundance of information in these media can easily overwhelm the users. What is an efficient way to edit virtual reality scenes? How can directors make sure that the audience don't miss the important scene in a 360-degree film? What is the equivalent of a timeline for such media?

Interactivity is an attractive property for media, and it has been shown to be an important factor in education as well as entertainment. However, creating and exploring interactive media has always been a challenge, whether it is an interactive simulation, game or animation. As the dimensionality and complexity of media grow, the principles outlined in this dissertation will be even more important. Taking advantage of inherent structures, multiple modalities and automatic algorithms can facilitate and direct our engagement with interactive media.

5.2.3 *Human-Interface-Media Interaction*

The Merriam-Webster dictionary defines **interface** as *the place at which independent and often unrelated systems meet and act on or communicate with each other.* As such, interfaces affect how users engage with the media and even what kind of media users produce.

We saw some examples of how interface affects users and media in our user studies. In the comparative study for navigating lecture videos (chapter 2.6), when the transcript text for the video was not structured users relied on the video instead even when they were looking for specifically textual information. On the other hand, with VisualTranscript, users preferred to read the text and the figures instead of playing the video. In the case of presentation interfaces (chapter 4), we did not allow instantaneous animation effects in Aparecium because we found that it led presenters to give fast presentations.

Along with the development of new interfaces, systematic studies should be made that investigate the interaction between human, interface and media. For instance, high dropout rates are a common problem for online lecture videos. Can we design video browsers that encourage students to stay and even pay attention for the entire lecture? Should there be a different kind of browser for students with attention deficiency disorder? How can a presentation interface balance between the audiences' needs and the presenters' needs? Can video editing interfaces boost users creativity to help them make more interesting videos? A deep understanding of human psychology, interface technology and media design principles and their relationship to each other will contribute to creating better interfaces and better media.

BIBLIOGRAPHY

[1] *Adobe Audition*. http://www.adobe.com/products/audition.html. Accessed: 2016-04-02. Apr. 2016.

[2] *Adobe Creative Cloud*. http://www.adobe.com/creativecloud.html/. Accessed: 2017-05-28. 2017.

[3] *Adobe Premier*. Accessed: 2016-04-11. Apr. 2016.

[4] *Adobe Story*. https://story.adobe.com/en-us/. Accessed: 2016-04-02. 2016.

[5] Richard J Anderson, Crystal Hoyer, Steven A Wolfman, and Ruth Anderson. "A study of digital ink in lecture presentation." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2004, pp. 567–574.

[6] Richard Anderson, Peter Davis, Natalie Linnell, Craig Prince, V Razmo, and Fred Videon. "Classroom presenter: Enhancing interactive education with digital ink." In: *Computer* 40.9 (2007).

[7] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. "Automatic editing of footage from multiple social cameras." In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 81.

[8] Barry Michael Arons. "Interactively skimming recorded speech." PhD thesis. Massachusetts Institute of Technology, 1994.

[9] Barry Arons. "SpeechSkimmer: a system for interactively skimming recorded speech." In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 4.1 (1997), pp. 3–38.

[10] *Audacity*. Accessed: 2016-04-02. Apr. 2016.

[11] *Avid Protools*. Accessed: 2016-04-02. Apr. 2016.

[12] Yunfei Bai, Danny M Kaufman, C Karen Liu, and Jovan Popović. "Artist-directed dynamics for 2D animation." In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 145.

[13] Connelly Barnes, Dan B Goldman, Eli Shechtman, and Adam Finkelstein. "Video tapestries with continuous temporal zoom." In: *ACM Transactions on Graphics (TOG)* 29.4 (2010), p. 89.

[14] Thomas Baudel and Michel Beaudouin-Lafon. "Charade: remote control of objects using free-hand gestures." In: *Communications of the ACM* 36.7 (1993), pp. 28–35.

[15] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. "Tools for placing cuts and transitions in interview video." In: *ACM Trans. Graph.* 31.4 (2012), pp. 67–1.

[16] H.M. Boettinger. *Moving mountains, or, The art and craft of letting others see things your way*. Collier Books, 1989. URL: https://books.google.com/books?id=eQLYAAAAMAAJ.

[17] John Boreczky, Andreas Girgensohn, Gene Golovchinsky, and Shingo Uchihashi. "An interactive comic book presentation for exploring video." In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM. 2000, pp. 185–192.

[18] Rita Borgo, Min Chen, Ben Daubney, Edward Grundy, Heike Janicke, Gunther Heidemann, Benjamin Hoferlin, Markus Hoferlin, Daniel Weiskopf, and Xianghua Xie. "A survey on video-based graphics and video visualization." In: *Proc. of the EuroGraphics conf., State of the Art Report*. Citeseer. 2011, pp. 1–23.

[19] Nicholas J Bryan, Paris Smaragdis, and Gautham J Mysore. "Clustering and synchronizing multi-camera video via landmark cross-correlation." In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE. 2012, pp. 2389–2392.

[20] Xiang Cao, Eyal Ofek, and David Vronay. "Evaluation of alternative presentation control techniques." In: *CHI'05 Extended Abstracts on Human Factors in Computing Systems*. ACM. 2005, pp. 1248–1251.

[21] Juan Casares, A Chris Long, Brad A Myers, Rishi Bhatnagar, Scott M Stevens, Laura Dabbish, Dan Yocum, and Albert Corbett. "Simplifying video editing using metadata." In: *Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM. 2002, pp. 157–166.

[22] *Celtx*. Accessed: 2016-04-02. Apr. 2016.

[23] Fan Chen and Christophe De Vleeschouwer. "Formulating team-sport video summarization as a resource allocation problem." In: *IEEE Transactions on Circuits and Systems for Video Technology* 21.2 (2011), pp. 193–205.

[24] Kelvin Cheng and Kevin Pulo. "Direct interaction with large-scale display systems using infrared laser tracking devices." In: *Proceedings of the Asia-Pacific symposium on Information visualisation-Volume 24*. Australian Computer Society, Inc. 2003, pp. 67–74.

[25] Pei-Yu Peggy Chi, Daniel Vogel, Mira Dontcheva, Wilmot Li, and Björn Hartmann. "Authoring Illustrations of Human Movements by Iterative Physical Demonstration." In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM. 2016, pp. 809–820.

[26] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. "MixT: automatic generation of step-by-step mixed media tutorials." In: *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM. 2012, pp. 93–102.

[27] Pei-Yu Chi, Joyce Liu, Jason Linder, Mira Dontcheva, Wilmot Li, and Bjoern Hartmann. "Democut: generating concise instructional videos for physical demonstrations." In: *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM. 2013, pp. 141–150.

[28] Chekuri Choudary and Tiecheng Liu. "Summarization of visual content in instructional videos." In: *Multimedia, IEEE Transactions on* 9.7 (2007), pp. 1443–1455.

[29] Michael G Christel and Adrienne S Warmack. "The effect of text in storyboards for video navigation." In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. Vol. 3. IEEE. 2001, pp. 1409–1412.

[30] Michael G Christel, Alexander G Hauptmann, Howard D Wactlar, and Tobun D Ng. "Collages as dynamic summaries for news video." In: *Proceedings of the tenth ACM international conference on Multimedia*. ACM. 2002, pp. 561–569.

[31] Wen-Sheng Chu, Yale Song, and Alejandro Jaimes. "Video Co-Summarization: Video Summarization by Visual Co-Occurrence." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[32] Wen-Sheng Chu, Yale Song, and Alejandro Jaimes. "Video co-summarization: Video summarization by visual co-occurrence." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3584–3592.

[33] Matthew Cooper and Jonathan Foote. "Summarizing popular music via structural similarity analysis." In: *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on*. IEEE. 2003, pp. 127–130.

[34] Timothee Cour, Chris Jordan, Eleni Miltsakaki, and Ben Taskar. "Movie/script: Alignment and parsing of video and text transcription." In: *Computer Vision–ECCV 2008* (2008), pp. 158–171.

[35] Andrew Cross, Mydhili Bayyapunedi, Edward Cutrell, Anant Agarwal, and William Thies. "TypeRighting: combining the benefits of handwriting and typeface in online educational videos." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2013, pp. 793–796.

[36] Abe Davis, Michael Rubinstein, Neal Wadhwa, Gautham J Mysore, Frédo Durand, and William T Freeman. "The visual microphone: passive recovery of sound from video." In: (2014).

[37] Leo Degen, Richard Mander, and Gitta Salomon. "Working with audio: integrating personal tape recorders and desktop computers." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 1992, pp. 413–418.

[38] Steven M Drucker, Georg Petschnigg, and Maneesh Agrawala. "Comparing and managing multiple versions of slide presentations." In: *Proceedings of the 19th annual ACM symposium on User interface software and technology*. ACM. 2006, pp. 47–56.

[39] Bruno Dumas, Denis Lalanne, and Sharon Oviatt. "Multimodal interfaces: A survey of principles, models and frameworks." In: *Human machine interaction* (2009), pp. 3–26.

[40] Ahmet Ekin, A Murat Tekalp, and Rajiv Mehrotra. "Automatic soccer video analysis and summarization." In: *Image Processing, IEEE Transactions on* 12.7 (2003), pp. 796–807.

[41] *Final Draft*. https://www.finaldraft.com/. Accessed: 2016-04-02. Apr. 2016.

[42] Jakub Fišer, Paul Asente, and Daniel Sỳkora. "Shipshape: A drawing beautification assistant." In: *Proceedings of the workshop on Sketch-Based Interfaces and Modeling*. Eurographics Association. 2015, pp. 49–57.

[43] Michael Fleischman, Brandon Roy, and Deb Roy. "Temporal feature induction for baseball highlight classification." In: *Proceedings of the 15th ACM international conference on Multimedia*. ACM. 2007, pp. 333–336.

[44] Barbara A Frey and David J Birnbaum. "Learners' Perceptions on the Value of PowerPoint in Lectures." In: (2002).

[45] Ran Gal, Olga Sorkine, Niloy J Mitra, and Daniel Cohen-Or. "iWIRES: an analyze-and-edit approach to shape manipulation." In: *ACM Transactions on Graphics (TOG)*. Vol. 28. 3. ACM. 2009, p. 33.

[46] *GarageBand*. Accessed: 2016-04-02. Apr. 2016.

[47] Yashesh Gaur. "The Effects of Automatic Speech Recognition Quality on Human Transcription Latency." In: *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*. ACM. 2015, pp. 367–368.

[48] Evan Golub. "Handwritten slides on a tabletPC in a discrete mathematics course." In: *ACM SIGCSE Bulletin*. Vol. 36. 1. ACM. 2004, pp. 51–55.

[49] Lance Good and Benjamin B Bederson. "Zoomable user interfaces as a medium for slide show presentations." In: *Information Visualization* 1.1 (2002), pp. 35–49.

[50] *Google Docs*. Accessed: 2016-04-02. Apr. 2016.

[51] *Google Drive*. https://www.google.com/drive/. Accessed: 2017-05-28. 2017.

[52] *Google Slides*. https://www.google.com/slides/about/. Accessed: 2017-04-1. 2017.

[53] Gary Hardock, Gordon Kurtenbach, and William Buxton. "A marking based interface for collaborative writing." In: *Proceedings of the 6th annual ACM symposium on User interface software and technology*. ACM. 1993, pp. 259–266.

[54] William Rainey Harper. *Chautauqua Movement*. Retrieved May 17, 2017 from https://archive.org/stream/ChautauquaMovement/ChautauquaMovement_djvu.txt. 1886.

[55] Alexander G Hauptmann and Michael J Witbrock. "Informedia: News-on-demand multimedia information acquisition and retrieval." In: *Intelligent multimedia information retrieval* (1997), pp. 215–239.

[56] Liwei He, Elizabeth Sanocki, Anoop Gupta, and Jonathan Grudin. "Auto-summarization of audio-video presentations." In: *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*. ACM. 1999, pp. 489–498.

[57] Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. "3D puppetry: a kinect-based interface for 3D animation." In: *UIST*. Citeseer. 2012, pp. 423–434.

[58] *Hindenburg Journalist Pro*. Accessed: 2016-04-02. Apr. 2016.

[59] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdelrahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97.

[60] HowStuffWorks.com. "What is a decibel, and how is it measured?" In: (2000). Accessed: 2016-04-11.

[61] Heloise Hwawen Hse and A Richard Newton. "Recognition and beautification of multi-stroke symbols in digital ink." In: *Computers & Graphics* 29.4 (2005), pp. 533–546.

[62] Won-Il Hwang, Pyung-Jun Lee, Bong-Kyung Chun, Dong-Sung Ryu, and Hwan-Gue Cho. "Cinema comics: Cartoon generation from video stream." In: *GRAPP*. 2006, pp. 299–304.

[63] *IBM Speech to Text Service*. Accessed: 2016-04-02. Apr. 2016.

[64] Takeo Igarashi, Sachiko Kawachiya, Hidehiko Tanaka, and Satoshi Matsuoka. "Pegasus: a drawing system for rapid geometric design." In: *CHI 98 conference summary on Human factors in computing systems*. ACM. 1998, pp. 24–25.

[65] Rick Kazman, Reem Al-Halimi, William Hunt, and Marilyn Mantei. "Four paradigms for indexing video conferences." In: *IEEE multimedia* 3.1 (1996), pp. 63–73.

[66] *Keynote*. http://www.apple.com/keynote/. Accessed: 2017-03-27. 2017.

[67] Aditya Khosla, Raffay Hamid, Chih-Jen Lin, and Neel Sundaresan. "Large-scale video summarization using web-image priors." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 2698–2705.

[68] Gunhee Kim, Leonid Sigal, and Eric P Xing. "Joint summarization of large-scale collections of web images and videos for storyline reconstruction." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 4225–4232.

[69] Juho Kim, Philip J Guo, Carrie J Cai, Shang-Wen Daniel Li, Krzysztof Z Gajos, and Robert C Miller. "Data-driven interaction techniques for improving navigation of educational videos." In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM. 2014, pp. 563–572.

[70] Donald E Knuth and Michael F Plass. "Breaking paragraphs into lines." In: *Software: Practice and Experience* 11.11 (1981), pp. 1119–1184.

[71] Joseph J LaViola Jr and Robert C Zeleznik. "MathPad 2: a system for the creation and exploration of mathematical sketches." In: *ACM SIGGRAPH 2007 courses*. ACM. 2007, p. 46.

[72] Joel Lanir, Kellogg S Booth, and Leah Findlater. "Observing presenters' use of visual aids to inform the design of classroom presentation software." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2008, pp. 695–704.

[73] Gierad P Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. "Pixeltone: A multimodal interface for image editing." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2013, pp. 2185–2194.

[74] Andrew Large, Jamshid Beheshti, Alain Breuleux, and Andre Renaud. "Multimedia and comprehension: The relationship among text, animation, and captions." In: *Journal of the American Society for Information Science* 46.5 (1995), pp. 340–347.

[75] Cheng-Te Li and Man-Kwan Shan. "Emotion-based impressionism slideshow with automatic music accompaniment." In: *Proceedings of the 15th ACM international conference on Multimedia*. ACM. 2007, pp. 839–842.

[76] Yang Li, James A Landay, Zhiwei Guan, Xiangshi Ren, and Guozhong Dai. "Sketching informal presentations." In: *Proceedings of the 5th international conference on Multimodal interfaces*. ACM. 2003, pp. 234–241.

[77] Leonhard Lichtschlag, Thorsten Karrer, and Jan Borchers. "Fly: a tool to author planar presentations." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2009, pp. 547–556.

[78] Alex Limpaecher, Nicolas Feltman, Adrien Treuille, and Michael Cohen. "Real-time Drawing Assistance Through Crowdsourcing." In: *ACM Trans. Graph.* 32.4 (July 2013), 54:1–54:8. ISSN: 0730-0301.

[79] James Lin, Mark W Newman, Jason I Hong, and James A Landay. "DENIM: finding a tighter fit between tools and practice for Web site design." In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM. 2000, pp. 510–517.

[80] Feng Liu, Yu-hen Hu, and Michael L Gleicher. "Discovering panoramas in web videos." In: *Proceedings of the 16th ACM international conference on Multimedia*. ACM. 2008, pp. 329–338.

[81] Zheng Lu and Kristen Grauman. "Story-driven summarization for egocentric video." In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE. 2013, pp. 2714–2721.

[82] Jeremy Lybarger. *Can Photojournalism Survive in the Instagram Era?* Retrieved April 21, 2017 from http://www.motherjones.com/media/2013/07/bending-the-frame-fred-ritchin-photojournalism-instagram. 2013.

[83] David Macaulay. *The new way things work*. Scholastic, 1999.

[84] Catherine C Marshall, Morgan N Price, Gene Golovchinsky, and Bill N Schilit. "Collaborating over portable reading appliances." In: *Personal Technologies* 3.1-2 (1999), pp. 43–53.

[85] *Micorsoft Word*. Accessed: 2016-04-02. Apr. 2016.

[86] Toni-Jan Keith Palma Monserrat, Shengdong Zhao, Kevin McGee, and Anshul Vikram Pandey. "NoteVideo: Facilitating navigation of blackboard-style lecture videos." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2013, pp. 1139–1148.

[87] Tomer Moscovich, Karin Scholz, John F Hughes, and D Salesin. *Customizable presentations*. Tech. rep. Technical Report CS-04-16, Computer Science Department, Brown University, 2004.

[88] Elizabeth D Mynatt, Takeo Igarashi, W Keith Edwards, and Anthony LaMarca. "Flatland: new dimensions in office whiteboards." In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM. 1999, pp. 346–353.

[89] Gautham J Mysore, Paris Smaragdis, and Bhiksha Raj. "Nonnegative hidden Markov modeling of audio with application to source separation." In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2010, pp. 140–148.

[90] Yuto Nara, Yukua Koide, Wataru Fujimura, Genki Kunitomi, and Akihiko Shirai. *Manga Generator: Immersive Posing Role Playing Game in Manga World*. Retrieved May 17, 2017 from https://www.youtube.com/watch?v=bXR163IjGM0. 2013.

[91] Saul B Needleman and Christian D Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." In: *Journal of molecular biology* 48.3 (1970), pp. 443–453.

[92] Les Nelson, Satoshi Ichimura, Elin Rønby Pedersen, and Lia Adams. "Palette: a paper interface for giving presentations." In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM. 1999, pp. 354–361.

[93] Chong-Wah Ngo, Yu-Fei Ma, and Hong-Jiang Zhang. "Video summarization and scene detection by graph modeling." In: *Circuits and Systems for Video Technology, IEEE Transactions on* 15.2 (2005), pp. 296–305.

[94] J.C. Oates and L. Milazzo. *Conversations with Joyce Carol Oates*. Literary conversations series. University Press of Mississippi, 1989. ISBN: 9780878054121. URL: https://books.google.com/books?id=wp030C3f9cUC.

[95] *Onshape*. https://www.onshape.com/. Accessed: 2017-05-28. 2017.

[96] Norman Papernick and Alexander G Hauptmann. "Summarization of broadcast news video through link analysis of named entities." In: *In AAAI-05 Workshop on link analysis*. 2005.

[97] Amy Pavel, Björn Hartmann, and Maneesh Agrawala. "Video digests: a browsable, skimmable format for informational lecture videos." In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM. 2014, pp. 573–582.

[98] Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. "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. 2015, pp. 181–190.

[99] Marcus J Pickering, Lawrence Wong, and Stefan M Rüger. "ANSES: Summarisation of news video." In: *Image and Video Retrieval*. Springer, 2003, pp. 425–434.

[100] *PowerPoint*. https://products.office.com/en-us/powerpoint/. Accessed: 2017-03-27. 2017.

[101] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. "Make it stand: balancing shapes for 3D fabrication." In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 81.

[102] *Prezi*. https://prezi.com/. Accessed: 2017-04-1. 2017.

[103] Mitchel Resnick. *Interactive Literacy*. Retrieved May 28, 2017 from http://mediashift.org/2008/10/interactive-literacy005/. 2008.

[104] J. K. Rowling. *Harry Potter and the Philosopher's Stone*. 1st ed. Vol. 1. London: Bloomsbury Publishing, 1997. ISBN: 978-0747532699.

[105] Steve Rubin, Floraine Berthouzoz, Gautham J Mysore, Wilmot Li, and Maneesh Agrawala. "Content-based tools for editing audio stories." In: *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM. 2013, pp. 113–122.

[106] Steve Rubin, Floraine Berthouzoz, Gautham J Mysore, and Maneesh Agrawala. "Capture-Time Feedback for Recording Scripted Narration." In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. ACM. 2015, pp. 191–199.

[107] Yong Rui, Anoop Gupta, and Alex Acero. "Automatically extracting highlights for TV baseball programs." In: *Proceedings of the eighth ACM international conference on Multimedia*. ACM. 2000, pp. 105–115.

[108] *SMART*. https://home.smarttech.com/. Accessed: 2017-04-01. 2017.

[109] Christopher Schmandt. "The intelligent ear: A graphical interface to digital audio." In: *Proceedings, IEEE International Conference on Cybernetics and Society, IEEE*. Citeseer. 1981.

[110] *Screen Flow*. Accessed: 2016-04-11. Apr. 2016.

[111] Vikas Seth, Prerna Upadhyaya, Mushtaq Ahmad, and Vijay Moghe. "PowerPoint or chalk and talk: Perceptions of medical students versus dental students in a medical college in India." In: *Advances in medical education and practice* 1 (2010), pp. 11–16.

[112] Anoop K Sinha, Michael Shilman, and Niraj Shah. "MultiPoint: a case study of multimodal performance for building presentations." In: *CHI'01 extended abstracts on human factors in computing systems*. ACM. 2001, pp. 431–432.

[113] Lisa Stifelman, Barry Arons, and Chris Schmandt. "The audio notebook: paper and pen interaction with structured speech." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2001, pp. 182–189.

[114] Min Sun, Ali Farhadi, Ben Taskar, and Steve Seitz. "Salient montages from unconstrained videos." In: *European Conference on Computer Vision*. Springer. 2014, pp. 472–488.

[115] Anh Truong, Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. "Quickcut: An interactive tool for editing narrated video." In: *Proc. UIST*. Vol. 16. 2016.

[116] Ba Tu Truong and Svetha Venkatesh. "Video abstraction: A systematic review and classification." In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 3.1 (2007), p. 3.

[117] Matthew Turk. "Multimodal interaction: A review." In: *Pattern Recognition Letters* 36 (2014), pp. 189–195.

[118] Shingo Uchihashi, Jonathan Foote, Andreas Girgensohn, and John Boreczky. "Video manga: generating semantically meaningful video summaries." In: *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*. ACM. 1999, pp. 383–392.

[119] Tang Wang, Tao Mei, Xian-Sheng Hua, Xue-Liang Liu, and He-Qin Zhou. "Video collage: A novel presentation of video sequence." In: *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE. 2007, pp. 1479–1482.

[120] Steve Whittaker and Brian Amento. "Semantic speech editing." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2004, pp. 527–534.

[121] Steve Whittaker, Patrick Hyland, and Myrtle Wiley. "Filochat: Handwritten notes provide access to recorded conversations." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 1994, pp. 271–277.

[122] Steve Whittaker, Julia Hirschberg, John Choi, Don Hindle, Fernando Pereira, and Amit Singhal. "SCAN: Designing and evaluating user interfaces to support retrieval from speech archives." In: *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1999, pp. 26–33.

[123] *Wikipedia*. https://en.wikipedia.org/wiki/Wikipedia/. Accessed: 2017-04-1. 2017.

[124] Lynn Wilcox, Francine Chen, Don Kimber, and Vijay Balasubramanian. "Segmentation of speech using speaker identification." In: *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on.* Vol. 1. IEEE. 1994, pp. I–161.

[125] Xiao Wu, Chong-Wah Ngo, and Qing Li. "Threading and autodocumenting news videos: a promising solution to rapidly browse news topics." In: *IEEE Signal Processing Magazine* 23.2 (2006), pp. 59–68.

[126] Jun Xie, Aaron Hertzmann, Wilmot Li, and Holger Winnemöller. "PortraitSketch: Face sketching assistance for novices." In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM. 2014, pp. 407–417.

[127] Jun Xing, Rubaiat Habib Kazi, Tovi Grossman, Li-Yi Wei, Jos Stam, and George Fitzmaurice. "Energy-Brushes: Interactive Tools for Illustrating Stylized Elemental Dynamics." In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM. 2016, pp. 755–766.

[128] Dongwook Yoon, Nicholas Chen, François Guimbretière, and Abigail Sellen. "RichReview: blending ink, speech, and gesture to support collaborative document review." In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM. 2014, pp. 481–490.

[129] Bin Zhao and Eric P. Xing. "Quasi Real-Time Summarization for Consumer Videos." In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[130] C Lawrence Zitnick. "Handwriting beautification using token means." In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 53.