Synthesis-Assisted Video Prototyping From a Document

Peggy Chi
Google Research
Mountain View, CA, USA
doc2video@google.com

Tao Dong
Google
Mountain View, CA, USA
doc2video@google.com

Christian Frueh, Brian Colonna, Vivek Kwatra
Google Research
Mountain View, CA, USA
doc2video@google.com

Irfan Essa
Google Research, Georgia Institute of Technology
Atlanta, GA, USA
doc2video@google.com

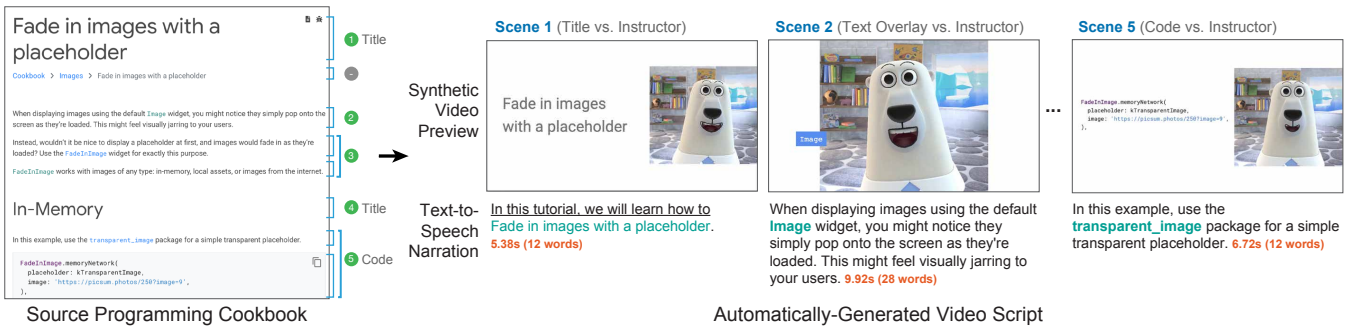

Figure 1: Doc2Video automatically converts a structural document into a video script of a series of scenes for interactive editing. For each scene, Doc2Video synthesizes a talking head video from text and composes a graphical layout showing the visual supports. Users can edit text, move elements, and replay the video preview in our Editing UI. *Source document: "Fade in images with a placeholder" by Flutter is licensed under CC BY 4.0.*

ABSTRACT

Video productions commonly start with a script, especially for talking head videos that feature a speaker narrating to the camera. When the source materials come from a written document – such as a web tutorial, it takes iterations to refine content from a text article to a spoken dialogue, while considering visual compositions in each scene. We propose Doc2Video, a video prototyping approach that converts a document to interactive scripting with a preview of synthetic talking head videos. Our pipeline decomposes a source document into a series of scenes, each automatically creating a synthesized video of a virtual instructor. Designed for a specific domain – programming cookbooks, we apply visual elements from the source document, such as a keyword, a code snippet or a screenshot, in suitable layouts. Users edit narration sentences, break or combine sections, and modify visuals to prototype a video in our Editing UI. We evaluated our pipeline with public programming cookbooks. Feedback from professional creators shows that our method provided a reasonable starting point to engage them in interactive scripting for a narrated instructional video.

UIST '22, October 29-November 2, 2022, Bend, OR, USA

ACM ISBN 978-1-4503-9320-1/22/10.
https://doi.org/10.1145/3526113.3545676

CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**.

KEYWORDS

Video creation; video prototyping; talking head videos; programming cookbooks; tutorials; voiceover; creativity tools.

ACM Reference Format:
Peggy Chi, Tao Dong, Christian Frueh, Brian Colonna, Vivek Kwatra, and Irfan Essa. 2022. Synthesis-Assisted Video Prototyping From a Document. In *The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22), October 29-November 2, 2022, Bend, OR, USA.* ACM, New York, NY, USA, **??** pages. https://doi.org/10.1145/3526113.3545676

1 INTRODUCTION

The Internet contains a vast amount of learning resources, from text documents, mixed-media articles, to videos [18, 42]. Each format supports learners differently: A document is easy to scan given its visual hierarchy, whereas a video is effective for presenting actions [11, 41] with auditory support [42]. While these tutorial presentations could reach audiences of various learning preferences, instructors may not necessarily have sufficient time and expertise in production for all formats and may choose one or the other. Prior research has shown methods to enhance existing tutorials, including video segmentation for interactive learning [41, 43, 47],

enabling voice-based interfaces [6], and video creation from step-by-step tutorials [9]. However, for articles that have a limited amount of visual assets, such as a text-based programming tutorial, it can be challenging to create an engaging instructional video.

Recent studies suggest that videos showing an instructor may better engage viewers through dialogues and first-hand demonstrations [8, 18, 42]. To produce video tutorials with a talking head view of the instructor, it is typical to start from composing a script with scene breakdowns: Each scene consists of *narrations or dialogues* that the instructor read out in the recording phase and *visual components* to be shown in the video [2, 31]. Given a text-based tutorial, scripting requires the creator copying content from the document, breaking into scenes, adding visuals, and iteratively refining the dialogue while considering the timing of each segment.

We propose a video prototyping approach that automatically converts a document to a script with a preview of synthetic talking head videos. To demonstrate the feasibility of automation, we focus on programming tutorials, a popular domain in online learning. We developed Doc2Video, an end-to-end pipeline that decomposes a source tutorial into as a series of rendered scenes. Extracted from the input document, each scene contains *narrations*, which are used to create a synthesized video of a virtual instructor, and *visual elements*, such as a text title and a code snippet, in a suitable layout. The synthesized videos are a preview to help creators observe the instructional content both visually and auditorily. Creators can edit sentences, break or combine scenes, and highlight keywords via our Editing UI to prototype a video, which encourages them to focus on the spatial and temporal composition. For the final production, creators can possibly follow the edited script to record a footage and replace the synthetic speaker preview.

We evaluated Doc2Video with 65 open-sourced programming cookbooks, which are tutorials in shorter lengths, each on a specific programming concept. Doc2Video generated a script for each cookbook, resulting in a total of 1,706 scenes that contain 1,301 talking head videos, 1,775 narration sentences, 488 code blocks, and 565 text highlights. We conducted an informal study with six professional creators to understand their existing practices and experiment our results. Feedback suggested that Doc2Video effectively jump started the creation process by providing a rough cut and previews with editing supports. Specifically, our work makes the following contributions:

- An automatic video prototyping approach supported by synthetic speaker videos based on a document input.
- Computational techniques to convert a structural document to a series of scenes, each showing a key message and a synthesized talking head video in a graphical layout.
- A set of editing methods for document-based video prototyping.
- An evaluation of automatically created videos from public cookbooks in an Editing UI reviewed by professional creators.

2 RELATED WORK

Bootstrapping Video Production. Video production involves multiple stages, from planning, scripting, recording, to editing. We focus on scripting and editing automation: Recent work enables content generation from text sentences using stock footage [28], app execution [48], and synthetic media of scenes [44] or faces [15, 37, 38]. These approaches allow users to focus on scripting or writing. For documents with a hierarchy and multimedia assets, there are methods to organize materials for video composition [9, 10, 24]. Live performances also drive video synthesis that go beyond text and a structural input [21, 36, 45]. Computer Vision techniques have automated video editing for specific domains. By analyzing dialogues and subjects, tools can place cuts [3, 27] or create a storyline [4, 20, 40]. When the footage contains a dynamic environment, human editors can guide a system on editing by providing abstract labels for physical activities [12, 39] or audio stories [35]. Our work shares the same vision to bootstrap video creation by making automatic edits from user input – a document. We adopt state-of-the-art

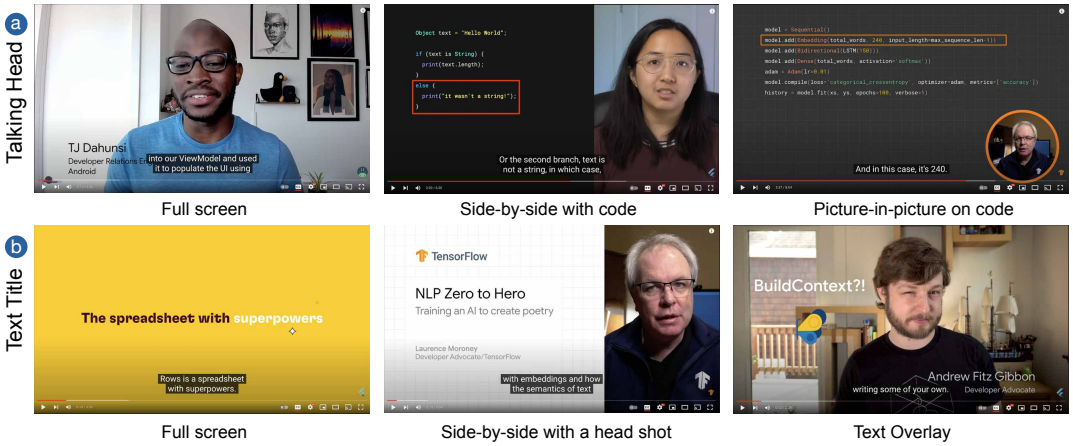

Figure 2: Example placements of a talking head video and text titles in programming tutorials. *Sources: (a) "Paging: Displaying data and its loading state - MAD Skills" by Android Developers; "Type Promotion | Decoding Flutter" by Flutter; "Training an AI to create poetry (NLP Zero to Hero - Part 6)" by TensorFlow; (b) "Creating the future of spreadsheets with Flutter" by Flutter;* "Training an AI to create poetry (NLP Zero to Hero - Part 6)" by TensorFlow; "BuildContext?! | Decoding Flutter" by Flutter.

synthesis techniques but focus on script creation, where each scene is composed differently based on the source content.

Document Editing and Conversion. Editing a document is an essential activity, especially for knowledge workers. Recent research has proposed methods to enhance editing workflows for professionals, including patent documents [13] and scientific writing [16]. To enable readers to consume document content in a non-static format, researchers introduce automatic methods that present a document as a voiceover [1, 19], a slideshow [7, 28, 46, 49], or an edited video [9, 10, 24]. We build our work on top of prior art to organize hierarchical information in a document. We further include talking head videos for the presentation, which is a popular format in education. The conversion output is used to support the video prototyping process with editing capabilities.

Video Navigation as Documents. We are inspired by prior work that presents video content in a document or table format, such as movie scenes [33], dialogue conversations [30], and tutorials for software applications [5, 32, 47] or physical tasks [41, 43]. By organizing continuous content in a hierarchical view, viewers can take advantage of the visualization for viewing and editing. Similarly, we present our generated video scenes in a script by sections that are aligned with the source document. Creators can review and edit individual components, while maintaining the overall video structure for navigation and comparison.

3 WHAT GOES TO A PROGRAMMING VIDEO?

To design an automatic approach that converts a structural document to a video script with talking head videos, we focus on programming tutorials as a proof of concept given that the structure and the average length of a cookbook are suitable for conversion to a relatively short video, and the input document includes consistent annotations of key knowledge that viewers intend to learn, such as function names or code snippets.

Nowadays, programming learners have easy access to online *web tutorials*, which include descriptions and code samples, and standalone *videos* that are commonly led by an instructor to walk through a concept, often with code snippets or a coding demonstration. Researchers have introduced novel methods that enhance viewing experiences of programming videos [25]. However, it remains critical for instructors to create quality educational videos. Studies have suggested several video production practices to engage learners, including showing an instructor's talking head view with a faster pace of dialogues [8, 18, 29]. These require scripting and editing expertise in video production.

To gain insight from existing editing practices of programming video tutorials, we analyzed 40 public videos from eight popular YouTube channels for different programming platforms (including Android, TensorFlow, Flutter, and others.) We selected videos with over 5,000 views that matched the following criteria: (1) focused on a specific topic of programming concepts, (2) showed an instructor's talking head and some amount of code, (3) had a total duration under 10 minutes, and (4) contained a certain degree of edits and scene changes. We filtered videos of uncut live coding or long lectures. We recruited four paid raters to annotate video frames with *head shots, code, IDE, screens, text titles, and animation*. The average length of the annotated videos was 4 minute and 52 seconds long.

We observed patterns based on the labels and identified common content structures and visual presentations. A video typically starts from a *title* and a transition to the *talking head shot* of an instructor, who gives an introduction on what the audience will learn from the tutorial. The video then quickly moves to instructional content, supported by visual materials such as code snippets with highlights, diagrams, screens, and animations. Throughout the video, instructor's head shots often switch between full screen, side-by-side (commonly taking 1/3 of the video frame), or a picture-in-picture view (commonly overlaid on a long code snippet, an IDE, or a full screencast) as shown in Figure 2a. Text titles and overlays are used to highlight a keyword or a concept, as shown in Figure 2b. All videos end with an *outro*, pointing the audience to code samples (e.g., GitHub links), documentations, or other video series.

These editing practices inspired us to extract critical components from a source document to an instructional video, such as titles, code snippets, and screenshots. Our goal is to combine these document elements with a talking head video presented in graphical layouts with scene changes. Since the tutorial content greatly varies, we should present the previews of rough cut decisions for creators to iterate toward the final video script.

4 VIDEO PROTOTYPING WITH DOC2VIDEO

To support creators prototyping videos from available materials such as a programming cookbook, we introduce Doc2Video, an automatic approach that converts a structural document to interactive scripting with synthetic video previews (see Figure 3). Doc2Video decomposes a Markdown-formatted source document based on sections, including titles, paragraphs, code snippets, and screenshots (see Figure 3a). It annotates detailed text elements, such as API classes, functions, and links. It organizes the content into a script of a list of scenes by turning descriptive sentences into narrations (see Figure 3b) and visually presenting critical elements in a scene (see Figure 3c) with an estimated total duration of the output video.

Synthetic videos and duration: For each scene, Doc2Video converts each of its narration sentences into a Text-to-Speech (TTS) audio segment. It concatenates the audios to synthesize a lip-sync talking head video of a virtual instructor. For example, in the first scene, Doc2Video adds a sentence, "*In this tutorial, we will learn how to*", to the cookbook's title, "*Fade in images with a placeholder*", as a total of 5.38 seconds in duration.

Video composition: In the preview, Doc2Video presents the automatically composed scenes. For example, the first scene in Figure 3c shows the tutorial title and the synthetic speaker view side-by-side, where the virtual instructor reads the narration with a continuous head motion and the lip synced. This enables users to preview how a final video would look and sound like with time estimation. The script view highlights text components that are visually seen in the video in yellow (see the first scene in Figure 3b for example.) This aims to help an instructor emphasize a keyword with a sense of timing, when they later follow this script and record a talking head video footage of themselves.

Narration and graphics editing: Users can modify both audio (narration) and visuals (text, images, layouts) in our UI. Removing or editing a sentence triggers a change in both duration and video playback (see Figure 3-1). Users can also remove a text highlight

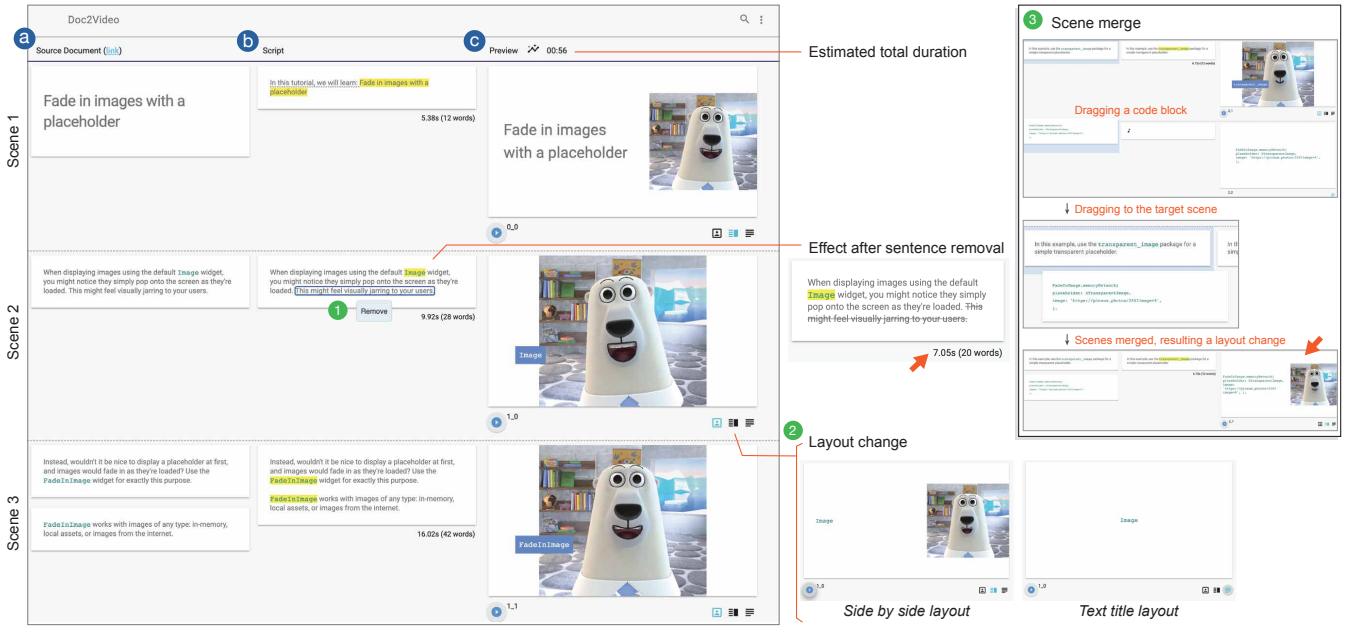

Figure 3: Doc2Video presents the automatically-generated script in an Editing UI. Each row represents a scene, which includes (a) source components from the input document, (b) narrations and estimated duration from a Text-to-Speech engine, and (c) a video preview that shows visual elements and a synthesized talking head video in a graphical layout. Users can modify the script by (1) editing or removing a narration sentence or a text highlight, (2) changing the layout, and (3) merging or removing scenes. *Source document: "Fade in images with a placeholder" by Flutter is licensed under CC BY 4.0.*

or switch between the layout options that Doc2Video pre-selects, from a full-screen text, a full-screen head shot (optionally with text overlay), to side-by-side presentations (see Figure 3-2).

Scene combination, breakdown, or removal: Doc2Video visualizes the source document, narration, and preview in a multi-column presentation, where each row shows a scene. This helps users see the correspondence and focus on one scene at a time. Doc2Video automatically combines consecutive components of the same topic into a scene, e.g., a code snippet with a screenshot. Users can combine or separate scenes by dragging and dropping elements, such as replacing a text highlight with a relevant code snippet (see Figure 3-3). For content that is not ideal for a video (e.g., long sample code), user can hide the rows.

5 AUTOMATIC SCRIPTING AND RENDERING

To provide an automatically generated video script from a document for video creation, Doc2Video consists of four main components (see Figure 4): (1) a *Document Parser* that acquires the doc structure and content annotations, (2) a *Script Planner* that converts a document to a list of video scenes with both narrations and graphical layouts, (3) a *Video Synthesizer* that renders lip-sync talking head videos, and (4) an *Interactive UI* that renders previews and supports editing. Below we discuss the detailed techniques.

5.1 Document Parser

Given a source document formatted in the Markdown language [17], Doc2Video parses and extracts the content into a hierarchical tree. The first level of the tree captures the major sections, where the underlying nodes represent detailed components, such as a title (e.g., `## In-Memory`), a paragraph, a code block (e.g., lines wrapped by ```), and a screenshot (e.g., an image node to `images.gif`). Our parser annotates links and names to API classes or functions in each sentence. For example, the word "*FadeInImage*" in "`` `FadeInImage` works with images of (...) `` " refers to an API function and is labeled.

5.2 Script Planner

Doc2Video's Script Planner converts the annotated hierarchical document into a video script, which consists of a linear list of video scenes. Each scene contains both or one of *text narration* and *visual components*. To support input documents of various lengths and levels, we represent the content as a graph and refine through the following process (see Figure 5).

Scene creation: For each document node, such as a title, a paragraph, or a code block, Doc2Video creates a scene that contains the node content. It turns a text title or a paragraph into narration N, and a code block or an image to its visual components V. Either N or V can be empty. The scenes are organized by top-level sections based on the source document.

Narration refinement: Considering that text in a written document is typically in a non-dialogue format, prior work suggested removing supplementary content and adding transition words for concise narration [9]. Doc2Video processes all narration sentences in N by removing parentheses and itemized lists (which can be added back by creators via our Editing UI.) It enhances targeted

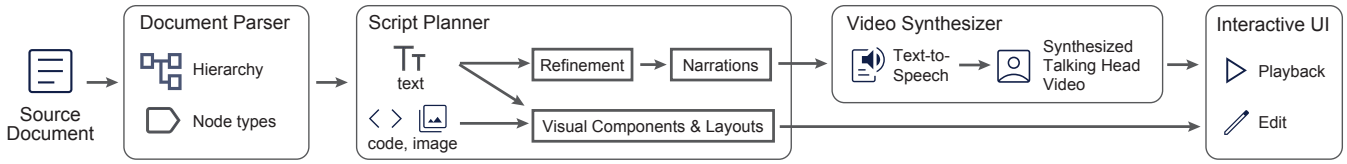

Figure 4: Doc2Video's pipeline consists of a *Document Parser* that retrieves the input document hierarchy and annotates the nodes, a *Script Planner* that converts the document into scenes with narrations and visual components, a *Video Synthesizer* that generates Text-to-Speech audios and talking head videos from narrations, and an *Interactive UI* for live composition, playback, and editing.

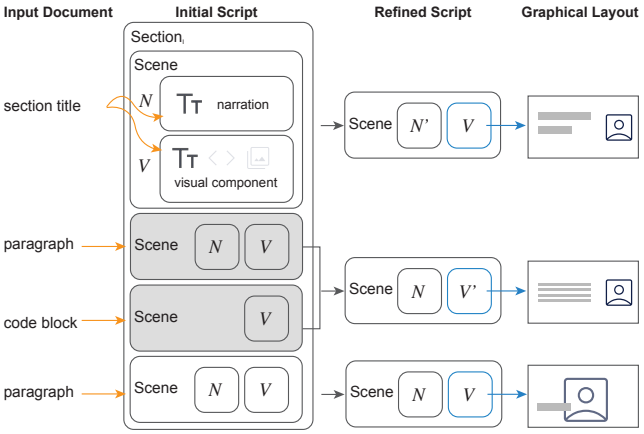

Figure 5: Doc2Video's Script Planner creates scenes based on the document content, where text serves as narrations and key assets are visual components. It combines consecutive scenes in a section that address the same concept and applies a suitable graphical layout.

elements using text templates, including titles of the document and sections. For example, an introductory sentence "*In this tutorial, we will learn how to*" is added to the tutorial title to provide viewers context as an opening. This results a refined narration list N'.

Text overlay: It is common that creators visually overlay or highlight keywords that match their voiceover to emphasize the key message (see Figure 2b right for example.) Doc2Video identifies API names in N' as keywords and adds to the visual component list as V', sorted by the ordering appeared in N'. Take the last example in Section 5.1, `FadeInImage` will therefore be visually highlighted.

Scene combination: To make a video concise and avoid long narration pauses, Doc2Video iterates through the scene list to condense the content. It combines consecutive scenes that have the same highlighted elements in V' (e.g., two paragraph sections describing the same API concept) or lack of narration where $N' = \{\}$.

Layout placement: Finally, for each scene, the planner preselects a suitable graphical layout to compose the visual components in V' with a talking head video if narration is available ($N' \neq \{\}$). For example, for $V' = \{title\}$ and $N' \neq \{\}$, it selects a side-by-side layout to place a text title next to the head shot. For another example where $V' = \{highlight\}$ and $N' \neq \{\}$, it selects the full head shot layout to overlay the keyword on the talking head video.

5.3 Talking Head Video Synthesizer

For a scene that has narrations of unrestricted length, Doc2Video renders a lip-sync talking head video. We rely on recent techniques that animate 3D talking faces from audio, named LipSync3D [26]. Given an audio input, such as a human-narrated voiceover recording or a TTS output from text, LipSync3D extracts its audio speech signals, generates and inserts a textured 3D face mesh to a target subject. We are interested to generate videos of both animated and photorealistic characters that LipSync3D supports. For an animated subject, we predefined a continuous path of the head pose that starts at the video center, slightly moves around, and backs to the center at the end of the audio input. For a photorealistic subject, we pre-processed available video footage of the subject to select a small set of video segments at various lengths. We specifically chose segments where the subject had limited facial expression and head movements for our instructional context.

For each narration sentence of N' in a scene, we generate a TTS audio file from the text and obtain its duration. The information will be used to label the exact start and end times of the sentence in the output video. Then, we concatenate audio segments of all sentences and attach to a base video (trimmed to the end of the audio length) to perform the lip-sync pipeline. We then present the lip adjusted video of a scene in the UI.

5.4 Interactive Composition and Editing

Based on the created script, Doc2Video composes a video preview live in the UI for interactive playback and editing. This enables users to see how a graphical layout changes the visuals upon selection, where our UI places the elements dynamically. For another example, when users remove a sentence, our UI skips the corresponding video segment for playback. When users drag to combine document nodes (e.g., to replace a text overlay with a code block), Doc2Video attaches the nodes to the target scene and re-selects a layout (e.g., from a full head shot to a side-by-side view.)

5.5 Implementation

We implemented the back-end offline pipeline in C++. Our Document Parser is similar to open-sourced tools [23] that output the document tree hierarchy and sentence breakdowns with labels (e.g., code blocks, links, and images.) We developed the Script Planner and implemented the talking head synthesis pipeline based on recent research [26]. We synthesize narrations using Google's Text-to-Speech API [22] and save audio output as MP3 files, which are used to create lip-sync videos in the MP4 format. We trained

the synthesis model on a set of characters, including animated characters (such as the polar bear shown in the figures in this paper) and real-world video footage of two speakers under permission.

The front-end Web UI composes text elements (e.g., a code snippet or a title), the output MP4 videos, and images (e.g., URLs to JPEGs or GIFs from the source document) in real-time. We position elements in pre-defined graphical layouts of 16:9 aspect ratio via Cascading Style Sheets (CSS). We develop the playback control engine in JavaScript to allow interactive editing, anchoring videos, and re-rendering video previews with layouts. For narration editing that goes beyond a sentence removal, our current implementation does not support real-time video rendering, although it's feasible for animated characters. Instead, we present a change in word count and estimated duration. We leave this specific real-time feature for future work.

6 RESULTS

To examine the quality of the created scripts and synthesized video previews, we performed Doc2Video's end-to-end pipeline on 65 open-sourced programming cookbooks from an open source repository [14]. The cookbooks cover 15 topics written by more than 10 authors since year 2018 based on the commit history. Our analysis shows that on average, a cookbook contains 395.5 words ([min, max]=[65, 1,003]) of 27.3 sentences ([min, max]=[7, 59]), excluding code blocks. Each cookbook contains 7.5 code blocks ([min, max]=[3, 17]) and 20.6 occurrences to API references ([min, max]=[1, 62]). Eight cookbooks show one to two GIF images that demonstrate the execution results, such as a fade-in effect. The repository does not have or link to videos.

We were able to generate both output scripts and TTS-added synthesized lip-sync talking head videos of an animated character for all cookbooks. Table 1 shows the output analysis. Our pipeline generated a total of 1,706 scenes that contained 1,301 videos (of 720 by 650 pixels), 488 code blocks, 410 text titles, and 565 keywords. On average, a cookbook has 26.25 scenes supported by 20 talking head videos, composed as a 162-second output video.

Figure 6 shows example outputs from our pipeline. We observed that Doc2Video was able to effectively break down a document into scenes. The graphical layouts make it easy to visually differentiate the content. Take Figure 6a for example. Step 1's title in Scene 3 has a larger font size, followed by the text highlight of the `Scaffold` concept in Scene 4 and a code snippet next to the head shot in Scene 5. Step 2 has a similar structure with visual consistency. Ideally, the line of code changed in Scene 7 (compared with Scene 5) should be highlighted to better guide viewers through the progression, which is beyond our current scope.

We found that TTS narration effectively illustrates the content. The animated lip-sync talking head videos appear engaging. The moving head pose, the motion of eye blinks, and the lip synced to the voiceover all support the preview (see Figure 6c). To develop synthetic characters responsibly, we suggest that it is important to avoid reinforcing unfair biases or harmful stereotypes.

We believe that editing is critical for prototyping. The source cookbook of Figure 6b shows a GIF screenshot of the scrolling parallel effect. By dragging the component into the title scene and playing back the voiceover that is unchanged, creators can immediately preview and compare the presentations. We observed that the formatting and writings styles from this repository had a certain degree of variety. Some cookbooks are relatively short with 10 nodes, while some are long with code progression. A complete code sample of 50 lines may not be suitable to be shown in a video. It's essential to allow users to edit, refine, and iterate.

7 USER EVALUATION

We conducted an informal user study to understand how the scripts and videos generated by Doc2Video could support the video creation process. In addition, we were interested in learning existing workflows and whether Doc2Video met creators' expectations.

7.1 Study Design

We invited six professional creators via internal listservs of over 100 recipients in our organization. The participants, including one female and five males, were specialized in Developer Education for three different public programming platforms. They had each created 5.6 documents and 22 YouTube videos on average in the past three months prior to the study. Their average work experiences in developer programs and video production was 9.5 years, ranged from 1.5 to 17 years. We did not record age information.

We selected three programming cookbooks (each reviewed by two participants) that contained similar compositions with 3-4 headings, 10-15 subsections, 2-3 code blocks, and 1-2 images from our dataset. Each study session began by asking the participant to review one source cookbook served on the original host site. We asked participants to verbally describe their common workflow if they were to create a video tutorial based on that cookbook.

Next, we presented our Editing UI that showed the generated script from the cookbook participants read. We walked through the automatically-generated results, video playback, and editing capabilities (which included sentence and highlight removal, text editing, and scene combination.) We then had participants open the same UI to review the output and experiment the features we introduced. Finally, we discussed their editing decisions and asked questions regarding the output quality, editing capabilities, and overall feedback. Each participant received a gift card or donation credit for their participation in a 60-minute remote session.

7.2 Understanding Existing Workflows

All participants had similar practices: They would start with a script in Google Docs with a table of two columns, one for the narration, the other for the video presentation, which would specify titles, code, screenshots, animations and the rough layouts. They would copy and paste content from a document or the code editor into the table, edit and highlight, and annotate timing. One participant shared three example scripts for references. Each included a list of two-column tables, separated by sections of different titles. Each row represented one scene, such as a code snippet (may include code highlights with time labels), a screenshot, or a diagram.

Participants had a common structure of a video, which began with an introduction (that included their name, the problem, and what the audience would learn from the video), main content with code snippets or a demo, and an outro (to point out the links in

Table 1: Output analysis of our pipeline on a test dataset of 65 open-sourced programming cookbooks.

	Source			Narration						Visual Components				Layout			Synthetic Video		
	# chapter	# scene (merged)	# scene (before)	# sentence	word count: total	word count: shortest sentence	word count: longest sentence	# API words	word per second	# titles	# code blocks	# highlights	# images	head shot	side-by-side	text only	total duration (second)	# videos	video length (second)
ave	7.00	26.25	26.78	27.31	395.46	1.91	39.82	20.60	0.42	6.31	7.5	8.69	0.28	13.71	6.31	7.45	162.10	20.02	8.10
min	3.00	7.00	7.00	7.00	65.00	1.00	13.00	1.00	0.36	2.00	3	1.00	0.00	2.00	2.00	3	31.95	5.00	1.99
max	12.00	59.00	59.00	59.00	1,003.00	5.00	77.00	62.00	0.53	13.00	17	23.00	2.00	37.00	13.00	17	385.75	42.00	41.80
total	455	1706	1741	1775	25705	-	-	1339	27.29	410	488	565	18	891	410	484	10536.18	1301	348.57

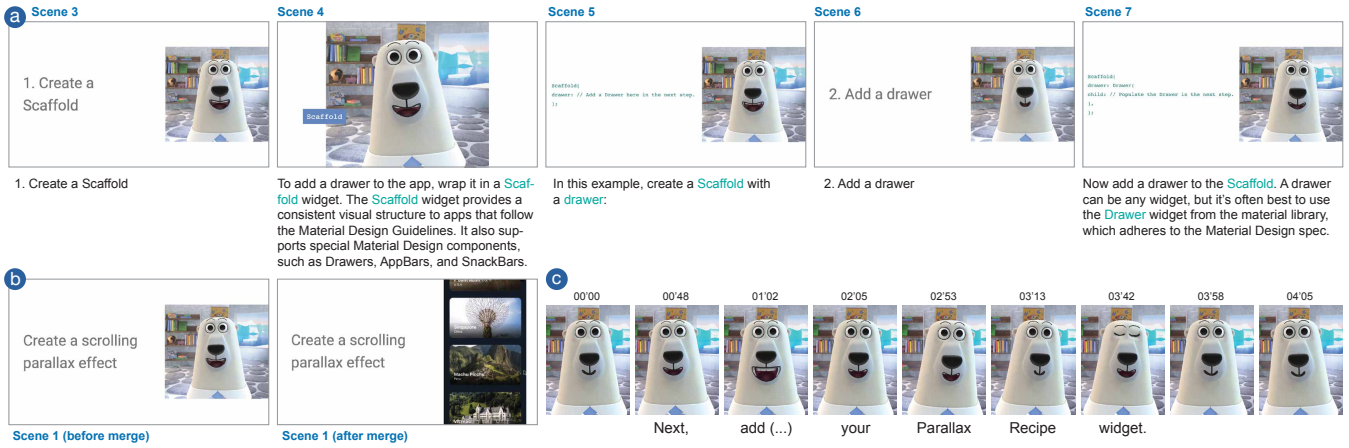

Figure 6: Sample output from Doc2Video's automatically-generated results of 65 programming cookbooks: (a) Scenes with different layouts help visualize the content structure, from a section title, an introduction, to code. (b) A head shot can be quickly replaced by an available asset via drag-and-drop that remains the original narration. (c) Our synthesis pipeline controls the head pose and lip and eye motions. We encourage readers to visit the cookbook sources for comparison: *(a) "Add a Drawer to a screen" by Flutter; (b,c) "Create a scrolling parallax effect" by Flutter.*

the video description or a video series.) All participants had the same practice to read partial code descriptively. We verified that the layouts we identified in Section 3 are commonly used.

Participants would iterate and finalize the script before recording their final talking head videos. To refine details, several mentioned that they would do dry-runs or table read [31] to verify what the dialogue would sound in a video. During the recording phase, they used a teleprompter to read the written dialogue from their script. Finally, they moved to the editing phase and would iterate the video edits on timing and animation, but rarely would re-record materials. Although our study participants were from the same organization, each team had different presentation styles for the designated developer community.

Table 2: Participant feedback on Doc2Video's results and editing features in a 5-point Likert scale.

		Med	Min	Max
Scripting	I found the pre-processed scenes useful.	4	4	5
	I found the preview useful.	4	3	5
	I found the TTS + lip-sync videos useful.	3.5	3	4
Editing	I found layout swap useful.	5	4	5
	I found text-based editing useful.	5	4	5
	I found doc element manipulation useful.	4	3	5
Overall	I found the concept of this tool promising.	4	3	5
	This tool'd be helpful for professional creators.	4	4	5
	This tool'd be helpful for amateur creators.	5	3	5

7.3 Feedback on Doc2Video

Table 2 shows the participant feedback to our questions on Doc2Video in the 5-point Likert scale from Strongly Disagree (1) to Strongly Agree (5). We do not report further statistics due to the limited scale of the study. An area of future research would be to expand the study to include more representation with self-identified participants across dimensions such as location and self-identified gender. Below we summarized our findings of our initial study.

Scene breakdown matched current workflows. Participants found the scene breakdown from a document useful (Median or M=4) and well aligned with their current practices. Rather than scripting from scratch, P1 commented that "*I like that I am starting with a default script based on an article*" and P3 liked how the tool helped "*get from zero to draft very quickly.*" As described in Section 7.2, participants commonly started from organizing the video script as a multi-column table, which Doc2Video attempted to automate. In addition, it was helpful to present source materials

next to processed scenes. P3 liked how Doc2Video "*shows me the source – it helps me understand why it made the decisions it made.*"

Video preview was helpful. Participants' feedback supported the value of previewing with pre-rendered narrated videos (M=4). As scripting needs to be as accurate as possible, the preview helped them *hear* (how the narration sounds) and *see* (how the talking head and layouts look) altogether for refinement. Several thought that a talking head preview helped them capture timing, while a placeholder of a static head shot with TTS playback would not be sufficient. P1 praised that "*I can get an accurate sense of how long the video will take*" and P2 liked the "*automatic timings*" of Doc2Video. Other participants commented that it "*Would be weird with out lipsyncing. It makes it more fun to use the tool.*" and "*I like that I don't have to use my imagination very much when reviewing the script.*" It was noted that TTS may fail in the programming context. For example, `yaml` is commonly read as "yah-mal", but TTS reads as "y-a-m-l" by letters. P6 shared that in practice, they would explicitly write the exact words to narrate for consistency.

Animated character was pleasing. We asked participants the quality of the talking head previews. All participants found it fun to see an animated character. Several were surprised by the natural tone and movements of the rendered videos. In terms of using a virtual character or a photorealistic footage, P2 commented that similar to showing wireframes in UI design, a 3D character helped him focus on the content and ignore any artifacts. P3 suggested that the polar bear made it easy to project himself as the subject.

Editing was essential. All participants addressed the importance of editing (M=5). P3 praised that "*I still had a lot of control over the final text and video*" and P1 "*like that I can make edits.*" We observed several editing aspects that participants found useful in our tool: (1) ***Removal***: Participants commonly removed phrases or sentences that did not fit the flow. They all wanted to remove sections that were not suitable for videos, especially full code samples. (2) ***Combine or highlight***: To make a video concise, participants would combine a screenshot with a code snippet or highlight partial code. Several pointed out that it is rare to present a full code snippet alone without animation or narration. (3) ***Layout change***: Participants found the instant layout preview useful (M=5). P6 commented that, "*My written scripts often include instructions like "When I say X, switch to a side-by-side view with <code> and then when I say Y switch back to full screen speaker." This tool could very much help in simplifying that communication.*"

Participants also commented additional editing capabilities to be considered in a future tool: ***Rephrase or tone change***: All participants addressed that the tone in a written tutorial might not always fit a video. P2, who immediately changed all questions to a direct tone upon reviewing the results, explained that he preferred to quickly get into the point in a video, but questions would take away the viewers. P4 and P6 preferred to make the narration more conversational to entertain the audience. ***Visualization***: To make an instructional video clear, it is important to polish the presentation format and video pacing for better content following and accessibility support in an iterative process. These suggestions open up for opportunities of developing advanced document content understanding and transformation techniques for video prototyping.

8 DISCUSSION AND OPPORTUNITIES

Overall, we found the feedback from professional creators encouraging. The concept of Doc2Video was promising (M=4), and the tool could be helpful for professional (M=4) and amateurs (M=5). Participants also suggested opportunities derived from this work:

Context-aware adjustment and synthesis: As participants showed preferences in tone (to be more dialogue or direct) and visuals (to combine materials or highlight code changes), we suggested that future work could include language models and content understanding for automatic content adjustments. A tool could highlight patterns (e.g., interrogative clauses) or suggest completion (e.g., add links to code examples or relevant screenshots.) In addition, understanding context could possibly improve synthesis techniques. For examples, TTS could emphasize keywords such as "id" and "where" in the programming context, and a synthetic character could change facial expressions with an excitement when a code change creates a surprising effect. It is important to develop synthetic characters responsibly, and to be aware of cultural contexts.

From scripting to video production: To support real-world production, participants expected to have fine-grained controls. Participants suggested that Doc2Video could further support the recording phase in a video production process and communications between stakeholders. They would want to see prompting or leave comments in the tool, which well aligned with insights from recent research [34, 36]. Given the script, it is also possible to make automatic video edits on the talking head footage based on the transcript [27].

From videos to documents: While our work focuses on video creation from a document, two participants suggested opportunities generating documents from an edited video and its script, where they currently manually converted. This aligns with prior art that demonstrated automatic approaches for video navigation [11, 41]. We look forward to integrating videos, code examples, and scripts to generate mixed-media documents for wider audiences of various learning preferences.

Beyond programming cookbooks: Participants suggested that our synthetic creation approach could support descriptive content beyond programming tutorials. To generalize to other domains, we believe that our method potentially applies to source documents that contain a hierarchy and annotations of neutral topics, such as step-by-step instructions or advertisement, and document or video translation in multiple languages. Inspired by recent work that enhances text content with screenshots [48] or stock footage [28], we look forward to supporting various types of source documents for making information accessible to the online community.

Ethical Considerations: Our method utilizes synthetic media to empower scripting in a video production process. We acknowledge that it may introduce concerns such as the potential for deepfakes and the effects of biased subject characteristics on gender, appearances, and speaker tone. We focus on trusted content and support users who create content responsibly. We believe that tools should provide sufficient controls to synthesize content closer to users' preferred production. Finally, our findings are limited to the sample size and the recruitment strategy from internal listservs. We suggest future studies should consider long-term and diverse user feedback.

9 CONCLUSION

In this paper, we introduced a video prototyping approach that automatically converts a programming cookbook to interactive scripting supported by synthetic lip-sync talking head videos. Our pipeline, Doc2Video, generates a series of scenes based on the document structure and elements, including titles, paragraphs, code blocks, and images. In each scene, sentences are used to create a synthesized video featuring a virtual instructor. Visual elements from the source document enhance the videos shown in suitable layouts, such as a code snippet or a screenshot. We provided an Editing UI for users to edit sentences and manipulate sections to prototype a video. We evaluated our pipeline with open-sourced cookbooks and received positive feedback from professional creators.

ACKNOWLEDGMENTS

We thank all the participants in our studies for their valuable insight to move this research forward. In addition, this work has been possible thanks to the support of people including, but not limited to the following (in alphabetical order of last name): Brian Curless, Molly FitzMorris, Andrew Fitz Gibbon, Senpo Hu, Philip Parham, and David Salesin.

REFERENCES

[1] Faisal Ahmed, Yevgen Borodin, Andrii Soviak, Muhammad Islam, I.V. Ramakrishnan, and Terri Hedgpeth. 2012. Accessible Skimming: Faster Screen Reading of Web Pages. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 367–378. https://doi.org/10.1145/2380116.2380164

[2] Daniel Arijon. 1991. *Grammar of the film language*. Silman-James Press.

[3] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2012. Tools for Placing Cuts and Transitions in Interview Video. *ACM Trans. Graph.* 31, 4, Article 67 (July 2012), 8 pages. https://doi.org/10.1145/2185520.2185563

[4] Juan Casares, A. Chris Long, Brad A. Myers, Rishi Bhatnagar, Scott M. Stevens, Laura Dabbish, Dan Yocum, and Albert Corbett. 2002. Simplifying Video Editing Using Metadata. In *Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques* (London, England) *(DIS '02)*. Association for Computing Machinery, New York, NY, USA, 157–166. https://doi.org/10.1145/778712.778737

[5] Minsuk Chang, Mina Huh, and Juho Kim. 2021. RubySlippers: Supporting Content-Based Voice Navigation for How-to Videos. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 97, 14 pages. https://doi.org/10.1145/3411764.3445131

[6] Minsuk Chang, Anh Truong, Oliver Wang, Maneesh Agrawala, and Juho Kim. 2019. How to Design Voice Based Navigation for How-To Videos. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. ACM, New York, NY, USA, Article 701, 11 pages. https://doi.org/10.1145/3290605.3300931

[7] Jiajian Chen, Jun Xiao, and Yuli Gao. 2010. ISlideShow: A Content-Aware Slideshow System. In *Proceedings of the 15th International Conference on Intelligent User Interfaces* (Hong Kong, China) *(IUI '10)*. Association for Computing Machinery, New York, NY, USA, 293–296. https://doi.org/10.1145/1719970.1720014

[8] Yan Chen, Walter S. Lasecki, and Tao Dong. 2021. Towards Supporting Programming Education at Scale via Live Streaming. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW3, Article 259 (jan 2021), 19 pages. https://doi.org/10.1145/3434168

[9] Peggy Chi, Nathan Frey, Katrina Panovich, and Irfan Essa. 2021. Automatic Instructional Video Creation from a Markdown-Formatted Tutorial. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 677–690. https://doi.org/10.1145/3472749.3474778

[10] Peggy Chi, Zheng Sun, Katrina Panovich, and Irfan Essa. 2020. Automatic Video Creation From a Web Page. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 279–292. https://doi.org/10.1145/3379337.3415814

[11] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: Automatic Generation of Step-by-step Mixed Media Tutorials. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. ACM, New York, NY, USA, 93–102. https://doi.org/10.1145/2380116.2380130

[12] Pei-Yu Chi, Joyce Liu, Jason Linder, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2013. DemoCut: Generating Concise Instructional Videos for Physical Demonstrations. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology* (St. Andrews, Scotland, United Kingdom) *(UIST '13)*. Association for Computing Machinery, New York, NY, USA, 141–150. https://doi.org/10.1145/2501988.2502052

[13] Han L. Han et al. 2022. Passages: Interacting with Text Across Documents *(CHI '22)*. Association for Computing Machinery, New York, NY, USA.

[14] Flutter. 2022. *Cookbook | Flutter*. Retrieved April, 2022 from https://github.com/flutter/website/tree/main/src/cookbook

[15] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. 2019. Text-Based Editing of Talking-Head Video. *ACM Trans. Graph.* 38, 4, Article 68 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323028

[16] Camille Gobert and Michel Beaudouin-Lafon. 2022. i-LaTeX: Manipulating Transitional Representations between LaTeX Code and Generated Documents *(CHI '22)*. Association for Computing Machinery, New York, NY, USA.

[17] John Gruber. 2004. Daring fireball: Markdown. (2004). https://daringfireball.net/projects/markdown/

[18] Philip J. Guo, Juho Kim, and Rob Rubin. 2014. How Video Production Affects Student Engagement: An Empirical Study of MOOC Videos. In *Proceedings of the First ACM Conference on Learning @ Scale Conference* (Atlanta, Georgia, USA) *(L@S '14)*. Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/2556325.2566239

[19] Joshua M. Hailpern and Bernardo A. Huberman. 2014. Odin: Contextual Document Opinions on the Go. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1525–1534. https://doi.org/10.1145/2556288.2556959

[20] Bernd Huber, Hijung Valentina Shin, Bryan Russell, Oliver Wang, and Gautham J. Mysore. 2019. B-Script: Transcript-Based B-Roll Video Editing with Recommendations. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article 81, 11 pages. https://doi.org/10.1145/3290605.3300311

[21] Corneliu Ilisescu, Halil Aytac Kanaci, Matteo Romagnoli, Neill D. F. Campbell, and Gabriel J. Brostow. 2017. *Responsive Action-Based Video Synthesis*. Association for Computing Machinery, New York, NY, USA, 6569–6580. https://doi.org/10.1145/3025453.3025880

[22] Google Inc. 2022. *Text-to-Speech: Lifelike Speech Synthesis*. Retrieved April, 2022 from https://cloud.google.com/text-to-speech/

[23] Christopher Jeffrey. 2018. *Marked: A markdown parser and compiler. Built for speed*. Retrieved April, 2021 from https://github.com/markedjs/marked

[24] Murat Kalender, Mustafa Eren, Zonghuan Wu, Ozgun Cirakman, Sezer Kutluk, Gunay Gultekin, and Emin Korkmaz. 2018. Videolization: knowledge graph based automated video generation from web content. *Multimedia Tools and Applications* 77 (12 2018). https://doi.org/10.1007/s11042-016-4275-4

[25] Kandarp Khandwala and Philip J. Guo. 2018. Codemotion: Expanding the Design Space of Learner Interactions with Computer Programming Tutorial Videos. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale* (London, United Kingdom) *(L@S '18)*. Association for Computing Machinery, New York, NY, USA, Article 57, 10 pages. https://doi.org/10.1145/3231644.3231652

[26] Avisek Lahiri, Vivek Kwatra, Christian Frueh, John Lewis, and Chris Bregler. 2021. LipSync3D: Data-Efficient Learning of Personalized 3D Talking Faces from Video using Pose and Lighting Normalization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 2754–2763. https://doi.org/10.1109/CVPR46437.2021.00278

[27] Mackenzie Leake, Abe Davis, Anh Truong, and Maneesh Agrawala. 2017. Computational Video Editing for Dialogue-Driven Scenes. *ACM Trans. Graph.* 36, 4, Article 130 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073653

[28] Mackenzie Leake, Hijung Valentina Shin, Joy O. Kim, and Maneesh Agrawala. 2020. Generating Audio-Visual Slideshows from Text Articles Using Word Concreteness. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3313831.3376519

[29] Bridjet Lee and Kasia Muldner. 2020. *Instructional Video Design: Investigating the Impact of Monologue- and Dialogue-Style Presentations*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376845

[30] Daniel Li, Thomas Chen, Albert Tung, and Lydia B Chilton. 2021. Hierarchical Summarization for Longform Spoken Dialog. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 582–597. https://doi.org/10.1145/3472749.3474771

[31] MasterClass. 2022. *What Is a Table Read? How to Set Up a Table Read, Including Who to Invite and What to Provide*. Retrieved April, 2022 from https://www.masterclass.com/articles/what-is-a-table-read-how-to-set-up-a-table-

read-including-who-to-invite-and-what-to-provide#what-is-a-table-read

[32] Alok Mysore and Philip J. Guo. 2017. Torta: Generating Mixed-Media GUI and Command-Line App Tutorials Using Operating-System-Wide Activity Tracing. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 703–714. https://doi.org/10.1145/3126594.3126628

[33] Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. SceneSkim: Searching and Browsing Movies Using Synchronized Captions, Scripts and Plot Summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software amp; Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 181–190. https://doi.org/10.1145/2807442.2807502

[34] Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. 2016. VidCrit: Video-Based Asynchronous Video Review. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 517–528. https://doi.org/10.1145/2984511.2984552

[35] Amy Pavel, Gabriel Reyes, and Jeffrey P. Bigham. 2020. Rescribe: Authoring and Automatically Editing Audio Descriptions. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 747–759. https://doi.org/10.1145/3379337.3415864

[36] Hariharan Subramonyam, Wilmot Li, Eytan Adar, and Mira Dontcheva. 2018. TakeToons: Script-Driven Performance Animation. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 663–674. https://doi.org/10.1145/3242587.3242618

[37] Synthesia. 2022. *Synthesia - AI Video Generation Platform.* Retrieved April, 2022 from https://www.synthesia.io/

[38] Sarah Taylor, Taehwan Kim, Yisong Yue, Moshe Mahler, James Krahe, Anastasio Garcia Rodriguez, Jessica Hodgins, and Iain Matthews. 2017. A Deep Learning Approach for Generalized Speech Animation. *ACM Trans. Graph.* 36, 4, Article 93 (July 2017), 11 pages. https://doi.org/10.1145/3072959.3073699

[39] Anh Truong, Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. 2016. QuickCut: An Interactive Tool for Editing Narrated Video. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 497–507. https://doi.org/10.1145/2984511.2984569

[40] Anh Truong, Sara Chen, Ersin Yumer, David Salesin, and Wilmot Li. 2018. Extracting Regular FOV Shots from 360 Event Footage. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, Article 316, 11 pages. https://doi.org/10.1145/3173574.3173890

[41] Anh Truong, Peggy Chi, David Salesin, Irfan Essa, and Maneesh Agrawala. 2021. Automatic Generation of Two-Level Hierarchical Tutorials from Instructional Makeup Videos. In *Proceedings of the 2021 ACM Conference on Human Factors in Computing Systems (CHI '21)*.

[42] Sylvaine Tuncer, Barry Brown, and Oskar Lindwall. 2020. On Pause: How Online Instructional Videos Are Used to Achieve Practical Tasks. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3313831.3376759

[43] Bryan Wang, Meng Yu Yang, and Tovi Grossman. 2021. Soloist: Generating Mixed-Initiative Tutorials from Existing Guitar Instructional Videos Through Audio Processing. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 98, 14 pages. https://doi.org/10.1145/3411764.3445162

[44] Miao Wang, Guo-Wei Yang, Shi-Min Hu, Shing-Tung Yau, and Ariel Shamir. 2019. Write-a-Video: Computational Video Montage from Themed Text. *ACM Trans. Graph.* 38, 6, Article 177 (Nov. 2019), 13 pages. https://doi.org/10.1145/3355089.3356520

[45] Nora S. Willett, Wilmot Li, Jovan Popovic, and Adam Finkelstein. 2017. Triggering Artwork Swaps for Live Animation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 85–95. https://doi.org/10.1145/3126594.3126596

[46] Haijun Xia, Jennifer Jacobs, and Maneesh Agrawala. 2020. Crosscast: Adding Visuals to Audio Travel Podcasts. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 735–746. https://doi.org/10.1145/3379337.3415882

[47] Saelyne Yang, Jisu Yim, Aitolkyn Baigutanova, Seoyoung Kim, Minsuk Chang, and Juho Kim. 2022. SoftVideo: Improving the Learning Experience of Software Tutorial Videos with Collective Interaction Data. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 646–660. https://doi.org/10.1145/3490099.3511106

[48] Mingyuan Zhong, Gang Li, Peggy Chi, and Yang Li. 2021. HelpViz: Automatic Generation of Contextual Visual Mobile Tutorials from Text-Based Instructions. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 1144–1153. https://doi.org/10.1145/3472749.3474812

[49] Douglas E. Zongker and David H. Salesin. 2003. On Creating Animated Presentations. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (San Diego, California) *(SCA '03)*. Eurographics Association, Goslar, DEU, 298–308.