

Designing Video-Based Interactive Instructions

Peggy Chi



Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2016-150

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-150.html>

August 15, 2016

Copyright © 2016, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Designing Video-Based Interactive Instructions

by

Pei-Yu Chi

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Björn Hartmann, Chair

Professor Eric Paulos

Professor Kyle Steinfeld

Summer 2016

Designing Video-Based Interactive Instructions

Copyright 2016
by
Pei-Yu Chi

Abstract

Designing Video-Based Interactive Instructions

by

Pei-Yu Chi

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Björn Hartmann, Chair

When attempting to accomplish unfamiliar tasks, people often look for tutorials to follow instructions. While it is easy to access online instructions shared by domain experts, navigating step-by-step guidance using existing tools remains inefficient. In addition, producing high-quality instructions that are easy to follow requires authoring expertise and a significant time investment in editing. This dissertation introduces video-based recording, editing, and playback tools optimized for creating and consuming tutorials from author demonstrations. Our interactive systems capture videos and high-level events that are important to a learner. Using video and audio analysis techniques, we develop algorithms that automatically produce high-quality instructions, which dramatically reduce the effort required for amateur creators. By introducing novel tutorial formats combined with video content, these designs in turn improve viewers' learning experience.

We present a series of authoring tools that enable amateur authors to create effective tutorials: 1) *MixT* is a system that automatically generates step-by-step mixed media tutorials from software demonstrations. 2) *DemoWiz* is a tool that provides an increased awareness of upcoming events in a software demonstration video. 3) *DemoCut* is a semi-automatic video editing tool for physical tasks. 4) *Kinectograph* is a recording device that automatically follows an instructor for filming a physical demonstration. 5) *DemoDraw* is a multi-modal system to generate step-by-step motion illustrations from author's body movements. Current authoring practices from professionals are encoded into automatic algorithms and interactive techniques. These systems are evaluated through a series of studies, which demonstrate that users can efficiently create and follow concise instructions using our tools.

To my Dad, Mom, Senpo, and Marissa

Contents

Contents	ii
List of Figures	iv
List of Tables	ix
1 Introduction	1
1.1 Challenges of Creating and Consuming Instructions	3
1.2 Thesis Contributions	8
1.3 Overview	8
1.4 Prior Publications	9
2 Background	10
2.1 Instructions: Terminology	10
2.2 Why Authoring and Consuming Instructions?	14
2.3 Techniques of Effective Instructions	14
2.4 Instruction Production Process	20
2.5 Summary	23
3 Related Work	24
3.1 Generating Instructions for Software Applications	24
3.2 Generating Instructions for Physical Activities	28
3.3 Working with Videos	33
4 MixT: Mixed Media Tutorials	36
4.1 Introduction	36
4.2 Related Work	38
4.3 Design Guidelines	39
4.4 Computer-Generated Mixed Media Tutorials	45
4.5 Evaluation	51
4.6 Conclusion	53
5 DemoWiz: Visualization for Software Demonstration	54

5.1	Introduction	54
5.2	Related Work	56
5.3	Design Guidelines	57
5.4	Computer-Generated Visualization	58
5.5	Evaluation	63
5.6	Conclusion	70
6	DemoCut: Instructional Videos from Demonstration	71
6.1	Introduction	71
6.2	Design Guidelines	73
6.3	Creating Video Tutorials with DemoCut	76
6.4	Automatic Effect Decision Pipeline	80
6.5	Evaluation	84
6.6	Conclusion	89
7	Kinectograph: Body-Tracking Camera Control	91
7.1	Introduction	91
7.2	Recording Video Tutorials with Kinectograph	93
7.3	Automatic Tracking Techniques	95
7.4	Evaluation	98
8	DemoDraw: Motion Illustrations from Demonstration	102
8.1	Introduction	102
8.2	Related Work	105
8.3	Principles and Methods	106
8.4	Creating Illustrations with DemoDraw	108
8.5	Generation Pipeline	111
8.6	Evaluation	115
8.7	Conclusion	120
9	Conclusion	122
9.1	Restatement of Contributions	122
9.2	Future Directions	123
9.3	Summary	127
	Appendices	128
	A. Materials for the MixT Formative Study	128
	B. Materials for the DemoCut Formative Study	130
	C. The Initial Design of Kinectograph	131
	D. Materials for the DemoDraw User Study and Results	132
	Bibliography	134

List of Figures

1.1	Motions arrows in visual instructions.	2
1.2	Our video-based approaches capture an author’s demonstration, analyze the captured materials, and automatically make editing decisions to produce effective instructions. Authors can review their recordings, modify the generated results, or re-perform a demonstration.	3
1.3	MixT generates step-by-step tutorials (left) that contain static and video information from task demonstrations. Videos are automatically edited and offer different views (right) to highlight the most relevant screen areas for a step. Visualizing mouse movement helps learners understand a complex action.	4
1.4	DemoWiz visualizes input events in a screencast video to help viewers anticipate the upcoming event for following a software demonstration.	5
1.5	DemoCut asks authors to mark key moments in a recorded video of demonstration using a set of marker types. Based on marker information, the system uses audio and video analysis to automatically organize the video into meaningful segments and apply appropriate video editing effects, which can be modified via a playback UI.	6
1.6	Composed of a Kinect sensor to track author movement and a motorized dock to pan and tilt the camera, Kinectograph continuously centers the author (or their hand) in the recorded video for filming physical activities.	7
1.7	DemoDraw’s multi-modal approach enables authors to capture motion, verify results, and re-perform portions to generate step-by-step motion illustrations.	7
1.8	A design space of the creation and consumption process for tutorials. It involves three phases of recording, editing, and playback in either software domain or a physical world. This dissertation proposes a series of systems that focus on various aspects in this design space.	8
2.1	Example activities in tutorial domains.	11
2.2	Major tutorial forms from online resources.	13
2.3	Color can differentiate between annotation (labels in black) and annotated information (parts in green in this diagram).	15
2.4	A 5-step static tutorial for a DIY task presented as a web document. Each step includes image(s) and text descriptions. Tutorial by David Hodson [108], licensed under CC BY 3.0.	17

2.5	Conventional video editing techniques are often seen in video tutorials, such as showing a sequence of overview and detailed shots (a) and a title scene to introduce a new section, which can include animation or movement as a preview (b). Images are obtained from the same video shown in Table 2.2.	19
2.6	Video index to a video tutorial helps viewers navigate between topics.	20
2.7	A common workflow of tutorial creation, which includes planning the task in detail, recording the process, editing the captured content into a readable form, and sharing with the communities.	21
2.8	Authors often create scripts for instructional videos. Here show examples used in food safety [217] (a) and cooking [71] (b) instructions. Each includes video shot(s) and narration, some with additional notes on the actions. High-level structure can also be specified, such as “introduction” and “conclusion.”	22
3.1	Real-time visual enhancements to GUI applications are commonly used in instructional videos. Mouseposé highlights a mouse cursor (a, left) and displays keyboard input (a, right). Prefab [62] creates effects such as target-agnostic afterglow [21] (b, left) and target-aware cursor [93] (b, right) by identifying and reverse engineering UI components.	25
3.2	Examples where software operations are automatically rendered on top of application screenshots, including moving the mouse, dragging, clicking, and scrolling by Nakamura and Igarashi [159] (A) and application-specific operations (a-b), parameter setting (c-f), and image manipulations (g-f) by Grabler et al. [91] (B).	26
3.3	A static tutorial automatically generated by Grabler et al.’s system [91].	27
3.4	Instructional systems that help learners compare image manipulations and similar tutorials using before and after images and event timelines by Grossman et al. [96] (a) and side-by-side documents by Kong et al. [126] (b).	27
3.5	Instructional diagrams can be automatically generated with a model-based approach, such as assembly instructions by Agrawala et al. [5] (a) and causal chain sequences of mechanical interaction by Mitra et al. [154] (b).	30
3.6	TeleAdvisor [99] provides an authoring interface (left) for an instructor to guide a remote worker through a repair task (right).	32
3.7	Work by Mohr et al. [155] automatically analyzes a technical document and augments a machine with AR animations in 3D to help novices operate an unfamiliar machine.	32
4.1	MixT generates tutorials that contain static and video information from task demonstrations. Videos are automatically edited and offer different views to highlight the most relevant screen areas for a step. Visualizing mouse movement helps user understand a complex action.	37
4.2	In our formative study, participants completed three tutorials with images similar but not identical to the originals.	40
4.3	In the mixed condition, participants saw an HTML page with static images and text; they could expand each step to view a video of that step (here: step 2.5).	41
4.4	Users tied for fewer errors with mixed tutorials.	42

4.5	In two of three tasks, participants made more repeated attempts at executing steps with static tutorials than with mixed tutorials. Video tutorials had the fewest attempts. . . .	43
4.6	MixT offers three video playback options: Normal mode (A), zoom mode (B) and crop mode (C).	46
4.7	Mouse visualization distinguishes moving and dragging.	47
4.8	MixT generates tutorials from video and log files.	48
4.9	Automatically-generated MixT results.	50
5.1	DemoWiz visualizes input events in a screencast video to help presenters anticipate the upcoming event for narrating a software demonstration in a live presentation.	56
5.2	DemoWiz workflow: Presenters capture a software demonstration, edit the video recording while rehearsing with our playback UI, and present the edited video to the audience using a presenter view.	59
5.3	DemoWiz visualizes input events in a graphical way. From the left to right we show a mouse click, double-click, a drag, a mouse scroll, and keystroke events. These glyphs are overlaid on the video recordings.	60
5.4	Three types of motion arrows in DemoWiz that guide presenters to the next event of different distances at a far (A), nearly the same (B), and near location (C).	60
5.5	A progress in time guides the presenter from the current event (left) gradually to the upcoming action (right) using relative timing with a progress bar (top) and absolute timing (bottom).	61
5.6	Examples of DemoWiz visualizations with four different systems and input event sequences.	61
5.7	Participants saw the presenter view, shown on the left, while giving a presentation in the study. The audience view on the right was shown in the other display with synchronized playback.	65
5.8	User feedback from questionnaire on the 7-point Likert scale.	66
5.9	The number of times events were anticipated by the narration, co-occurred, or occurred after the fact.	67
6.1	DemoCut automatically segments a single-shot demonstration recording and applies video editing effects based on user markers (A), including subtitles, fast motion (B), leap frog, zoom (C), and skip (D).	72
6.2	We analyzed 20 DIY instructional videos. Examples included (clockwise from top left): Microcontroller circuit design, tablet screen replacement, custom shoe painting, and creating latte art.	74
6.3	DemoCut users first mark their recorded video in the Annotation Interface. DemoCut then segments their recording and suggests video edits, which users can review and change in the Editing Interface.	76
6.4	With DemoCut's Annotation UI, users add markers to their recorded video (A). Each marker can be labeled with a descriptive string (B).	77

6.5	DemoCut accelerates playback of video with intermittent audio narration through Fast Motion (A) and Leap Frogging (B).	78
6.6	DemoCut’s Editing Interface shows automatically generated segments with effect suggestions (A). Users can change the effect (B) applied to each segment (C).	79
6.7	Users can annotate a video with visual highlights using the Editing Interface, such as adding an arrow to point out an important area (A). Annotations will be rendered on the fly (B).	80
6.8	Given user markers, DemoCut analyzes both video and audio to segment the demonstration video and apply editing effects.	81
6.9	DemoCut looks for similar video frames before and after a marked frame T^m to find candidate start (T^s) and end (T^e) frames for the corresponding segment.	82
6.10	We use RMS energy of the audio to find silent and non-silent regions. We determine the threshold for silence by analyzing the histogram of the RMS energy.	82
6.11	Illustrative frames from the seven videos used to assess DemoCut. Labels correspond to task labels in Table 6.2.	85
6.12	Our user study setup.	87
7.1	Kinectograph includes a Kinect camera to track user movement and a motorized dock to pan and tilt the camera so that the user (or their hand) remains centered in the recorded video. Here the device follows the user’s hand while he is illustrating.	92
7.2	Kinectograph UI on a tablet device.	94
7.3	Kinectograph tracks and provides a digital zoom view (right) captured from the Kinect camera view (left) in real-time based on user specified area.	95
7.4	Kinectograph architecture.	95
7.5	Kinectograph tracks the position of the target (A) and computes the tilt (B) and the pan (C) angles in order to center the target. It digitally zooms the camera view based on user specified region on the tablet UI (D).	97
7.6	Examples of camera views captured by a static camera and Kinectograph at two specific moments in time.	99
8.1	Examples of manually generated human movement illustrations: (a) for sign language [56]; (b) for weight training [8]; (c) for dance steps [unknown]; (d) for a gestural interface [54].	103
8.2	DemoDraw’s authoring interfaces and results: (a) multi-modal <i>Demonstration Interface</i> to capture motion, verify results, and re-perform portions if needed; (b) conventional <i>Refinement Interface</i> for refinement and exploring other visualization styles; (c-d) examples of illustration styles (annotated with camera viewing angle θ , motion arrow offsets δ , stroboscopic overlap ratio ρ , and numbers of intermediate frames n).	104
8.3	Canonical authoring workflow consisting of a Motion Definition task then a Motion Depiction task. Design decisions associated with a task are shown in bold with design parameters in italics.	107

8.4	DemoDraw authoring UI: Using the Demonstration Interface, an author sees an avatar following her real-time movement (a). During recording (initiated by voice command “Start”), real-time feedback shows the speech labels (b). Once a recording is completed by voice command “Stop”, the motion visualization and a timeline are immediately available (c) for the author to review, and a step-by-step overview will be generated. . .	109
8.5	Using DemoDraw’s Refinement Interface, the author can refine the visuals (a) and explore more illustration effects (b, c).	110
8.6	DemoDraw system components and pipeline.	111
8.7	Illustration of motion analysis algorithm (two joints shown due to space): significant moving periods of joint movements (pink) are mapped to speech labels to define motion segments (blue). Note the right hand period is mapped to “two” because it begins shortly after the left hand period.	112
8.8	Study 2 median ratings for Q1 and Q5 by illustration step.	117
8.9	Different illustration effects conveying the same motion recording using DemoDraw’s Refinement Interface: a and c are created by the authors of this work and a was used in Study 1; b by Study 3-P1 using 4 intermediate frames with zero offset; d by Study 3-P2 using 5 frames, positioned as a sequence.	119
9.1	Online instructions often include external links (a) to other materials (b), which enhance or expand a step-by-step tutorial. Example by Jeff Suovanen [196], licensed under CC BY 3.0.	124
9.2	A recent Augmented Reality (AR) application enables reviewing character animation beyond a desktop in a room-size environment [32], licensed under CC BY 2.0. . . .	126
D.1	Tasks provided in Study 1: We showed the printouts of these two sets of 4-step motions generated by DemoDraw using both the Demonstration Interface and the Refinement Interface. We asked participants to re-perform in front of a camera.	132
D.2	Step-by-step illustrations generated by participants in Study 2 using the Demonstration Interface: 1) Results from P9 and P7 show the same four gestures of interface control in task 1, and 2) Results from P6 show 8-step moves in task 2.	132
D.3	Selected illustrations from the open-ended task created by three different participants using the Demonstration Interface in Study 2: P5 performed to conduct a 4/4 beat pattern; P8 and P10 each performed four and eight free moves.	133

List of Tables

2.1	A list of annotation techniques that are often used to provide instructions. Examples are selected from stroller instructions [34]. Reproduced with permission.	16
2.2	Annotation techniques for static tutorials (listed in Table 2.1) are often used in instructional videos, such as arrows and highlights to show product operations.	18
4.1	Participants watched videos most often for brushing, control point manipulation, and parameter adjustments.	44
4.2	Error rates for automatically generated tutorials.	52
5.1	Survey of software demonstration preferences from presenters' (N=60) and audience's (N=70) point of views.	57
6.1	Background information about interview participants. * <i>Numbers of videos published on personal YouTube channels, excluding those on the professional channels.</i>	75
6.2	A list of how-to videos we recorded to assess the robustness of the DemoCut system. .	84
6.3	Quantitative analysis of the user evaluation.	87
7.1	Task information and results collected in the preliminary user study.	99
8.1	Incorrect movements performed by participants in Study 1.	116

Acknowledgments

I would first like to thank my advisor, Björn Hartmann, for opening up research opportunities and collaborations that led me to where I am today. Björn’s full support of my interest in video, which I derived from my previous work at the MIT Media Lab, introduced me to interactive instructional design that I truly enjoy exploring. Working with Björn made five years short as ideas emerged quickly in every discussion. I thank him for his encouragement to pursue my four summer internships that led me to my full-time position. I must mention that it was fun biking together at the Google campus, where we brainstormed about which cafés to visit, and of course, research topics. I also want to thank Eric Paulos, Kyle Steinfeld, and Marti Hearst on my qualifying exam and thesis committees for their valuable and encouraging feedback, which drove my work forward as a whole.

I feel so grateful to be able to work closely with researchers from outside UC Berkeley over the last five years: soon after joining Berkeley, I began working with Mira Dontcheva and Wilmot Li at Adobe Research. This evolved into our first exciting collaboration, and eventually three fruitful projects. Their thoughtful, patient, and cheerful guidance indirectly encouraged me to join industrial research. Steven M. Drucker and Bongshin Lee at Microsoft Research sparked my interest in visualization under their mentorship. Daniel Vogel, while an interviewee at first, became a key author on one of my works, which was very fortunate for me. Finally, Yang Li at Google Research expanded my view to another research topic, programming tools for cross-device interaction. My work with Yang and Björn led to a best paper award at CHI.

This dissertation could not have been completed without the support from the many students who collaborated with me in my research: Sally Ahn, Amanda Ren, Joyce Liu, Jason Linder, Derrick Cheng, and Taeil Kwak. I wish to extend thanks to members of my research groups at the Berkeley Institute of Design, Visual Computer Lab, and EECS: Andrew Head and Amy Pavel for their continuous research support, Fu-chung Huang, Nicholas Kong, Kenghao Chang, Tsung-hsiang Chang, and Chung-wei Lin for advice as fresh PhDs, Valkyrie Savage and Shiry Ginosar for making the best and very first “b-crew”, Steven Rubin for his expertise in audio research, William McGrath for brainstorming about AR authoring tools, and Tim Campbell and Cesar Torres for being part of my “research family” studying tutorials.

I thank the university and my industry collaborators for their encouragement in support. In particular, UC Berkeley and Google have graciously supported my studies through a Fellowship for Graduate Study and a PhD Fellowship. My longtime mentors Hao Chu, Robin Chen, Jimmy Lin, and Henry Lieberman have guided me in navigating the research world.

Finally, this dissertation is dedicated to my dear family: my parents, Rock Chi and Joanna Chen, my sister, Hsin-yu Chi, my husband, Senpo Hu, and my adorable daughter, Marissa Hu. I thank them for their tremendous love and support.

Chapter 1

Introduction

When attempting to accomplish unfamiliar, complicated tasks, people often look for tutorials to follow instructions. From performing daily tasks such as cooking and operating a machine, using software applications, to physical activities like sports and dance performance, each domain involves specific “how-to” knowledge with a certain degree of complexity [184]. Instructions, which describe how a specific goal can be accomplished, are a tool for people to self-learn a task [192]. Studies have shown that visual instructions are cognitively favorable by people as they are easier to comprehend and remember than text information [101, 150, 104]. In history, pictorial instructions have been created from the Middle Age to explain dancing or weapon operations [153]. It was found that the first use of letters in technical drawing for text referral was by Italian polymath Leonardo da Vinci. In 1737, French engineer de Bélidor [23]’s diagrams were the first to apply arrows to indicate direction of movements (see Figure 1.1). From 1760, when the Industrial Revolution introduced mass production, instructions have been seen widely for various products and uses.

Since the late 1990s, the advance in computer technologies has introduced more versatile instructional design. Instructions can now be created via software tools rather than hand drawing; they can include multimedia such as images and videos in several forms; they can be accessed through the Internet, as well as in hard copy. This advancement also enables consumers or end-users to document and share their domain knowledge [130]. As of today, popular tutorial sharing sites like Instructables has over 220,000 articles [111], wikiHow provides over 192,000 articles [215], Food.com serves over 500,000 user-generated recipes with 125,000 photos [78], and YouTube hosts over 285 million How-To videos¹. The variety of topics, content, and presentation styles provides learners more options to understand domain knowledge. However, navigating a tutorial using existing tools remains inefficient for following step-by-step instructions. It can be challenging to observe details from text and images or find specific piece of information in a video through a timeline with conventional video players. On the other hand, producing high-quality instructions that are easy to follow requires authoring expertise and a significant time investment. It involves several stages to design, record, and edit multimedia materials of a task using a variety of creation tools [206, 208, 157].

¹ YouTube, <https://www.youtube.com/>, accessed June 2016

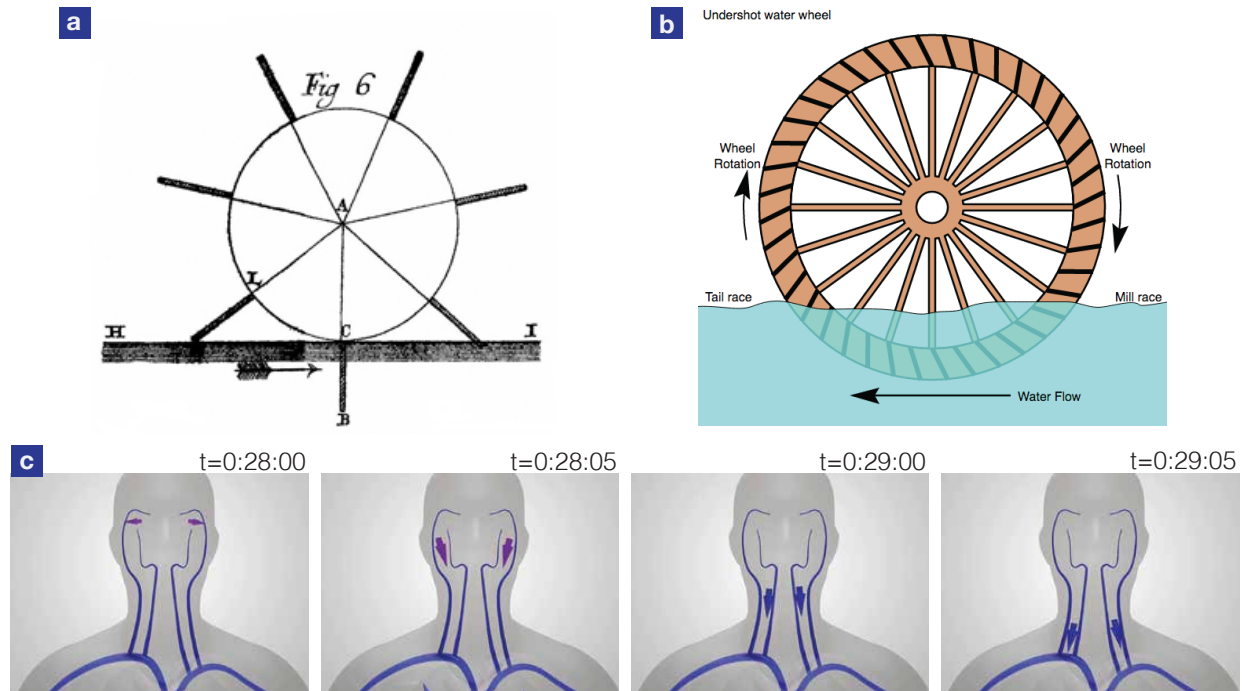


Figure 1.1: Motions arrows in visual instructions: (a) Year 1737: The first use of a motion arrow in an illustration explains the impact of water flow of a water wheel [23], (b) Year 2002: Similarly, arrows are used to explain the water flow and rotation of an undershot water wheel¹, (c) Year 2014: An animation visualizes the blood flow using motion arrows².

¹ Original artwork by Daniel M. Short, “Schematic diagram of an undershot water wheel”, licensed under CC BY-SA 2.5

² Video by Bioscience Credentials, “Blood Flow in the Human Body”, <https://youtu.be/GwX41xm9esY>, licensed under CC BY 3.0

The goal of this dissertation is to investigate interactive instructional design and develop computational tools that support the authoring process. To contribute to computational methods of authoring user-generated instructions, two research questions that this work focuses on are:

- How can authoring tools support domain experts in efficiently creating effective, high-quality instructions based on video-recorded demonstrations?
- How can new tutorial formats help authors better express their intent and help learners understand and follow the author’s instructions?

This dissertation presents video-based computational approaches that enhance tutorial creation and consumption from author demonstrations. We encode the current practices from professional authors into automatic algorithms and interactive techniques. We develop authoring tools that follow

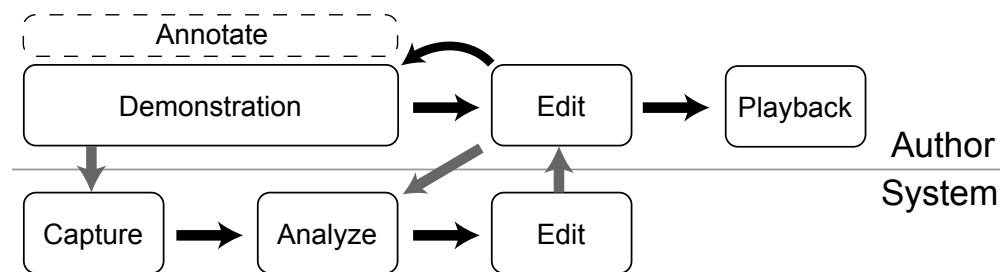


Figure 1.2: Our video-based approaches capture an author’s demonstration, analyze the captured materials, and automatically make editing decisions to produce effective instructions. Authors can review their recordings, modify the generated results, or re-perform a demonstration.

a general workflow (see Figure 1.2): An author first performs a demonstration of an instructional task. Our systems capture videos and high-level information that is important to a learner for understanding a task. Automatic analysis on the captured materials is performed during or after the performance. Based on the analysis, the systems integrate videos, author annotations, and automatic editing decisions to produce effective instructions. Authors can review the generated results, edit, or iteratively re-perform a task. Using this workflow, our work dramatically increases the quality of amateur-produced video instructions, which in turn improves learning for viewers who interactively navigate the content.

We will introduce five interactive systems that we develop to address these challenges. These tools cover both software applications (e.g., image manipulation tasks or browser navigation) and physical activities (e.g., Do-It-Yourself projects or dance movements) for recording, editing, and replaying instructional content.

1.1 Challenges of Creating and Consuming Instructions

Visual instructions are the dominant form of instructional design [153]. Cognitive load theory of multimedia learning suggests that learners process information using distinct channels, one for visual and the other for verbal formats [198, 197, 164]. It was found that learners performed better when received a pictorial summary of a scientific system than those who received the full text alone or the full text with the summary [150].

Among all the multimedia support, videos are a common form to present instructions. We suspect that the great popularity of videos is due to the following reasons: First, consumer devices and software have become affordable for authors to quickly record activities and later share via online platforms at minimum cost. Second, videos can be an efficient medium to document activities. Transferring know-how concisely and effectively to the audience is challenging. It especially requires efforts when a task involves *tacit knowledge*, which is a kind of knowledge that is difficult to articulate in a written or verbal form [173, 124]. Examples of tacit knowledge include dancing, riding a bike, or driving nails with a hammer. Dancers can perform movements fluently with music. If they are asked to focus on the composite pieces, such as the arm and foot

actions or rhythm, they might get confused and fail to express the entire movement [173]. Very often, recording a video eases the difficulties of describing the entire activities in an explicit form. This leads to another motivation that videos also provide an effective channel to convey ideas with adequate amounts of details. Learners can visually observe the exact actions in a video as if an expert were coaching in person [128].

However, while videos are easy to produce, they can include a lot of unnecessary footage. Inevitable content such as pauses, mistakes, and long repetitive actions makes it difficult for learners to focus on the most important steps and actions. A lot of authoring effort commonly goes into extracting footage, applying visual effects, and adding subtitles and annotations. In addition, even with a well-edited video, navigating using a conventional video player remains inefficient. Learners with various needs could have a hard time skimming to an interesting moment or perceiving high-level overviews. Alternatively, a pictorial summary or static step-by-step tutorials presented with text and images can effectively guide knowledgeable learners through familiar tasks.

New Tutorial Formats

Both static and video tutorials have strengths, but neither format alone is well suited for all learning needs that learners may have. To combine the benefits, we design a new instructional presentation called *MixT* (mixed-media tutorials) that improves learners' success in following instructions (see



Figure 1.3: MixT generates step-by-step tutorials (left) that contain static and video information from task demonstrations. Videos are automatically edited and offer different views (right) to highlight the most relevant screen areas for a step. Visualizing mouse movement helps learners understand a complex action.

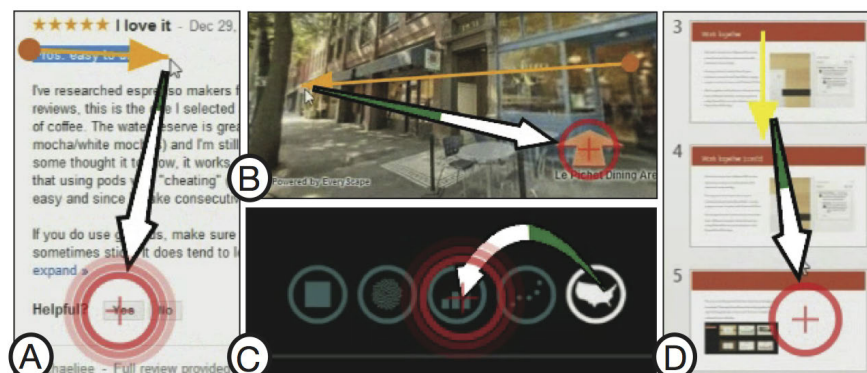


Figure 1.4: DemoWiz visualizes input events in a screencast video to help viewers anticipate the upcoming event for following a software demonstration.

Figure 1.3). MixT presents step-by-step static instructions and includes in-place video clips for each operation. With MixT, learners can quickly scan forward and backward on a web page to obtain an overview of a task. Embedded videos help them understand continuous, complex manipulation, such as brushing on a canvas and adjusting control points. MixT's playback UI allows learners to interactively control *when* to see images or videos, and *how* to render videos. Video editing techniques are applied to emphasize instructions, including cropping salient screen regions and highlighting interaction. In our within-subject experiment, MixT successfully reduced numbers of errors and attempts made by learners when following image manipulation tasks.

MixT offers a novel way of navigating instructional content with a combination of static step-by-step and embedded video presentations. If an author wants to narrate over a video recording to illustrate a demo, it can be challenging to pace oneself at the suitable timing without expecting *when* and *what* action is taking while a video is playing. We design *DemoWiz*, a system that augments a screencast video with visualizations (Figure 1.4). By logging the input events of a software demonstration, DemoWiz overlays glyphs to visually guide viewers to the next action along with the time remaining before the action occurs. This enables viewers to anticipate the video content rather than react to it. Our study showed that fewer anticipation errors and narration delays were made with DemoWiz.

Tutorial Generation from Software Demonstration

While new tutorial formats are shown to be useful, manually creating instructions can be extremely time- and effort-consuming. In response, we design computational methods to automate the creation process from an author demonstration. MixT and DemoWiz capture screencast video and input device events from demonstration of a task in a software application. MixT also records application commands for video analysis. Computer vision and visualization techniques are integrated to segment a video into steps, extract salient information, and add visual highlights. In addition, DemoWiz supports an editing phase where authors can adjust the timing of events in a video. Playback speed of recorded actions can be modified or skipped via an editing UI. Our studies

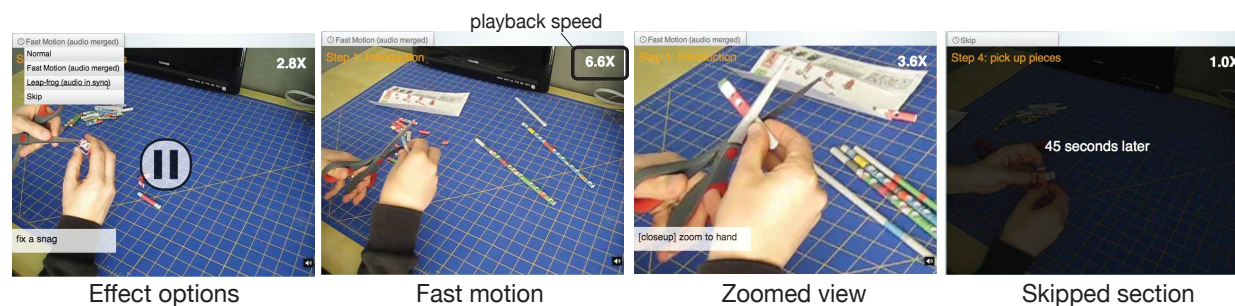


Figure 1.5: DemoCut asks authors to mark key moments in a recorded video of demonstration using a set of marker types. Based on marker information, the system uses audio and video analysis to automatically organize the video into meaningful segments and apply appropriate video editing effects, which can be modified via a playback UI.

showed that our algorithms for step segmentation, event detection, and visualization were effective (<8% error rate in MixT and 0% in DemoWiz).

Interactive Tutorial Authoring from Physical Demonstration

Moving beyond software applications, support for authoring instructions of tasks that take place in the physical world is lacking. Activity recognition remains an open research question, and making authoring decisions during a demonstration can be difficult. To address this problem, we first look into Do it yourself (DIY) project tutorials, which help people learn knowledge and skills to complete a task independently. We developed *DemoCut*, a semi-automatic video editing system that improves the quality of amateur instructional videos for physical tasks (Figure 1.5). DemoCut asks authors to describe key “moments” in a recorded demonstration video using a set of markers. Based on the annotations, our system analyzes the audio and visual activities to automatically organize the video into meaningful segments. Editing decisions are applied to support both *temporal effects* that increase playback speed or skip segments, as well as *visual effects*, such as zooming, subtitles, and visual highlights. A playback interface allows authors to quickly review and edit the automatically generated effects. Our studies showed that video tutorials created by DemoCut in five DIY domains were concise in terms of video length and descriptive instructions with low effect error rates.

Through the process of designing DemoCut for automatic DIY video editing, we observed that for tasks that require larger space and more movements, instructors often have to adjust the position and viewing angle of a camcorder. Some authors choose to set up multiple cameras and later select the best shot from video streams, while some invite another person who controls the camcorder during a demonstration. To enable authors to record their demonstration without acquiring additional cameras or cameraman, we design *Kinectograph*, a video recording device with a single camera that automatically tracks and follows specific body parts, e.g., hands, of an instructor in a video (see Figure 1.6). It utilizes a Kinect depth sensor to track skeletal data and adjusts the camera angle via a 2D pan-tilt gimbal mount. Authors can freely move around in space to demonstrate a task and monitor real-time video preview through a tablet application.

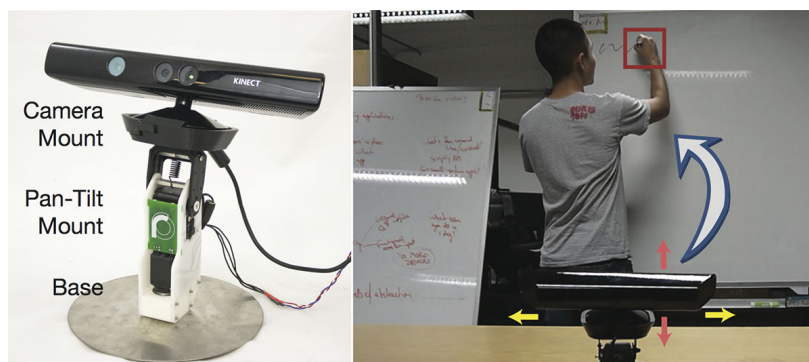


Figure 1.6: Composed of a Kinect sensor to track author movement and a motorized dock to pan and tilt the camera, Kinectograph continuously centers the author (or their hand) in the recorded video for filming physical activities.

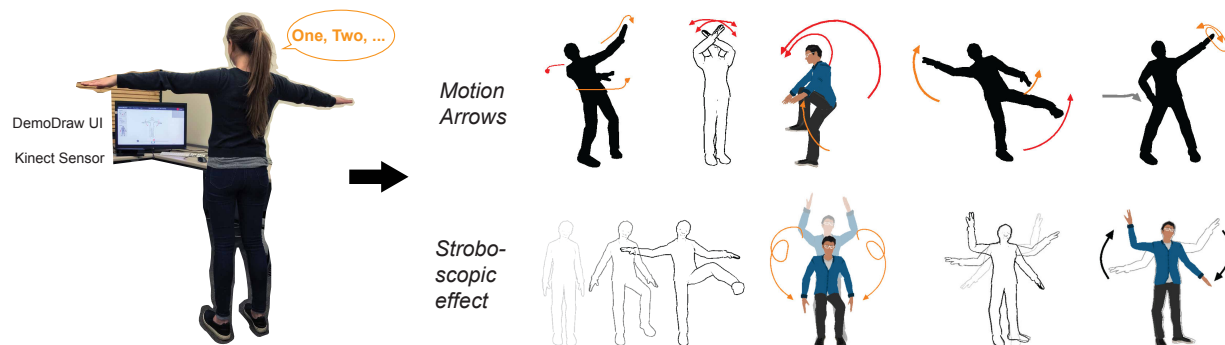


Figure 1.7: DemoDraw’s multi-modal approach enables authors to capture motion, verify results, and re-perform portions to generate step-by-step motion illustrations.

The successful experiences supporting motion-based recordings motivated me to apply our demonstration-based approach to a domain that is entirely driven by movements. In sports, dance performance, and body gesture interfaces, movement instructions are often conveyed with drawings of the human body annotated with arrows or stroboscopic effects [57]. However, current practices require authors to manually sketch or trace subjects from photographs, which is time-consuming and difficult to make changes once created. We design *DemoDraw*, a system that generates concise illustrations from author demonstration (see Figure 1.7). With *DemoDraw*, an author records one or more motions by physically demonstrating in front of a Kinect sensor. In a multi-modal Demonstration Interface, *DemoDraw* segments speech and 3D joint motion into a sequence of motion segments, each characterized by a key pose and salient joint trajectories. Based on this sequence, a series of illustrations is automatically generated using a stylistically rendered 3D avatar annotated with arrows to convey movements. Once a suitable sequence of steps has been created, a Refinement Interface enables fine control of visualization parameters. In a three-part evaluation, our results show 4 to 7-step illustrations can be efficiently created in 5 or 10 minutes on average.

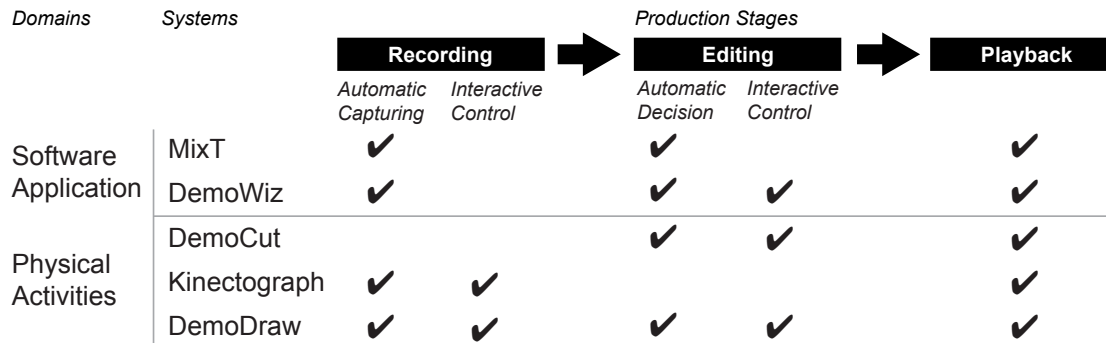


Figure 1.8: A design space of the creation and consumption process for tutorials. It involves three phases of recording, editing, and playback in either software domain or a physical world. This dissertation proposes a series of systems that focus on various aspects in this design space.

1.2 Thesis Contributions

Overall, our video-based approaches consider key events or moments that are important to a learner. This information can be derived from software event logs or human annotation of physical tasks when automatic recognition remains challenging. Based on the metadata and video streams, we propose automatic methods to generate concise instructions for two task domains, software applications and physical tasks (see Figure 1.8). Our approaches support authors from recording demonstrations to editing and reviewing system-generated instructions. Interactive controls are available in different stages via desktop or multi-modal interfaces. We demonstrate a series of systems that consider production stages of tutorial creation and learning. We present the rationale and technical challenges of these interactive system designs. Each system is evaluated both quantitatively and qualitatively to study the usability in authoring and learning.

The contributions of this dissertation include:

- New instructional formats that consider the learning needs from several domains, including software applications and physical activities.
- Multi-modal interaction techniques for novice or amateur authors to create effective instructions by demonstration.
- Automatic or semi-automatic approaches using video and audio analysis that includes authors in the loop to produce high-quality instructions.

1.3 Overview

The rest of this dissertation is structured as follows: In Chapter 2, we define terminology used in instruction creation and consumption process based on literature. We review studies on why people

rely on tutorials in general, how the formats of instructions matter, and the current practices of authoring instructions. In Chapter 3, we review the literature on research and technologies used in supporting activities of authoring and consuming instructions.

We presented two systems that generate interactive tutorials for software applications. In Chapter 4, we present our study on how a new tutorial format supports learners in following step-by-step instructions with mixed media, including static text, images, and video clips. We introduce our creation tool called MixT, which automatically generates such new tutorial format from a software demonstration. Chapter 5 introduces DemoWiz, a system that assists viewers in capturing the timing of input events in a screencast demo video. DemoWiz supports recording, editing, and reviewing stages in a production process with an authoring and playback UI.

Then, we introduced three systems designed for real-world tasks that involve physical demonstrations. In Chapter 6, we present a semi-automatic tool for DIY video editing. Our system, called DemoCut, provides two authoring interfaces, annotation and editing, that enable authors to mark a demo video and review and modify the automatically edited results. The design is based on an fundamental understanding of DIY activities. In Chapter 7, we focus on a recording device that automatically follows a demonstrator for filming instructional videos. The Kinectograph system tracks an author’s position and body parts and provides an authoring interface for real-time camera control. Finally, in Chapter 8, we introduce a multi-modal approach for authors to generate motion illustrations by physically demonstrating the movements. DemoDraw is a system that segments speech and 3D joint motion into a sequence of motion segments and renders effective illustrations. Two authoring interfaces enable authors to navigate, re-perform, and edit visualization parameters.

Throughout this dissertation, we discuss how our video-based approaches increase the quality of amateur-produced video instructions. Chapter 9 concludes our work on tutorial creation and consumption in both software and physical instructions. New directions for future research on interactive tutorials are proposed.

1.4 Prior Publications

This dissertation is based on papers published in previous ACM conference proceedings: the MixT system was published at UIST 2012 [46], DemoWiz at CHI 2014 [48], DemoCut at UIST 2013 [50], and Kinectograph at CHI 2013 [44]; DemoDraw will be published at UIST 2016 [47].

While I am primary author on all publications and led the described projects, this research could not have been completed without my advisor Björn Hartmann and my collaborators that I have been fortunately to work with. Specifically, Dr. Mira Dontcheva and Dr. Wilmot Li at Adobe Research provided valuable guidance on three projects (MixT, DemoCut, and DemoDraw); Dr. Steven M. Drucker and Dr. Bongshin Lee at Microsoft Research guided the DemoWiz project with their expertise on visualization; Professor Daniel Vogel at University of Waterloo greatly contributed to the DemoDraw project. A group of MS and undergrad students at UC Berkeley and Adobe contributed to implementation, design, and user study in the projects, including Sally Ahn and Amanda Ren in MixT, Joyce Liu and Jason Linder in DemoCut, and Derrick Cheng and Taeil Kwak in Kinectograph.

Chapter 2

Background

This dissertation proposes computational methods to create interactive tutorials for software applications and physical tasks. To ground our work in existing practices and principles, in this chapter, we define the terminology commonly used by tutorial researchers and online communities (Section 2.1). We survey research studies and literature about the motivations of people creating, sharing, and consuming tutorials (Section 2.2). Finally, We discuss the common techniques and current practices of creating instructions (Section 2.3 and 2.4). These insights are obtained from the following resources:

- Studies of tutorial authorship and learning [206, 205, 212, 28, 208].
- Guidelines for creating instructions [122, 55].
- An online survey of 2600 individuals across DIY communities [128].
- An analysis of 600 comments to web-based tutorials [131].
- Books and papers of principles about visualization [209, 57, 4], technical instructions [153, 192], and user experiences [92, 36].
- Formative studies from our research projects [46, 48, 50, 47].

2.1 Instructions: Terminology

A *tutorial*, or a *How-To*, is a representation that transfers domain-specific *know-how* by describing a set of *instructions* on how to accomplish a specific task. Torrey et al. [206] defined:

“A How-To refers to online content that describes how something is done.”

Instructions have been widely created for various domains, including:

- Software applications, such as creating a motion blur effect of an image or creating a pie chart from a spreadsheet using specific software,

- Do-It-Yourself (DIY) projects, such as wrapping a gift box, assembling a piece of furniture, or building electronics with Arduino,
- Everyday activities, such as cooking or operating a vacuum cleaner, and
- Sports, such as performing a dance move or correctly executing a golf club swing.

Figure 2.1 shows example activities of these domains from online tutorials.

Instructional content is commonly structured as a list of *steps* or *subtasks* in a linear, *step-by-step* order. It is important to include essential information required to complete a task, such as materials, tools, preparation, expected outcome, and tips [206]. In some domains, instructions have developed into canonical domain-specific formats. For example, cooking recipes contain not only actions but also food ingredients and required amounts.



Figure 2.1: Example activities in tutorial domains: (a) image manipulations using a software application¹, (b) gift wrapping, a DIY task², (c) cooking, an everyday activity³, and (d) golf lessons in sports⁴.

¹ “Photoshop Playbook: Selective Focus”, <https://youtu.be/Wh3ahxqDnyw> ©2016 with express permission from Adobe Systems Incorporated.

² “How to do a Japanese Gift Wrap” by Rouge Shop, <https://youtu.be/Mf3IyeMF8ug>, licensed under CC BY 2.0

³ “How to Cook a Turkey in a Convection Oven” by Six Sisters’ Stuff, <https://youtu.be/QNkwKj1Vsuc>, licensed under CC BY 2.0

⁴ “Pat’s Golf Tips Top of Swing Position” by Blue Rock Golf Academy, <https://youtu.be/H06o7fMQSi4>, licensed under CC BY 2.0

Note that instructions are different from (software) *documentation*, *manuals*, or *user guides*, which are technical documents that contain non-task-centric details about a particular hardware or software system. A manual does not provide step-by-step guidance to follow. To accomplish a task, one needs to identify relevant information from the documentation and reason the workflow.

Instructions can be shown in different forms. There are two main forms of visual tutorials (see Figure 2.2):

- *Static tutorials*, which use text and figures to describe the set of operations required to accomplish a task. Such format is presented as a written document with a step-by-step list, while each step includes text description and/or an image or abstract illustration to describe the subtask. In some domains such as block or furniture assembly, a *diagram* or a pictorial summary shows the step procedure. Static tutorials are suitable for printing and are easy to scan because they show all instructions.
- *Video tutorials*, which are edited or raw videos of the tutorial author performing the task. Examples include screen recordings of a software application or a camera recording of a DIY task. Video tutorials are composed of one or more “shots,” which are “*the basic unit of an editor’s film language, consisting of footage captured over a continuous duration of time without interruption*” [86], to provide the animation or capturing of movements in time. Authors’ faces are often seen in instructional videos to engage viewers [123]. They are effective for presenting the work in action, especially when the activities are hard to describe in text.

Creating Instructions

Instructions are created by one or more *authors* who document the process of completing a task and refine the material to a final deliverable. Professional instructions for industrial products, including software applications (e.g., Adobe Photoshop and Illustrator) and consumer devices (e.g., consumer electronics and cars), are created by technical writers or publishers at product companies. These instructions are provided along with the products, often in a printed format or can be accessed via company websites. From the early 90s when personal computers and the Internet became gradually pervasive, consumer-generated content has thrived [82]. Today, it vastly outnumbers instructions created by companies. Take Adobe Photoshop for example. On YouTube, the official channel¹ hosts 381 videos as of today, but the entire YouTube site hosts over 1.4 million Photoshop tutorials made by its software users around the world. Authors who create these tutorials are often *professionals* or *experts* in the domains of their work. However, in tutorial production, they might be *amateurs* or *novices* who are not familiar with software tools for documentation.

Throughout this dissertation, we call the process of author completing a task a *demonstration* or a *performance*. Authors can therefore be *demonstrators*. Since the demonstration processes often involve innovations and creativity, authors can also be referred to as *creators*, *makers*, or *hobbyists*.

¹ <https://www.youtube.com/user/Photoshop>, accessed July 2016

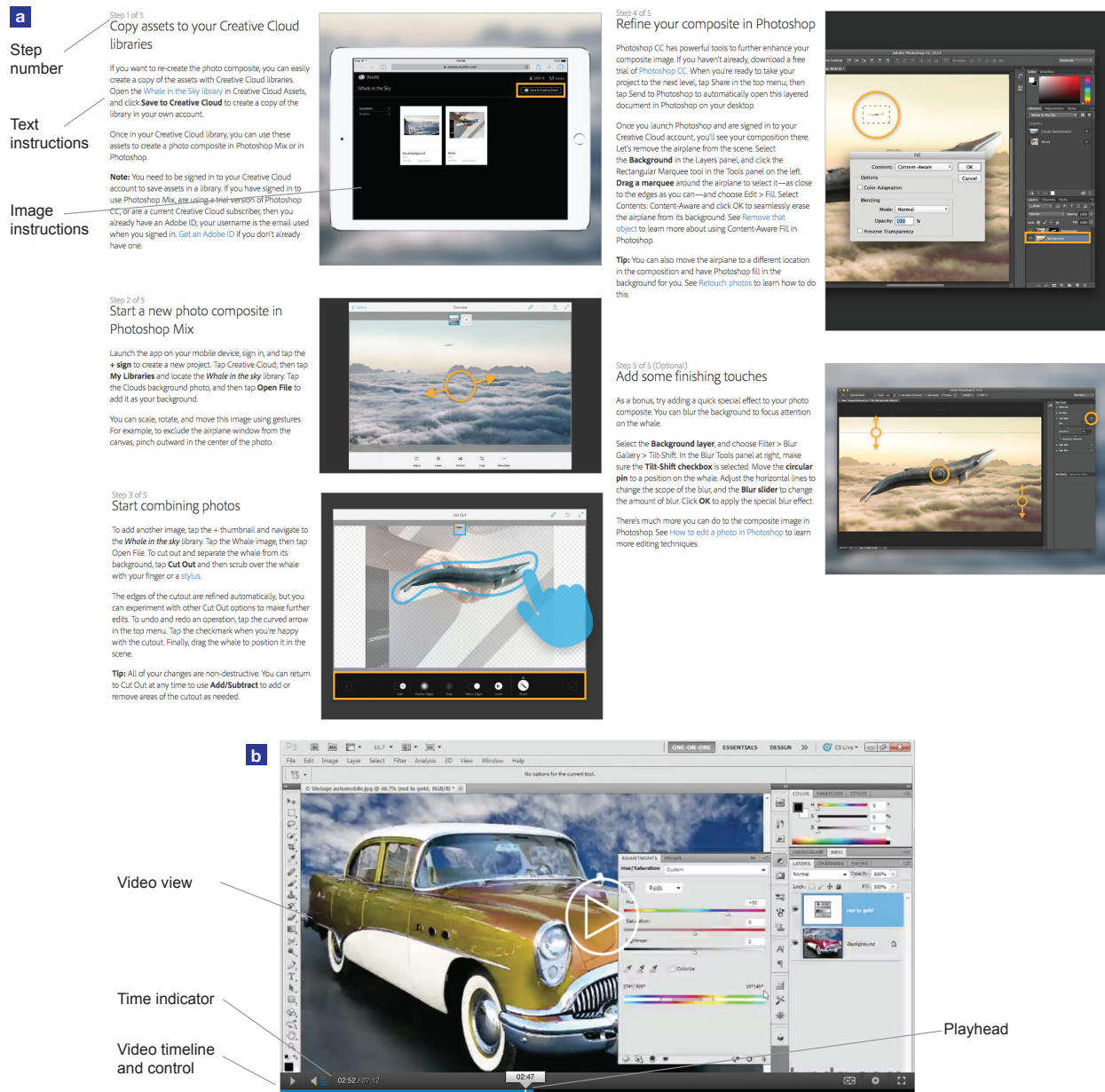


Figure 2.2: Major tutorial forms from online resources^{1,2}: (a) Step-by-step static tutorials show a list of steps, each with text and figure(s) that describe a subtask, such as “*Start combining photos*” or “*Refine your composite*”, and (b) video tutorials show an author performing the task, which can be reviewed and controlled via a video player.

©2016 with express permission from Adobe Systems Incorporated.

¹ Combine photos on the go, <https://helpx.adobe.com/mobile-apps/how-to/combine-photos-photoshop-mix.html>,

² Change the color of an object, <https://helpx.adobe.com/photoshop/how-to/change-color-object-photoshop.html>

Consuming Instructions

We define people who review instructional content as *viewers* or *learners*. Viewers may also be *followers* if they choose to follow the instructions in action in order to achieve the same or similar tasks. In the next section, we discuss why learners rely on instructions.

2.2 Why Authoring and Consuming Instructions?

The research community has been investigating the motivations behind tutorial authors creating written instructions and how-to videos. In the early 90s, researchers studied users who formed online communities to collaboratively customize CAD systems [82]. “*Local experts*” were found to play the key role within user groups. They provided supports to CAD system users of various levels of expertise by sharing customized environments and programmatic extensions. These experienced users were not professional programmers, but they were motivated by the frustrations of following manuals with existing software. They shared materials in order to enhance manuals and help others acquire necessary skills.

The rise of the maker movement from the 2000s introduced massive DIY project sharing via online services. Of the online survey that Kuznetsov and Paulos [128] conducted in 2010, 97% of their 2608 respondents shared and contributed to projects to “*Express myself/be creative.*” In addition, studies showed that one primary motivation of sharing DIY work is to demonstrate expertise [206, 128]. Published tutorials serve as a way to broadcast skill and as an online portfolio. In turn, authors may derive recognition and revenue through advertising or referrals [130], which was similar to how local CAD experts gained formal supports and official recognition, as found by Gantt et al. [82].

Viewers, on the other hand, typically seek technical explanations. It was shown that people prefer web-based tutorials over manuals as they provide “*an immediate, specific goal to accomplish*” and can help learners “*shadow and experience an expert’s work practices*” [131]. Learners also use online tutorials to search for inspiration [205] and look for validation of existing skills [130].

These studies present motivations beyond broadcasting and consuming instructions. Next, we discuss current techniques and practices of tutorial production.

2.3 Techniques of Effective Instructions

Following instructions can require heavy cognitive load and effort for a learner, especially when performing unfamiliar, complicated tasks [198, 197]. Therefore, tutorial authors have to carefully provide concise, effective instructions that can be interpreted by a follower efficiently. The goal is to provide minimal instructional content that includes essential information but retain learners’ attention [28]. A set of techniques has been developed and widely used in technical and everyday instructions. As Agrawala et al. [4] suggested, formulating design principles for effective visualizations requires examining successful hand-designed examples. In this section, we discuss common techniques found in static and video tutorials.

Visual Annotations

To help learners quickly identify key information, visual annotations are commonly used. Table 2.1 shows a list of annotation techniques in practices that we summarized from book material for static tutorials [153, 57, 92, 36, 209], but some of these techniques are often applied in video tutorials (see Table 2.2). These visual elements are effective for indicating the movement of action or objects (via arrows or paths), identify important parts (via highlights), present the details and overview (via call-outs), and provide additional messages such as a warning (via icons or text).

To present annotations, Tufte [209] described the importance of layering information: “*Among the most powerful devices for reducing noise and enriching the content of displays is the technique of layering and separation, visually stratifying various aspects of the data.*” Color, for example, is effective for separating content. Shown in Table 2.1, motion arrows are presented in red, paths are in a darker gray color, and the key components are highlighted in orange or white. The design goal is to help learners efficiently capture important actions and tips. Figure 2.3 shows another example of using colors to separate abstract information. In this exploded diagram of the Hubble Space Telescope, the labels or text annotations are in black, with blue lines pointed to individual parts, that differentiate from the machine components presented in green.

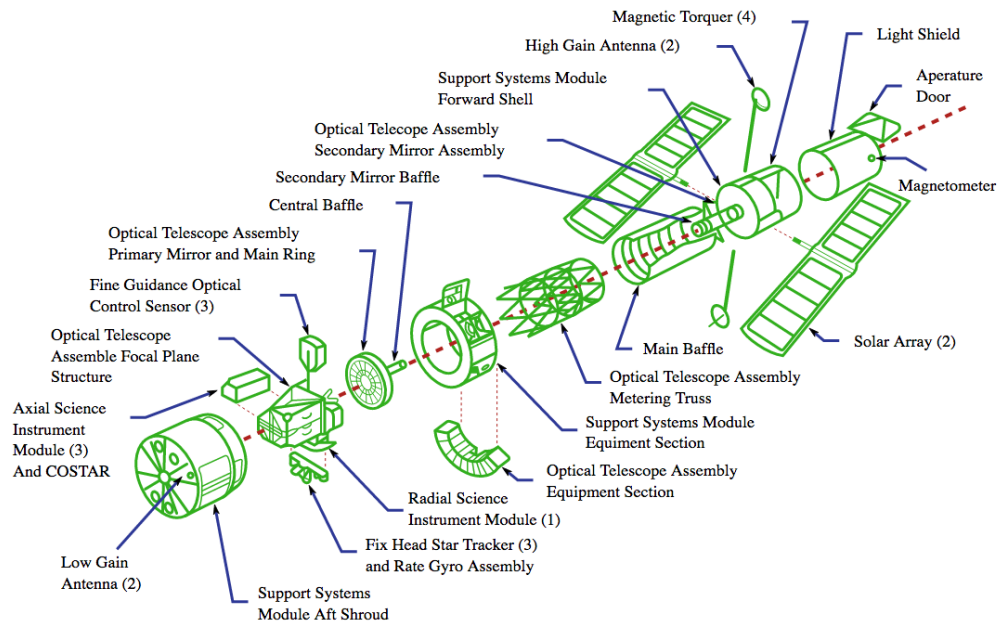


Figure 2.3: Color can differentiate between annotation (labels in black) and annotated information (parts in green in this diagram¹).

¹ Image by AndrewBuck (Own work), “Exploded diagram of the Hubble Space Telescope”, licensed under CC BY-SA 3.0

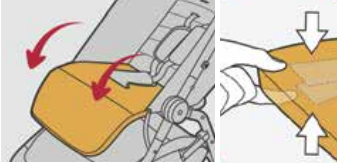

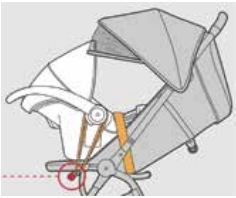
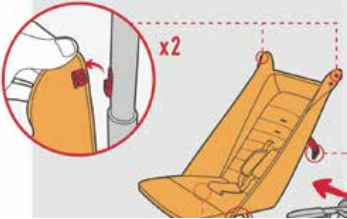
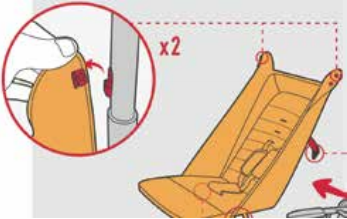


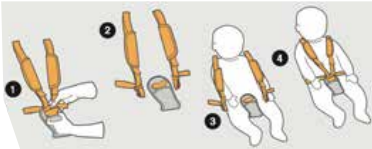
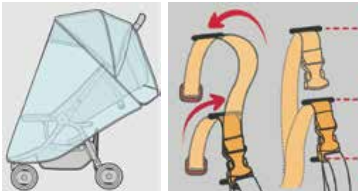
Technique	Example	Intention
Arrow		To show the movement of action or to call for attention. Arrows can be curved and colored. Double-headed arrows can be used to indicate
Path		To show the trail of object movement.
Highlight		To address attention via shapes (such as the red circle here to highlight a hook) and colors (such as orange here to highlight the belt).
Call-out		To present a detailed view while preserving the context.
Text annotation		To provide explanation, such as numbers of similar parts (e.g., "x2") or repeats actions.
Icon		As warnings (e.g., the red cross here as "incorrect" or the green checkmark as "correct"), sound (e.g., the red lines as a click), or other meaning (e.g., "locked").
Hand(s) or tool		To indicate how and where to operate the object with hands or additional tools.
Number		To demonstrate the sequence or ordering of a series of actions.
Overlay or ghost view		To provide the relative spatial relations or to see through the object(s).

Table 2.1: A list of annotation techniques that are often used to provide instructions. Examples are selected from stroller instructions [34]. Reproduced with permission.

Static Diagrams

Static, step-by-step instructions are composed of one or more figures, which can be either illustrations, images, or software screenshots, often with text descriptions. One popular format is to list steps in a document with a sequence of numbered steps. For examples, Figure 2.2a shows a software application tutorial and Figure 2.4 presents a DIY How-To. Images are carefully chosen, often being edited, resized, and annotated from a large set of photos that are taken during demonstrations [208]. In some domains, tutorials are composed of figures without text annotations. Tasks of these domains focus on operating physical components—to illustrate their orientations, positions, and sequences. Furniture assembly instructions made by IKEA² present successful examples. Another common form is to present a series of actions in one combined diagram with a clear layout and step labels, which is often seen in product instructions [153].

² http://www.ikea.com/ms/en_US/customer-service/about-our-products/assembly-instructions/



Figure 2.4: A 5-step static tutorial for a DIY task presented as a web document. Each step includes image(s) and text descriptions. Tutorial by David Hodson [108], licensed under CC BY 3.0.

Video Editing Techniques

Visual annotation techniques are often used in video tutorials (see Table 2.2), such as placing motion arrows next to an action and highlighting with icons or text. These visual enhancements appear before or when an action begins in a video (sometimes with a fade-in effect), and move away while or after an action ends.

In addition to enhancing a shot with annotations, conventional video editing techniques are often seen. A **close-up** is useful to zoom in a shot to demonstrate a detailed view. Figure 2.5a shows one example that the video switches from an overview (to provide context of the position), to a detailed view (to show how exactly the action should be performed), back to an overview (to walk to the other side), and again to a detailed view. Static tutorials can present several steps in a document that learners can quickly skim to get an overview. Since video tutorials present instructions continuously, it is important to help viewers keep track of important steps. **Title scenes** are one common technique for providing a distinct change for a new section (see Figure 2.5b).

When presenting instructions, *operation time* is an important factor to guide learners through a


Technique	Example
Arrow	
Path	
Highlight, text annotation	
Icon	

Table 2.2: Annotation techniques for static tutorials (listed in Table 2.1) are often used in instructional videos, such as arrows and highlights to show product operations¹.

¹ “How to use Loola 3 stroller” by Maxi-Cosi, <https://youtu.be/p6MzLXeWBJw>, licensed under CC BY 2.0



Figure 2.5: Conventional video editing techniques are often seen in video tutorials, such as showing a sequence of overview and detailed shots (a) and a title scene to introduce a new section, which can include animation or movement as a preview (b). Images are obtained from the same video shown in Table 2.2.

task in order to provide awareness of how much time one should allocate. In a concise video tutorial, time is often manipulated by two ways: 1) applying a **fast-forward** or **slowdown** effect to a shot, or 2) jumping (a “cut”) between two consecutive shots, often with a visual **transition** effect, such as fades, dissolves, or wipes. These techniques are commonly used to condense long or repetitive actions in raw footage. The latter is especially common when partial footage is removed, such as mistakes or long breaks [208]. When these effects are applied, it is important to clearly convey the time manipulation to a viewer. Subtitles, text annotation, or narration can provide such indication, e.g., “5 minutes later” or “x10” (which means 10 times faster than the original playback speed).

Online video tutorials are commonly found to be 2-10 minutes long [50]. Our formative study with 6 YouTube authors showed that this strategy provides enough information for viewers to understand the demonstration, but at the same time keep the video lively and interesting. This editing goal is applied to online videos when viewers tend to have shorter attention spans [77, 183], not for conventional industrial video production.

As videos contain a series of steps and details, common video players, however, only support a timeline for navigation. Viewers can jump to a certain point of a video with the timeline, or adjust the playback speed, but the mapping between time and steps is missing. It can also be difficult to glance through the video and reason about the complexity in each step. To fill in the gap between a video and a list of steps or topics, some tutorial authors choose to provide additional text descriptions listing the **index** to the video. Figure 2.6 shows an example from a YouTube video. A list of 12 items and their starting time (e.g., “Assemble (0:04)”) in the video are included in the video description, which is placed below the video player. The YouTube web interface automatically adds hyperlinks to the timestamps shown as blue links. By following these links, viewers can skip to a specific topic.

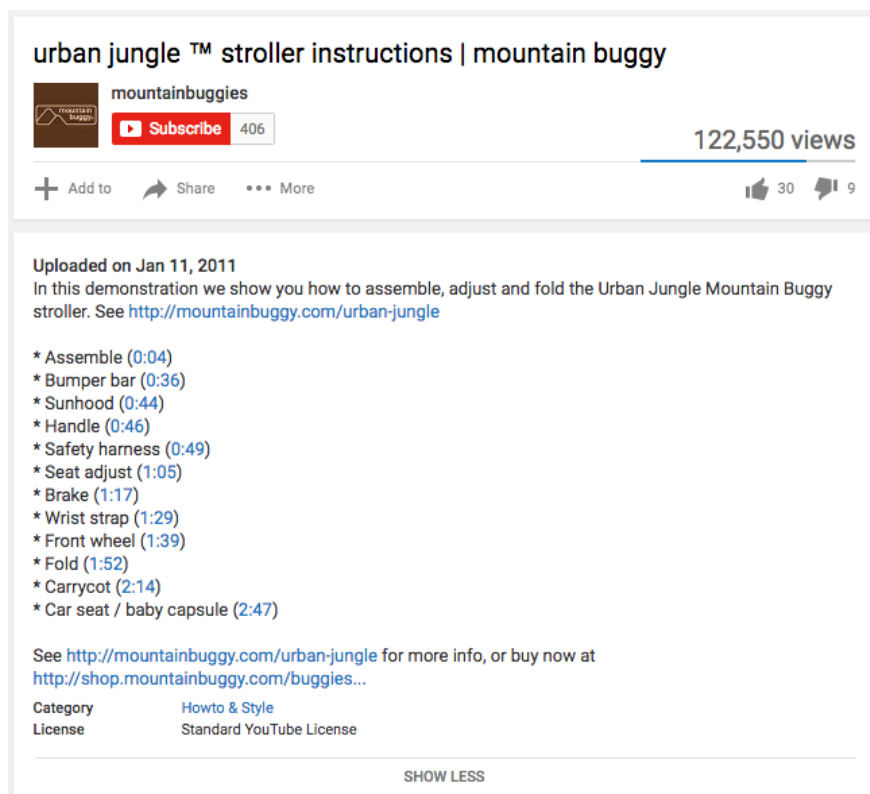


Figure 2.6: Video index to a video tutorial¹ helps viewers navigate between topics.

¹ Mountain Buggy, Urban Jungle™ Stroller Instructions, <https://youtu.be/QwCtdpDmYu8>

2.4 Instruction Production Process

“The practice of writing and sharing DIY tutorials is at the heart of the distributed production and creativity of DIY. Tutorials not only provide tutorship of particular projects, they also develop the skills and competences of those involved in DIY and, in doing so, expand the culture and practices of DIY. It is fair to say that the vitality of DIY practices relies on the effectiveness and quality of tutorials.” – Wakkary et al. [212]

In their paper discussing tutorial authorship, Wakkary et al. [212] pointed out the values of producing DIY tutorials, which we believe can be applied to instructions of other domains. However, the time required to create a tutorial is the primary concern for authors [128, 208], especially for activities that involve multiple steps, tools, and materials. To produce a tutorial, an author may go through several stages:

Torrey et al. [206] proposed a project lifecycle of three phases. First, the “project” involves a goal or a challenge to achieve. Second, the “story” documents the making via words and text. Third, the “contribution” broadcasts the How-To. Similarly, Tseng and Resnick [208] found that “documenting” and “designing” a DIY project for documentation are often separate processes. Our

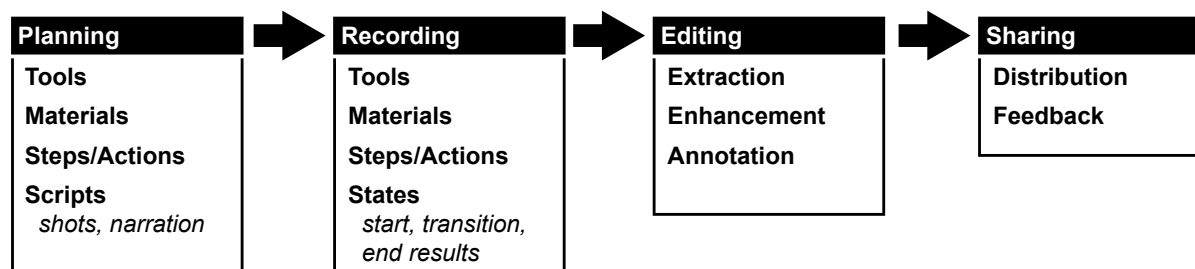


Figure 2.7: A common workflow of tutorial creation, which includes planning the task in detail, recording the process, editing the captured content into a readable form, and sharing with the communities.

interviews with instructional authors on YouTube suggested that a production process can be similar to a filming production [50, 172]. In sum, Figure 2.7 shows a typical tutorial creation workflow that we identified, which includes four main stages: *planning* a procedure of achieving the goal of a task or a project, *recording* the process of completing the project, *editing* the captured material into a form that can be followed, and finally *sharing* the instructions with the communities. Some tutorial creators, especially professionals, may work as a team with multiple role players in a production process, while some work individually. The time required to create a tutorial may vary from a few minutes to weeks. Below we describe the characteristics of each stage in detail.

Planning

Planning or preparation helps authors gather ideas, inspirations, and necessary details prior to building a tutorial project [206]. At this stage, authors decide the project scope and expected outcome, confirm the material and tools required, and design subtasks or steps to be executed. Some authors choose to explicitly write detailed scripts to explain their activities during the recording stage [50]. Figure 2.8 presents examples of instructional video scripts.

Recording

Documenting or recording is the core of creating instructions in order to guide viewers through a specific task. At this stage, authors document the process of a task demonstration via recording devices, such as digital cameras, camcorders, and mobile devices. They capture necessary actions using tools and materials, as well as the changing of states of the elements. Multimedia material has been found to be effective to document procedural knowledge [128, 212, 156], including:

- Static photographs that capture specific moments in a procedure.
- Video footage that records a computer screencast or a scene of a demonstration.
- Audio recordings that preserve the sound of activities or author narration. The narration can be transcribed into text for reading.

a		b	
Video/Screen Captures or Text/Graphic	Script/Narrative	VIDEO	AUDIO
Introduction		FADEUP ON:	
Fade from black to faded out people image for text overlay statistics about food safety related deaths and illnesses	Did you know that on average 1 in 6 Americans gets a foodborne illness each year and an estimated 3,000 Americans die from a foodborne illness annually? That is why it is important to follow the 4 steps in food handling behavior; clean, separate, cook and chill.	1. CAM-1: MS, COLLINS BEHIND KITCHEN COUNTER FULL OF FRUITS & VEGETABLES	COLLINS: With temperatures over 80 degrees year-round, it's no wonder that many great summer recipes hail from the islands of the Philippines. Today, I'll show you a perfect combo that your friends and family will crave no matter what the weather is like. I'm Chef Genevieve Collins, and you're watching Philippine Cuisine.
Video: Spokesperson standing behind a kitchen counter talking medium shot	Let's take a closer look at these 4 important food handling behaviors starting with clean.	1A. CHYRON KEY: GENEVIEVE COLLINS INTERNATIONAL CHEF	
Body		DISSOLVE TO:	
Screen graphic with the word <i>separation</i>	Clean is all about your workspace when preparing food. Illness-causing germs are everywhere around your kitchen. Including your	2. VTR A: PHILIPPINE CUISINE MONTAGE INTRO	PHILIPPINES CUISINE INTRO MUSIC (10 SEC) (10-second intro)
	Be sure you throw food out before harmful bacteria grow	TAKE:	COLLINS:
Conclusion		3. CAM-1: MS, COLLINS	Many of you may be familiar with the traditional Philippine staple food "lumpia" (LOOMP-pee-ya), or eggroll. Well, today I'm presenting lumpia with a dessert twist, using banana as a filling instead of the classic meat and vegetables.
Video: Spokesperson standing behind a kitchen counter talking medium shot, transition into family eating around the table, putting away food.	Some foods are more frequently associated with food poisoning or foodborne illness. With these foods, it is especially important to: CLEAN: Wash hands and food preparation surfaces often. And wash fresh fruits and vegetables carefully. SEPARATE: Don't cross-contaminate! When handling raw meat, poultry, seafood and eggs, keep these foods and their juices away from ready-to-eat foods. COOK: Cook to proper temperature. See the Minimum Cooking Temperatures chart for details on cooking meats, poultry, eggs, leftovers, and casseroles. CHILL: At room temperature, bacteria in food can double every 20 minutes. The more bacteria there are, the greater the chance you could become sick. So, refrigerate foods quickly because cold temperatures keep most harmful bacteria from multiplying. Get the latest tips and techniques to keep these foods safe and prevent food poisoning. Keeping your family your family safer by follow the 4 steps in food handling behavior; clean, separate, cook and chill. Children, the elderly and those with weakened immune systems are especially at risk. Remember, clean, separate, cook and chill!	3A. CHYRON KEY: GENEVIEVE COLLINS INTERNATIONAL CHEF	
Closing graphic with text of web address.	To learn more about the 4 steps to keep your family safe from foodborne illness, please visit foodsafety.gov . This message has been brought to you by the USDA, HHS, and the Ad Council.	4. CAM-2: MCU, PLATE OF EGGROLLS & GLASS OF MANGO SMOOTHIE	Later, I'll pair it with a mango smoothie to create the perfect after-dinner indulgence. Or, you could use the combo as a much more fruitful alternative to your kids' after-school snack.
		4A. CHYRON KEY: BANANA LUMPIA & MANGO SMOOTHIES	

Figure 2.8: Authors often create scripts for instructional videos. Here show examples used in food safety [217] (a) and cooking [71] (b) instructions. Each includes video shot(s) and narration, some with additional notes on the actions. High-level structure can also be specified, such as “introduction” and “conclusion.”

- Other domain-specific content (e.g., sample code, circuit board layouts, 3D models, and sketches) or resources (e.g., books or URLs to other material), which often serve as supplementary material.

Editing

After collecting necessary material, the majority of effort and time is devoted to editing—to turn the raw material to a concise form. Photographs need to be cropped and resized [208], sometimes annotated [206]; videos require removing footage, applying visual effects and transitions; audio has to be processed to omit utterances and condense narration [50]. Popular editing tools include Adobe Photoshop, Adobe Premiere, and Apple Final Cut Pro. Emerging mobile applications such as Snapguide have been adopted by novices.

Different multimedia forms can be mixed to create tutorial formats mentioned in Section 2.1. Videos, in particular, have become a popular medium to document and demonstrate the functionality of the work:

“Videos, for instance, require a long time to edit, but can influence the viewer in at least three powerful ways: 1) by physically illustrating the steps required to create an artifact; 2) by showcasing a new idea in its functional form; and 3) by directly ‘speaking’ to and engaging with the audience.” – Kuznetsov and Paulos [128]

Sharing

Finally, authors may release the refined content and share the deliverable with others. Common media or platforms include: personal blogs, content sharing sites, forums, emails, or personal networks. Some of these channels offer ways for the audience to provide feedback and contribute. Lafreniere et al. [130] specifically analyzed the online comments of web tutorials. Their study shows that the audience has several purposes to communicate via comments, including technical validation and refinement, which can be useful for authors to improve their documentation techniques.

2.5 Summary

These studies and observations suggest that tutorials have a larger variety of purposes and uses than merely communicating technical content. In this dissertation, we strive to make authoring of instructions more accessible to amateurs while maintaining opportunities for adding individual style through control over editing effects. Next, we survey prior work on computational methods of supporting authoring and following instructions.

Chapter 3

Related Work

The HCI and computer graphics communities have introduced novel technologies for authoring tutorials, including automatic generation methods and interactive editing tools. In this chapter, we survey state-of-the-art techniques for generating instructions for both software applications (Section 3.1) and physical tasks (Section 3.2). Furthermore, today’s tutorials are often offered within limited conventional media, such as static tutorials (print-outs or web) or videos. In recent years, interactive systems have shown versatile ways for users to review instructional content. We will discuss the forms that instructions have taken in prior research, which will lead us to discuss shortcomings of tool support for creating and navigating tutorials. Finally, in Section 3.3, we review the methods of video analysis and playback to discuss how they support video-based interactions.

3.1 Generating Instructions for Software Applications

Input Event Visualization

Studies have shown that visualizing input events during user operations can make software applications more learnable [62]. When viewing software instructions, visualizations can help learners anticipate where the cursor is moving, tell what actions are being performed, and locate activated UI components that may be otherwise hard to notice.

Input events that can be visualized range from low-level, application-agnostic input device events (e.g., mouse actions, cursor movements, or keyboard strokes) to higher-level, application-dependent information (e.g., menu selections or UI component changes). Commercial tools such as Mouseposé¹ and ScreenFlow² visualize mouse events and keystrokes with special effects, e.g., drawing a circle around a mouse cursor (see Figure 3.1a). These tools capture input information (e.g., mouse position and event type) and render visualizations on top of the screen activities. This approach has been widely used by online video tutorial authors. However, it does not consider

¹ <http://www.boinx.com/mousepose>

² <http://www.telestream.net/screenflow>

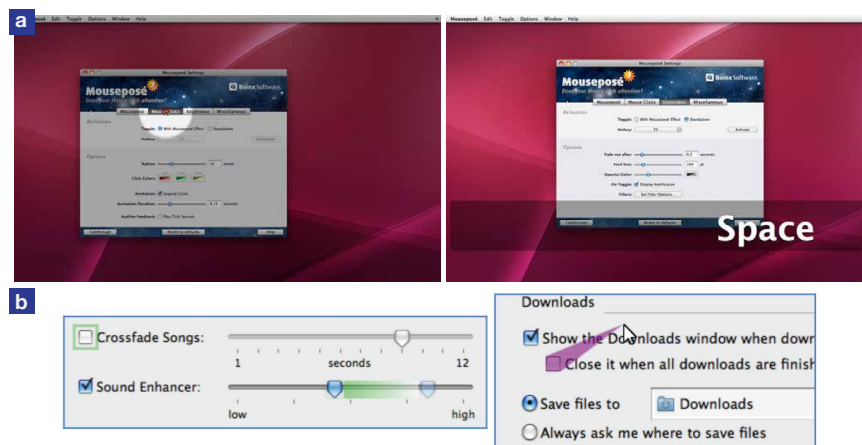


Figure 3.1: Real-time visual enhancements to GUI applications are commonly used in instructional videos. Mouseposé highlights a mouse cursor (a, left) and displays keyboard input (a, right). Prefab [62] creates effects such as target-agnostic afterglow [21] (b, left) and target-aware cursor [93] (b, right) by identifying and reverse engineering UI components.

application context, which can be difficult for learners who want to follow specific instructions at a semantic level, such as observing text field completion or menu option selection.

To enhance UI components (e.g., a checkbox, button, or editable text field), Dixon et al.’s Prefab [62, 63] modifies screen pixels in real-time by detecting target features, such as region corners. This semantic understanding of GUI elements enables component-based highlighting effects, such as afterglow [21] that visualizes user operations (see Figure 3.1b left). It also improves interactions by incorporating techniques like the Bubble Cursor [93], a target-aware pointing method that suggests the nearest target (see Figure 3.1b right).

The methods above enable real-time visualization of input events when a user is interacting with an application or viewing playback of an application in use, which can be useful for following a video tutorial. But to present events in a static tutorial, screenshot images are commonly annotated using different techniques in order to visualize continuous actions, such as navigating a menu from the root to a sub-panel. Applying motion arrows is a popular way to provide a sense of direction and start and end positions of mouse events. Researchers have investigated automatic approaches that capture and visualize these types of events in representative screenshots from author demonstrations. Nakamura and Igarashi [159] proposed a capturing and rendering system independent of specific GUI applications. Their system logs mouse events during a software demo, including mouse movement, dragging, and clicking. Operations are rendered as markers and arrows on screenshot images to present the linear event history (see Figure 3.2A). Grabler et al.’s approach [91] further annotates a screenshot with bounding boxes and call-outs, which help learners identify parameters and specific functionality available in the interface (see Figure 3.2B).

The systems we present adopt some of these successful techniques to enhance software instructions. By capturing event information at both input device and application levels, we visualize author operations differently based on how the learner is viewing instructions. During *video playback*,

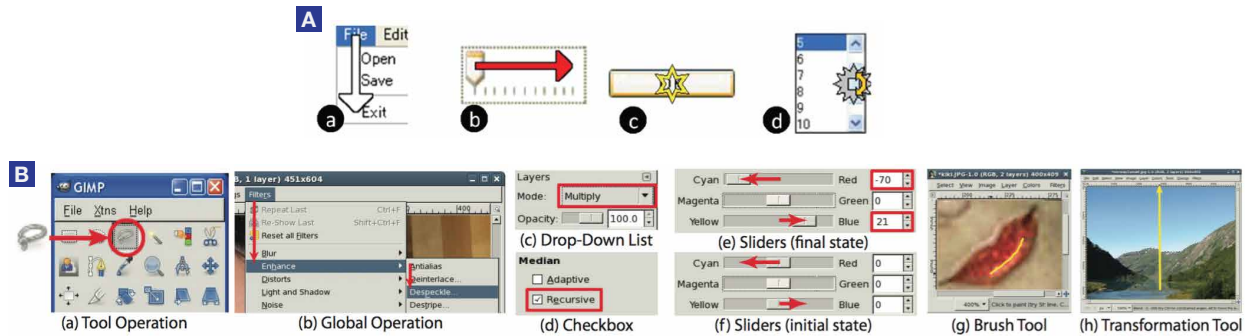


Figure 3.2: Examples where software operations are automatically rendered on top of application screenshots, including moving the mouse, dragging, clicking, and scrolling by Nakamura and Igarashi [159] (A) and application-specific operations (a-b), parameter setting (c-f), and image manipulations (g-f) by Grabler et al. [91] (B).

MixT shows mouse events and trails, and DemoWiz overlays glyphs and arrows to guide viewers from the current input event to the next. Screenshot images in MixT’s *static* tutorials are annotated with mouse visualizations, such as highlighting a drop-down menu item that will be selected.

Workflow Capture and Tutorial Generation

In addition to visualizing input events, it is important to present the entire workflow in a tutorial and provide concise instructions. Grabler et al.’s system [91] generates a step-by-step tutorial from an author’s demonstration (see Figure 3.3). Generated tutorials include textual description, such as “*Select the **path** tool from the **toolbar** to **create and edit paths***” from text templates, along with annotated images of user operations. Designed to guide image manipulation tasks, their system analyzes the application context, including facial features and outdoor scenes in manipulated images, to enhance instructions. Such demonstration-based approaches have been applied to generate instructions for software that involves complicated manipulations or gestures, including 3D mesh construction [59] and touch-based mobile applications [213]. Beyond logging input events during an author’s demonstration, researchers have shown that workflows and user interface actions can be captured using computer vision by analyzing the pixels of desktop regions [221, 42] and existing screencast videos [14].

With many tutorials to choose from, it can be useful for learners to have sophisticated ways to compare alternatives. To compare effects of individual operations in a workflow, showing a list of “before” and “after” thumbnails, video clips, and event timelines can be effective [96], as Chronicle demonstrated for image manipulation tasks (see Figure 3.4a). When there are multiple workflows that produce similar results, side-by-side documents and a union graph, which shows the edit distance as a cluster view, can help learners choose the right workflow based on the operations each one uses [126] (see Figure 3.4b).

These systems provided insights on 1) methods to automatically generate step-by-step tutorials and 2) workflow presentations serving different purposes. Our MixT system draws on these past

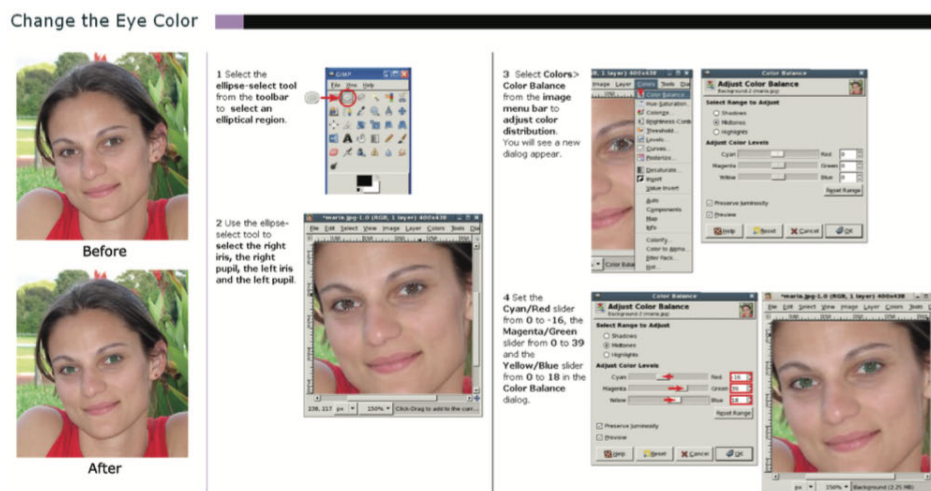


Figure 3.3: A static tutorial automatically generated by Grabler et al.'s system [91].

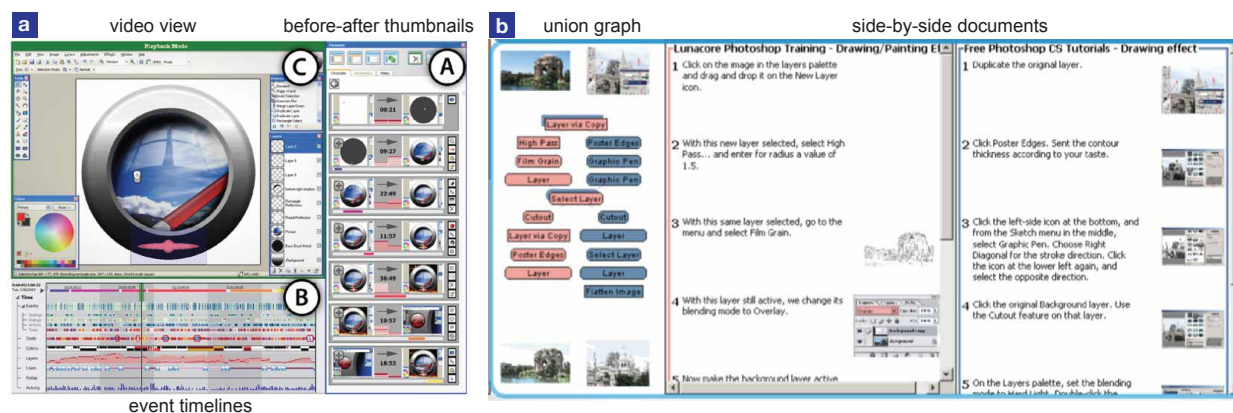


Figure 3.4: Instructional systems that help learners compare image manipulations and similar tutorials using before and after images and event timelines by Grossman et al. [96] (a) and side-by-side documents by Kong et al. [126] (b).

insights. It is build on top of a Photoshop plugin³ derived from Grabler et al.'s approach [91] to produce a step-by-step document with text descriptions of each workflow operation. We enhance the static tutorial format by embedding instructional video clips for each operation that can be interactively viewed.

The paradigm that these research projects explore opens a design space to create new tutorial formats for software applications. Our approaches show promise for helping learners effectively use tutorials. In recent years, researchers have shown that learners perform better when using responsive video tutorials [160] and learning-by-doing-activities [129], than when using standard video or static instructions.

³ Adobe labs. Tutorial Builder. <http://labs.adobe.com/technologies/tutorialbuilder/>

In-Application Support

The above methods introduce innovative ways for learners to review workflows and instructions. However, learners usually review these materials in an interface completely separate from the application they are learning to use. Learners might have to switch between the main application they are using and a separate set of instructions. This could widen the gulfs of execution and evaluation [109] by making it more difficult for users to recall the actions to perform (*“How do I perform the action that the instructions describe?”*), or to evaluate whether they successfully followed instructions (*“Am I doing this right as the instructions explain?”*).

To reduce these gulfs, researchers have proposed another approaches to provide “in-application” assistance, often in real time, in a specific application context. First, there has been a considerable amount of research devoted to offering interactive help to support learners comprehend the functionalities within the application. Crystal [158] enables software users to ask questions about why something did or did not occur in an application. ToolClips [94] shows that video snippets can be embedded in application tooltips to explain specific functionality, which were shown to be seven times more effective than conventional tooltips for for users trying to complete unfamiliar tasks.

Second, interactive, step-by-step instructions can be integrated into an application in several forms: To help software users identify specific UI components, tutorials can be shown via a translucent “stencil,” which visually directs a user’s attention to relevant UI elements [116]. By tracking a user’s current actions, tutorials can be integrated with an application to provide indications of progress with check marks [76]. Video tutorials can be programmed to automatically pause and play as users perform each step [174]. Instructions can be captured from demonstrations as “scripts” to later guide step-by-step navigation within the interface [24] or shown as ambient help [145]. In-application controls [139] and game elements [135, 64] can further engage users in learning about the application.

Third, as tutorials are built for a broad community with many authors and learners, content can be dynamically updated by a community through user contributions [132, 148, 35].

Our work focuses on authoring tools to create novel tutorial format designs, though it doesn’t specifically focus on learning support. We see opportunities to provide the tutorials we generate as a form of in-application guidance. For example, a video clip from a MixT or DemoWiz tutorial could be automatically replayed when a system detects a slowdown of a learner’s progress on a specific step. However, we do not claim to make a contribution in this direction.

3.2 Generating Instructions for Physical Activities

The above approaches to capture software demonstrations open the door to enable interactive tutorials that can respond to user progress. However, capturing and tracking user behavior in the physical world, rather than in software, remains challenging. How can technologies track humans and objects in a space to support real-time feedback? What are the available authoring techniques to generate instructions for physical tasks? This section discusses the challenges to approaches that

researchers have taken from four perspectives, including tracking activities, authoring instructions, presenting guidance, and enabling interactive instruction following in the real world.

Tracking Physical Activities

To record activities and provide interactive feedback, a computer system needs to detect user operations and objects in real-time. Computer vision techniques can automatically track specific physical targets shown in a video and enable novel interactions. This includes two types of targets:

- **Tracking objects.** Instructional tasks often involve one or multiple objects, such as a ball in sports, utensils in cooking, and a variety of tools in DIY projects. Common techniques to track an object include: 1) Tracking specific *colors* or *visual features* of pre-defined objects. Examples include tracking a fast-moving Ping-Pong ball for automatic camera control [163] or paper puppets for creating animations [16]. Color and feature information can be supplemented with depth information, which could obtain a better understanding of the objects' positions in the 3D world. Tracking objects in this way is useful to capture activities that involved object manipulations, such as block or toy assembly tasks [98, 218] and 3D puppet control [105]; 2) Tracking *motion-capture markers*. The most common method is to attach reflective markers to an object's surface. This enables accurate capturing of motions like an animator's continuous movements [65].
- **Tracking humans.** Human activity can be tracked in many ways, often determined based on what activity is being monitored. Targets include *faces* (e.g., to show a close-up of a speaker during video conferencing [179], to provide real-time camera control guidance when filming an interview [37], and to capture facial performances [190, 204]); *hands* (e.g., to enable gestural control [202] and camera control of a repair task [178]); *user movements* (e.g., to provide augmented content such as ambient information [216] and dancing [6]). Motion capture markers can be used to track actors in addition to objects in professional filmmaking. However, since markers are visible in video recordings, visual effects (VFX) post-processing is necessary to hide them from the final rendering.

For many physical tasks, it's important to track both objects and humans. One example is reconstructing 3D scenes of athletes throwing basketballs [66]. Some of these vision-based systems require a high-speed camera [163] or a RGB-D camera [98, 218, 105, 216, 6, 66], while some require users to wear reflective markers [178]. Other non-vision tracking methods rely on sensors attached to an object or human, including GPS sensors [107] and wireless radio frequency sensors [162]. These are often used to track a moving target in a larger space, such as a flying drone. Finally, if video content is difficult to be extracted into semantic information, crowdsourcing algorithms have been introduced to segment step-by-step videos with the help of online workers [120].

The detection mechanisms from these systems inspired us to design interactive systems that can react to an author's activities without requiring them to carry a sensor. For example, Kinectograph and DemoDraw track authors' body parts using a Kinect sensor that is widely available to consumers.

However, tracking techniques to detect high-level task information, such as the *intent* of a certain action, is still difficult to achieve. When automatic activity recognition is difficult, we include authors in the loop to annotate a task. DemoCut provides an annotation interface for describing high-level actions in DIY videos; DemoDraw provides a multi-modal interface to label continuous body movements.

Authoring Instructions for Real-World Tasks

From current literature, we identified two major approaches to create instructions for physical tasks: model-based and demonstration-based. A *model-based* system analyzes the structure of an object or a task to generate instructions of how the object is constructed or the task is performed. Early approaches by Feiner and Seligmann [74, 187] considered communicative intent and rules of object manipulation to create effective illustrations from 3D models. Their automatically-generated diagrams showed that actions such as snapping latches can be effectively expressed by motion arrows and a cutaway view. By analyzing object geometry and other attributes, Agrawala et al.’s [5] system automatically renders step-by-step assembly instructions for furniture and toys (see Figure 3.5a). Technical diagrams can also be generated, such as exploded views that explain mechanical assembly parts [136] and motion illustrations that describe how individual parts are operated [154]. Parts are often highlighted using colors, labeled with text; causal chain sequences of mechanical interaction can be shown as a list of highlighted figures, annotated with motion arrows (see Figure 3.5b). Conversely, an existing technical document can be automatically analyzed and transformed into a 3D animation by parsing its parts, orientation, and visual annotations [155].

A *demonstration-based* system records an author’s physical demonstration of a workflow. The captured materials can be either automatically analyzed by the system or manually edited by an author to produce instructions. One approach is to employ templates to help users capture sequences

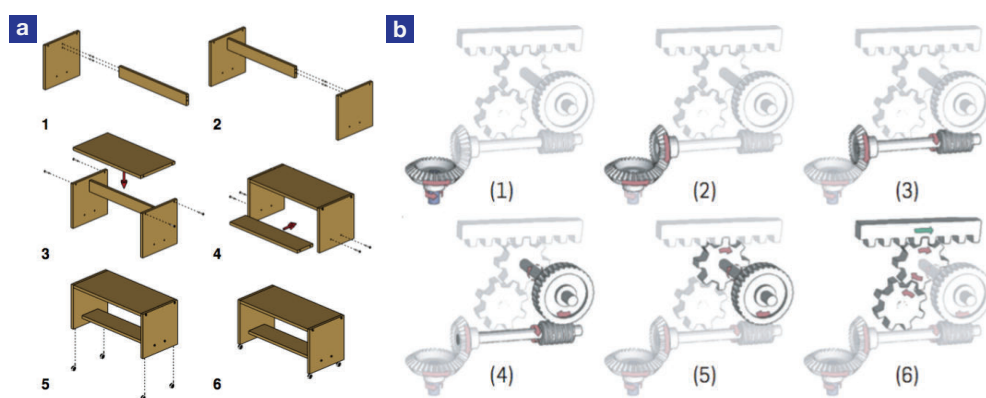


Figure 3.5: Instructional diagrams can be automatically generated with a model-based approach, such as assembly instructions by Agrawala et al. [5] (a) and causal chain sequences of mechanical interaction by Mitra et al. [154] (b).

of distinct shots (e.g., Snapguide⁴) or provide a limited set of operations to record specific moments (e.g., adding or removing a part in a block-assembly task [177, 98]). Another approach is to provide a general-purpose interface to support authors produce multimedia materials during or after a demonstration, for example through an interface on a head-mounted video capture device [38]. For certain tasks, new recording devices need to be specially designed, such as an integrated device that includes an IR camera to capture a knitting process [182] and a turntable to record the building process of a DIY project [207].

In this dissertation, we support a wide variety of physical tasks from craft to home repair and cooking where tracking user activities cannot be done completely automatically. Since the instructions produced in these domains are often highly creative and tailored to the author’s style, we focus on the demonstration-based approach, which we feel is better capable of preserving the author’s sensibilities. Kinectograph and DemoDraw each allow authors to perform a physical demonstration in front of a camera, while capturing the activities given author’s authoring decisions. DemoCut’s annotation interface enables users to describe high-level, step-by-step instructions, such as tools and actions.

Presenting Guidance

To help learners follow instructions for physical tasks, we discuss how guidance has been displayed in the 3D world. There are four main approaches to present real-time guidance.

First, rich information can be shown via an *external display* placed next to the work area. Applications in several domains adopt this method, in the domains of cooking [210] and block assembly tasks [98, 218]. One challenge of external displays is that a learner may need to switch their attention often between the task and the instructions. To make instructions more easily accessible, Knibbe et al. [125] designed a table with an embedded display as a physical workspace that monitors, records, and assists users. In this way, workers could review the information on the table’s surface while making a project.

Second, information can be *overlaid on top of the work area* with a projector. Examples include remote repair tasks [99] (see Figure 3.6 right), assembly tasks [121], cooking [114], and learning to play the piano [219]. For tasks such as dance movements, guidance can be shown on an augmented mirror that learners can reference [6]. This method can effectively present instructions at a location that can be viewed most of the time by a learner as they perform the activities. However, this often requires a tailored indoor environment setup and a calibrated projector, which is difficult to integrate into most realistic settings where learners are doing these tasks.

Third, instructions can be *displayed through a head-mounted device or mobile phone*. Researchers have built AR applications that provide visual highlights for machine maintenance [106, 155] (see Figure 3.7), enable interactive touring in a city [75], explain product functionalities [143], and present dynamic user interfaces based on head orientation [222]. Compared to other approaches that project information onto a publicly visible surface, this method allows learners to access information privately. Multiple users can interact with an augmented system at the same time.

⁴ <http://snapguide.com/>

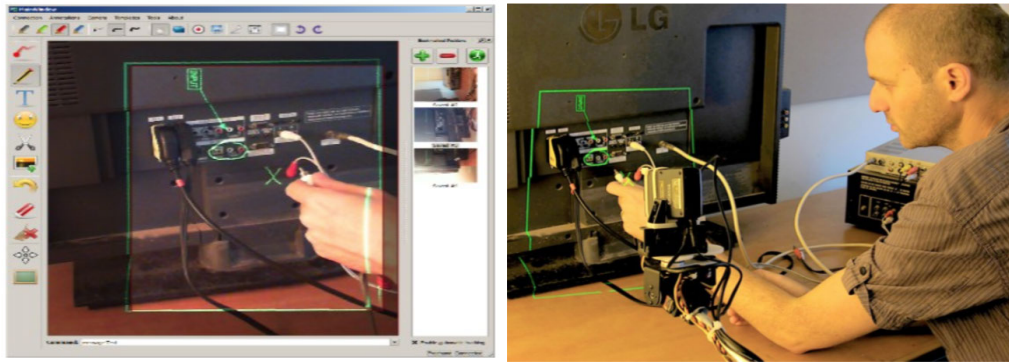


Figure 3.6: TeleAdvisor [99] provides an authoring interface (left) for an instructor to guide a remote worker through a repair task (right).

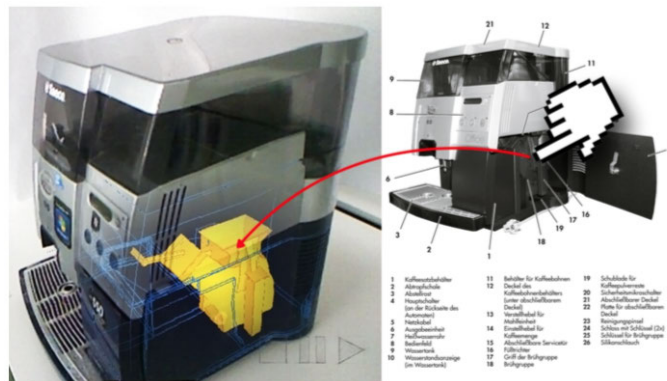


Figure 3.7: Work by Mohr et al. [155] automatically analyzes a technical document and augments a machine with AR animations in 3D to help novices operate an unfamiliar machine.

However, learners have to wear or carry a device, which might limit their mobility while performing a task, such as machine repair that requires both hands.

Last, for specific tasks, information can be conveyed *directly via target objects*. Haptic feedback has shown to be useful to help learners follow steps for physical tasks, including sculpting [225, 3], building multi-material assemblies [186], and learning Frisbee [193]. Visual cues such as LED patterns on a device can direct user's attention to physical features they must interact with [193, 211]. This approach is task-specific and can be difficult to generalize to other task domains.

Providing Activity-Based Guidance

Finally, we discuss how assistive technologies track a learning process and support guidance based on learners' progress. For some specific tasks such as block assembly [98, 218] and dancing [6], learners' progress can be accurately tracked in 3D. This enables a tutorial system to provide real-time feedback about the learner's performance and what to do next, e.g., where to place the next block to the current model or suggested adjustment on a dance pose. However, as we discussed earlier,

tracking activities at the instruction level for real-world tasks can still be inaccurate. Therefore, the majority of systems supporting interactive physical guidance enable learners to navigate instructions. For a pair learning scenario, a remote instructor can manually provide instructional input based on the learner's progress that he or she sees, such as highlighting a component for remote repair tasks [99, 121] (see Figure 3.6 left).

Our work focuses on designing tools for tutorial authors to create instructions from task demonstrations. DemoCut is designed for authoring instructions after capture time. Kinectograph provides a control interface on a tablet device for authors to carry and place in a space. DemoDraw's demonstration interface is similar to Anderson et al.'s [6] augmented mirror. But instead of overlaying instructions, we present a real-time render view of an author's movements via an external display placed in front of the user.

3.3 Working with Videos

Research in video understanding has introduced new ways to work with videos. How should authoring tools help users record, organize, and edit necessary materials? What interaction techniques can help authors and viewers navigate one or more videos? In this section, we review these questions and their answers briefly by considering current research tools designed to help users capture, edit, and navigate video.

Tools for Capturing Video

Tools that support video capture focus on these subtasks:

Shooting Suggestion. Several research systems guide users at capture time to record higher-quality videos. Real-time suggestions can help camera operators frame subjects (e.g., NudgeCam for interview videos [37]) and provide suggestions for actors' performance (e.g., to speak louder or exaggerate a performance [103, 58]). Shot suggestions can also be bootstrapped through user dialogs [1]. Other researchers recommend patterns from expert storytellers and common sense to help novice authors capture materials and develop a story structure [17, 118].

Automatic Camera Control. Viewpoints of stationary cameras can be automatically determined based on heuristics at record time in order to track actor, area, or object [178, 163]. In recent years, quadrotor cameras enable a wide range of trajectories to capture subjects from different viewing angles. Roberts and Hanrahan [181] proposed an authoring tool for authors to plan and preview a camera trajectory. Some tools have enabled actors to control quadrotor cameras by using 3D gestures as they are being filmed [40, 171].

We proposed Kinectograph prior to these systems of quadrotor camera control. A recent commercial system has included a similar feature to track a moving user [107], which is based on GPS information instead of specific body parts.

Tools for Editing Video

Tools that support video editing focus on these subtasks:

Annotation. Researchers have investigated interactions that enable efficient, fluid annotation or labeling of video data. One example is the EVA system [142] that encourages authors to annotate materials at capture time. More recent interfaces accept pen input (e.g., VideoTater [61]) and touch or gestural input [185] for content-based annotations, such as tagging a subject in a video.

Storytelling. When working with a repository of video clips, it can be challenging to compose a compelling story. Several new interaction techniques have been proposed to make it easier to explore story elements: A storyline can be created non-linearly based on relevant characters, emotions, and themes of the current edited clips [189]. Tangible controllers with a specialized table interface can enable collaborative, non-linear editing [19, 18]. Live authoring at capture time with a tablet device can allow an author to quickly organize clips and apply editing decisions [79].

Editing. Frame-based video editing is very time intensive, as it forces users to operate on very minute details. Editors can leverage *metadata*, such as shot boundaries [39] and transcripts [27] that help users place cuts and transitions. This gives users higher-level editing operations at the shot level rather than the frame level. Techniques of *computer vision* and *speech analysis* can automate certain visual effects, such as creating cinemagraphs [13, 113], automatically editing lecture videos [102], creating zoomable tapestries and synopses [15, 175], and stabilizing shaky amateur videos [140]. Edits can take place during recording, such as switching to a close-up view of a person who is speaking [179]. When material like character animations can be reviewed in 3D, camera angles can be optimized to render a new video by analyzing the actor's motion data [10, 11]. Finally, when video analysis is a matter of subjective taste, identifying salient frames or highlights can be outsourced to crowd workers [26, 200].

MixT, DemoCut, Kinectograph, and DemoDraw also use computer vision techniques for making automatic editing decisions. They differ from previous approaches in their focus on two particular application domains – software and physical demonstration videos. By focusing on a specific domain, MixT and DemoCut can make assumptions about the structure of the input and output video, such as the fact that there is a linear set of steps. DemoCut offers an authoring interface that makes it easier to create high quality instructional videos. Kinectograph makes editing decisions (e.g., pan-and-tilt, zooming) based on an actor's body location in video frames. DemoDraw includes a multi-modal interface where authors can annotate a recorded body motion using speech while physically performing movements.

Tools for Navigating Video

Video playback can be controlled by inferring user intention from their actions. For example, segments can be played back [174] or speed can be modified [45] based on a user's actions in a software application. Videos can be navigated at the content level beyond a linear timeline. For instance, subjects' movements can be visualized in a storyboard [84] or continuous image mosaic [203], and timelines can be navigated by manipulating a target in 2D [68, 85, 115] or 3D [161]. These techniques help viewers understand content flow and navigate videos, and

have been applied to screencast videos [60, 160]. Canvases of video tiles and timelines [100] or thumbnails [147] can make navigating long videos or sets of videos faster. Video digests can be an effective way for viewers to browse and skim video content [170].

These novel forms of video navigation inspired us to help viewers navigate video content more effectively with our tools. MixT supports per-step video navigation embedded in a static tutorial. DemoWiz augments a screencast video with novel visualization to help users view the content. DemoDraw renders a series of human movements as concise motion illustrations.

Summary

Our approaches take video as system *input* (i.e., to track changes of salient UI components of screencast videos in MixT, DIY activities in DemoCut, and body movements in Kinectograph and DemoDraw), as well as *output* (i.e., to offer interactive instructions with MixT’s per-step video segments, DemoWiz’s augmented screencast recording, DemoCut’s concise video, and Kinectograph’s instructor-focused video). We design user interfaces and algorithms for authors and learners to interact with video-based instructional content.

Chapter 4

MixT: Mixed Media Tutorials

Users of complex software applications often learn concepts and skills through step-by-step tutorials. Today, these tutorials are published in two dominant forms: static tutorials composed of images and text that are easy to scan, but cannot effectively describe dynamic interactions; and video tutorials that show all manipulations in detail, but are hard to navigate.

This chapter presents a novel design called *mixed-media tutorials*¹, which include static instructions and per-step videos to combine the benefits of both formats. After providing an overview of related work (Section 4.2) on interactive tutorials, in Section 4.3, we describe a comparative study of static, video, and mixed image manipulation tutorials with 12 participants and distill design guidelines for mixed tutorials. Section 4.4 introduces MixT, a system that automatically generates step-by-step mixed media tutorials from user demonstrations. MixT segments screencapture video into steps using logs of application commands and input events, applies video compositing techniques to focus on salient information, and highlights interactions through mouse trails. Finally, in Section 4.5, informal evaluation suggests that automatically generated mixed media tutorials were as effective in helping users complete tasks as tutorials that were created manually.

4.1 Introduction

Learning how to use software applications often happens opportunistically as users need to accomplish specific tasks. When it is unclear how to achieve the desired results, many users turn to step-by-step tutorials, which describe the set of operations required to complete a task. Visual editing applications, such as applications for drawing, photo editing, and 3D modeling, require visual tutorials that show not only how to navigate the user interface but also how to manipulate the canvas, image, or 3D model.

There are two main forms of visual step-by-step tutorials. *Static tutorials* use text and images to describe the set of operations required to accomplish a task. *Video tutorials* are screen recordings of the tutorial author performing the task. Both forms of instructional content have strengths and weaknesses. Static tutorials are easy to scan forward and backward because they show all

¹ This work was published at UIST 2012 [46].

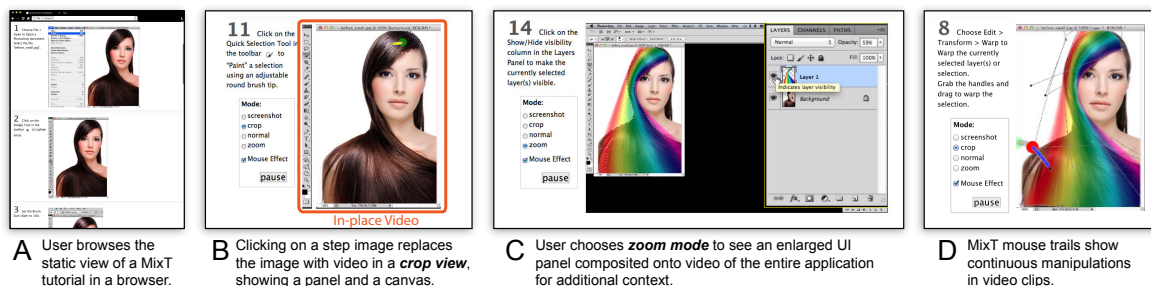


Figure 4.1: MixT generates tutorials that contain static and video information from task demonstrations. Videos are automatically edited and offer different views to highlight the most relevant screen areas for a step. Visualizing mouse movement helps user understand a complex action.

instructions. Offering both text and images, they are well suited for people who prefer to learn by looking at images and those who prefer to learn by reading text [101]. However, it can be difficult for users to understand continuous, complex manipulations such as painting a region, adjusting control points, or rotating a 3D object in static tutorials. In contrast, videos are effective at showing exactly how an application responds to user interaction, but it is hard to navigate back to previous steps or to look ahead in a video timeline [174].

We hypothesize that a combination of static and video instructions can improve users' success in following tutorials. We focus on image-editing software in particular, because it is widely used and has a large collection of tutorials accessible in bookstores (books and magazines) and on the web (e.g., user forums and video sharing sites), but we suspect that our findings are generally applicable to visual editing software. With mixed static and video tutorials, users may effectively learn complicated actions (e.g., applying brush strokes) from tutorial video clips, and quickly access basic actions (e.g., copying a layer) from static text and images.

To test our hypothesis, we carried out a within-subjects study comparing static, video, and mixed media tutorials. 12 participants completed three workflows, one for each format. We found that videos are especially valuable for actions that involve brushing, control point manipulation, and adjustment of continuous parameters. We also found that the availability of video reduces the number of repeated attempts users make to execute a step. The study results led to four design guidelines for mixed media tutorials: 1) offer a scannable overview of steps; 2) include small but legible videos; 3) add visualizations of canvas interactions such as brushing to the videos, and 4) enable users to choose the most appropriate visual representation for each step.

To enable instructors to create mixed media tutorials, we introduce MixT, a system that takes a user demonstration and automatically generates mixed tutorials that show static step-by-step content and also include in-place video clips for each operation (Figure 4.1). MixT generates these materials from screencapture video and recorded traces of application commands and input device events. MixT segments video into steps, applies video compositing techniques to focus on salient screen regions, and highlights canvas interactions through mouse trails. The web-based tutorials give users interactive control over when to see images or videos, and how to render videos. A quantitative

analysis of nine automatically generated MixT tutorials indicates that our algorithms for segmenting videos into steps and detecting salient regions within frames are effective ($<8\%$ error rates). In addition, informal user feedback suggests that MixT tutorials were as effective as manually created tutorials in helping users complete tasks.

In summary, the main contributions of this work include:

- A categorization of the types of user operations for which video is useful.
- A set of design principles for how to embed video in step-by-step tutorials, derived from a formative study.
- A general approach for automatically generating mixed media tutorials from demonstrations, and algorithms for implementing this approach for Adobe Photoshop.
- An evaluation of automatically generated mixed media tutorials.

4.2 Related Work

Previous HCI research on instructional content falls into four main categories: 1) new tutorial formats and interfaces [24, 76, 94, 116, 148, 146, 174]; 2) automated methods for generating tutorials [59, 91, 96, 174]; 3) studies evaluating the effectiveness of different instructional formats [91, 94, 101, 166, 165]; and 4) techniques for searching and analyzing collections of tutorials [72, 126]. Our work on generating and evaluating mixed-media tutorials addresses the first three of these topics. In this section, we describe related work on new tutorials formats and automatic generation of instructional content. The following section discusses previous studies on tutorial effectiveness in the context of our own formative study.

New forms of instructional content. While static step-by-step and video tutorials are the most prevalent forms of instructional content, researchers have been exploring new formats and interfaces for learning materials. Many efforts propose instructional aids that work in conjunction with the target application, including in-application step-by-step wizards [24, 116, 76] and Q&A forums [146], video-based tooltips [94], interactive video tutorials [174], command recommenders [148], and interface facades for mapping commands between applications [176]. Some systems also include features that facilitate navigation within tutorials, such as annotated video timelines [96, 174] and reactive “current step” indicators [76]. Our work introduces an interactive mixed-media tutorial format that combines aspects of both step-by-step and video-based learning aids.

Automatic generation of learning materials. One of our key contributions is a method for automatically generating mixed-media tutorials from user demonstrations. Most existing tutorial generation approaches analyze traces of user interactions through the application or screencast video of the demonstration. Methods that analyze user action traces [91, 59, 96] often capture application-level semantics about specific tools and their purpose, while techniques that analyze screen recordings [18,20,4] use computer vision to identify GUI interactions such as clicking on an icon or button. In MixT, we combine and extend these approaches to create step-by-step tutorials that incorporate text, images, and several formats of video.

4.3 Design Guidelines

Researchers provide different findings on the effectiveness of media formats of software tutorials. Evaluating the instructional potential of videos began in the 1990s. Palmiter, Elkerton [166, 165], and Harrison [101] studied the effect of animated demonstrations on learning and instruction recall. More recently, Grabler et al. [91] compared how users followed book tutorials, videos, and automatically generated static tutorials. Their results showed that automatically generated text and image tutorials outperformed video or book instructions on time and errors. Grossman et al. [94] studied the effectiveness of embedding short (10-25 second) video clips in applications. They found that participants who had access to video-based tooltips were significantly faster in completing tasks than those who viewed static ones.

While these studies suggest that there is still some debate over the trade-offs between step-by-step static and video tutorials, they provide strong support for two key claims: step-by-step tutorials help users make fewer errors by allowing them to work at their own pace, while videos can help provide subtle details of complex interactions that are difficult to represent statically. Based on these findings, we designed a formative user study that investigates whether video clips can be incorporated into a step-by-step framework to help users follow certain types of image-editing tasks within a tutorial.

Formative User Study

Our formative study aims to test the following two hypotheses:

H1. Image manipulation tutorials that mix static images and video clips are more effective than all-static or all-video tutorials.

H2. Users benefit more from seeing video clips instead of static text and images for certain types of commands.

Participants

We recruited 12 participants (5 males and 7 females, aged 20-52), 4 from a campus student design group and 8 from a computer software company, and compensated each with a \$15 gift card for participating. Our tutorials focused on achieving specific tasks in Adobe Photoshop. We recruited participants who had prior expertise with Adobe Photoshop, but who were not expert users. To demonstrate expertise, potential participants first completed an online screening test that asked them to follow a short image manipulation tutorial and submit the resulting file. The selected participants had between 1 and 20 years of experience using Photoshop.

Tasks and Material

The study was based on a within-subject design. We looked through Photoshop books and selected 3 different image manipulation tasks with similar levels of difficulty and complexity (see Figure 4.2; Appendix A shows the complete tutorial of Task 2). Each tutorial comprised 15-20 steps. We

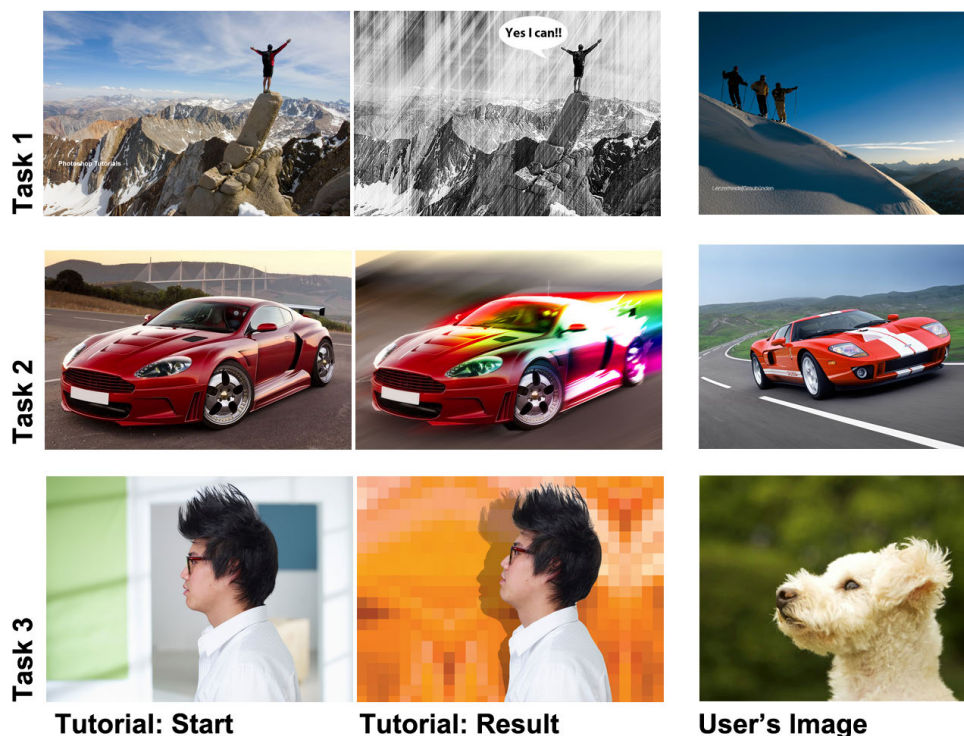


Figure 4.2: In our formative study, participants completed three tutorials with images similar but not identical to the originals.



focused on tutorials that included new, less common features such as the *liquify* tool, *gradient warp* tool, and *puppet warp* tool to increase the chance that participants would encounter unfamiliar tools. For each tutorial, we created three types of presentations: 1) *static* (in HTML format displayed on the screen), 2) *video* (on YouTube with audio narration), and 3) *mixed* (web interface shown in Figure 4.3 without audio). To ensure that different formats presented equivalent information where possible, we first recorded and narrated our video tutorials, then manually generated the static version by writing text instructions based on the narration and annotating and cropping frames of the video. To create mixed tutorials we started with the static tutorials and added the corresponding screencapture video segment for each step. To view the video segment for a step in the mixed tutorial, the user had to click on the image for that step. We scaled these videos to a fixed resolution of 800x500 pixels so that at least 2-3 steps would fit on screen when the videos were expanded. Many online tutorials do not offer full-screen resolution videos; even when high-resolution videos are available, they are hard to use as they force users to continually switch between the video and application windows. We disabled the soundtrack in the mixed tutorial to avoid situations when users only relied on auditory instructions instead of learning from static or video formats.

For each task, participants were given a source image that was distinct, but thematically similar to the image manipulated in the tutorial itself. This study design choice was motivated by the fact that users typically want to transfer the techniques found in tutorials to their own images.

Rapid Movement Effect

How to use Tutorial Builder

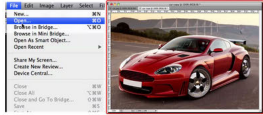
By Adobe CTL; revised by UC Berkeley research team

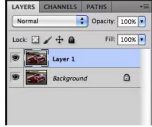
Before

After


1 In this tutorial, we will create a rapid movement effect on a photo of a still car. Choose File > Open to open the desired photo in Photoshop.



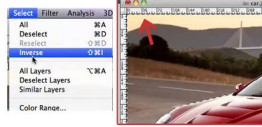
2.1 Duplicate layer with Cmd+J.




2.2 With Quick Select tool, select out the car from the copied layer. You can decrease the brush size (shortcut key: []) for finer details. If you make a mistake, hold down the Option key while brushing to deselect the region.



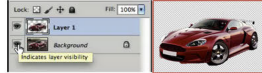
2.3 With the car selected, go to Select > Inverse and press Del. Now the background is removed from this layer.



2.4 Choose Edit > Clear to Delete the selected pixels.



2.5 Click on the eye next to the Background layer to hide and see the difference. Click again to show the background.



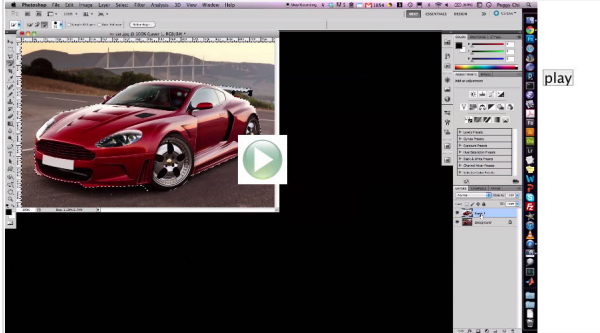


Figure 4.3: In the mixed condition, participants saw an HTML page with static images and text; they could expand each step to view a video of that step (here: step 2.5).

Procedure and Environment

Each session consisted of 1 warm-up task and 3 experimental tasks. The warm-up task was a short 5-step static tutorial. In the 3 experimental tasks the format and task order were randomized. Each 60-minute session was conducted in a lab environment, using computers running Mac OS X, Adobe Photoshop CS5.1 and a web browser (Google Chrome) for viewing tutorials. Each participant was provided with a keyboard and a mouse and was allowed to adjust the equipment setting such as the monitor position and mouse tracking speed during the warm-up task. Photoshop and the web browser were arranged side-by-side on a 30-inch monitor with a resolution of 2048x1280 pixels. During the study, we used screen capture software to record user performance.

Measurement

To evaluate **H1**, we report the number of *errors* and *repeated attempts* that the participants made for each task. While our ultimate goal is skill acquisition and retention, we focus on the pragmatic goal of improving users' success in following tutorials and performing the instructions. We record an *error* if the participant performed a command incorrectly or skipped a step in the tutorial. While

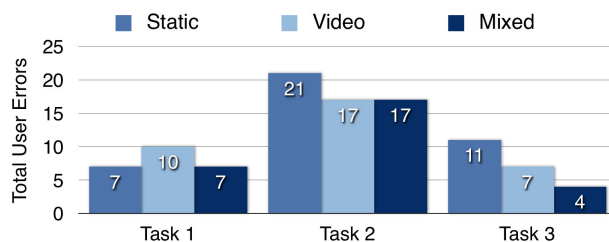


Figure 4.4: Users tied for fewer errors with mixed tutorials.

errors give a sense for the effectiveness of the tutorials, they do not measure the extraneous work users might have to perform when they have trouble understanding the correct outcome of a step. For example, if a user makes an error and then correctly executes several steps before recognizing the problem, we count this as a single error, even though the user must go back to fix the problem and then redo the subsequent steps. In addition, users may select the right command, but be dissatisfied with the result of their image and try again (e.g., redrawing a gradient). In such cases, we record all executions of the same step following the first attempt as a *repeated attempt*. Note that we do not count adjustments of continuous parameters or refinements of selection regions as repeated attempts because in these cases, the user is focusing on a single action rather than repeating a previously executed step. We do count a repeated attempt if the user entirely undoes a step to then retry it.

To evaluate **H2**, we count the number of different users who click on the video for each step in the mixed tutorials. To determine whether some types of commands benefit from videos more than others, we bin each step into one of the following five command categories based on the types of user interaction and UI elements it involves: brushing/drawing, manipulating control points (e.g., mesh-based warping, spline editing), parameter adjustment (e.g. using a slider to change opacity), UI navigation (e.g., switching tools, finding menu items), and layer operations.

We also collect qualitative data by observing how users follow the presented information and obtain additional feedback via 5-point Likert-scale questions (e.g., “The <condition> tutorial was easy to follow.”) and open-ended questions (e.g., “Compared with static tutorials, what were the pros and cons of the mixed media tutorial?”).

Results of Formative Study

Based on the quantitative data and observations from our study, we gained several insights about how users interact with static and video content.

User Performance on Image Editing Tasks

Our analysis of user performance supports **H1**. As Figure 4.4 shows, mixed tutorials resulted in the fewest total number of errors (28 for mixed, 34 for video, 39 for static) across all three tasks and produced an equivalent or fewer number of errors compared to static and video for any given task. In terms of extraneous work, the mixed condition resulted in many fewer repeated attempts

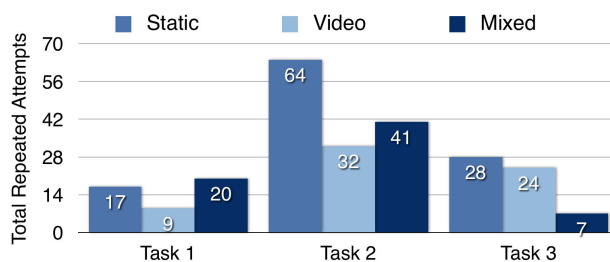


Figure 4.5: In two of three tasks, participants made more repeated attempts at executing steps with static tutorials than with mixed tutorials. Video tutorials had the fewest attempts.

than static tutorials and slightly more than video tutorials (65 for video, 68 for mixed, 109 for static, see Figure 4.5). Although the differences in errors and repeated attempts are not statistically significant—likely due to the small study size and differences between the tasks—the overall trends suggest that mixed tutorials help users make fewer errors and do less extraneous work compared to static and video tutorials.

In addition to these quantitative results, we observed a few specific behaviors that had an impact on user performance:

The scannable nature of the static and mixed tutorial formats helped users follow along and avoid missing steps that might result in errors. In the video condition, users were more likely to accidentally skip steps because they were working at a different pace than the video. They also had trouble finding previous steps when trying to identify the source of an error.

In the static condition, users had trouble understanding how to perform steps that involve complex or unfamiliar UI elements and interactions. As we discuss in the following subsection, these were often the same steps where users decided to play the videos in the mixed condition. With only static text and images, users often made errors or had to repeat such steps multiple times.

Participants used the video clips in the mixed condition in a few different ways. Some users played the video *before* attempting the step to familiarize themselves with the relevant UI elements and interactions. In some cases, users also played the video at the same time as they performed the action, which corresponds to what Palmiter and Elkerton described as “mimicking actions” [166]. We suspect both of these behaviors helped reduce errors, especially for complex or unfamiliar steps. In addition, several users also played the video *after* completing a step, as a way to confirm that they had performed the step correctly and “debug” what went wrong if they made an error. This confirmation behavior helped reduce repeated attempts by making it easier to recognize and fix errors sooner.

In some cases, users had trouble seeing all of the relevant details in the mixed videos because the videos were scaled down to 800x500 pixels. For example, when using the *puppet warp* tool, users missed that dragging in the vicinity of a control point (instead of on top of the control point) initiated a control wheel for a rotation rather than a translation maneuver. Although participants neither complained about not being able to resize the video in the MixT condition nor chose full-screen mode in the video condition, they explained that they would hope to clearly see the key part of the demonstration video.

Command Type	L_V : Likelihood of a video view	Steps of this type in corpus
Brushing, Drawing (e.g., Gradient, Brush tool)	35.4%	12
Manipulating Control Points (e.g., Warping)	25.0%	9
Parameter Adjustment (e.g., Levels, Curves)	19.4%	9
UI Navigation (e.g., finding a submenu)	5.4%	37
Layer operations (e.g, Duplicate Layer)	0%	13

Table 4.1: Participants watched videos most often for brushing, control point manipulation, and parameter adjustments.

Which Commands Led to Video Views?

Our analysis of which mixed steps prompted users to click on the corresponding videos suggests that users did indeed find the video clips more beneficial for some types of steps over others (**H2**). To compute the *likelihood of a video view* (L_V) for the five command categories described earlier, we first determine L_V for each individual step (across all three mixed tutorials) by computing the fraction of users who clicked to view the video for that step, and then we average likelihoods across all the steps in each command category.

As Table 4.1 shows, L_V is highest for steps that involve brushing/drawing, manipulating control points and parameter adjustments. Based on our observations, videos help users perform the first two types of commands (brushing/drawing and manipulating control points) by explicitly demonstrating the necessary mouse movements rather than requiring the user to infer what to do from text and images alone. In addition, we noticed that some participants used the video clips to determine how precise they needed to be for certain brushing or selection tasks, which is hard to convey with a static representation. Text descriptions such as “rough” and “detailed” are relative and can be interpreted differently by individuals, but seeing how much time the demonstrator devotes to the task provides a much clearer estimate of the required precision for that task. As for parameter adjustments, users may be relying on the videos to provide some context about the range of visual effects the relevant parameters cover in order to determine what parameter values to use for their own image. Unlike static images, which only show the final parameter values and resulting effect, video demonstrations often show how the canvas changes as continuous parameters are updated (often via sliders), which may give users a better sense for the desired outcome of the step.

User Preferences for Tutorial Types

The results of our questionnaire show that while participants had varying opinions on the static and video tutorials, all users strongly agreed that the mixed tutorial was easy to follow. Participants had difficulty finding the tools that the static tutorials referenced and remarked that there were not enough visuals. For full-length videos, participants disliked having to pause the video to complete each step. For the mixed tutorial, half the participants found that video was the most useful among

different media components for understanding a step instruction. One participant acknowledged that because the mixed tutorial allowed him to “break down the process into simple steps,” he was able to easily find the point where he had made a mistake. Another user explained that videos would be most helpful if the tasks were more advanced and in-depth. On the other hand, users had different preferences within a task. Expertise could be tool-specific: users might find static instructions sufficient for one set of operations (e.g., duplicating layers), and needed to watch videos for another set they were less familiar with (e.g., the *puppet warp* tool). Overall, these responses suggest that users appreciated being able to choose from static and video features in mixed tutorials.

Design Guidelines

Based on the findings from our formative study, we propose four design guidelines for creating effective mixed media tutorials that combine text, images and videos.

Scannable steps. Scannable steps provide valuable context and facilitate navigation within tutorials. To leverage these benefits, videos in mixed tutorials should be presented in a format that supports scanning.

Small but legible videos. To make mixed media tutorials scannable and enable users to work with the tutorial and their application side-by-side, the videos for individual steps should use the minimum amount of screen real estate while still being legible. Ideally, videos should clearly depict the most important portions of the UI for each step (e.g., dialog boxes, panels, or canvas) while hiding or deemphasizing less relevant regions.

Visualize mouse movement. Our study indicates that videos are most useful for steps that involve brushing, drawing and manipulating control points, but even in videos, it can be difficult to see the exact motion or path of the mouse during such interactions. Visualizing mouse movement and events helps viewers understand the relevant spatio-temporal characteristics of the demonstration.

Give control to the user. Our observations of user behavior suggest that expertise and familiarity with the specific tools or interactions in a tutorial is likely to have an impact on which instructional format (static or video) is best for a given user. Videos help users understand, confirm and debug steps with unfamiliar tools, while static images and text are quicker and easier to skim. Thus, mixed tutorials should let users choose the most appropriate format at the granularity of individual steps.

4.4 Computer-Generated Mixed Media Tutorials

The benefits of mixed media tutorials are unlikely to be realized if creating such materials is too tedious, time-consuming, or if it requires more expertise than creating other tutorial formats. To lower the authoring barrier, we designed MixT, a system that automatically generates mixed media tutorials from user demonstrations. While the MixT architecture can apply to different media creation applications, our current implementation is specific to creating interactive tutorials for Adobe Photoshop.

Overview

Tutorial Format

MixT generates HTML tutorials with embedded videos that follow the design guidelines identified in our study. By default, our interface presents a textual description and screenshot for each step, just like a standard static tutorial (see Figure 4.1A). Clicking on the screenshot replaces the static image with a video player that plays the segment of the original demonstration that corresponds to the written step instructions. For example, a screenshot of a layer panel enhances the instruction “*Select Soft Light from the drop-down menu for Blend Mode,*” and the corresponding video clip shows continuous mouse action to the menu, expanding the drop-down menu, moving down to click on the feature, and shows the canvas change. By presenting steps as text and images with video clips that are accessible on demand, MixT tutorials retain the scannability of static tutorials while giving users the option of static- or video-based instruction at each step. To ensure that steps remain scannable and that the tutorial can still be viewed alongside the image editing application (without window switching), we scale each in-place video to at most 700 pixels wide and display text instructions on the left.

Video Playback Options

The MixT video player gives users additional control over the format of playback. Three different modes (normal, zoom, and crop) each emphasize different types of information (Figure 4.6). In addition, users can display a pointer trace visualization to clarify the path of mouse interactions.

Normal mode shows the entire application window (Figure 4.6A). This mode preserves all context, but because MixT scales videos down to at most 700x440 pixels positioned next to the text instructions, it may be hard to see precise manipulation or small widgets or handles in the UI.

Zoom mode also shows the entire application window, but performs a non-uniform enlargement of specific UI regions. In particular, the application area being manipulated (e.g., a menu, dialog, or the canvas) is enlarged to fill the full height of the frame and composited on top of the original video in another video layer (Figure 4.6B). If a dialog also modifies pixels on the canvas, both areas are enlarged and positioned such that they do not overlap. This video composition effectively creates

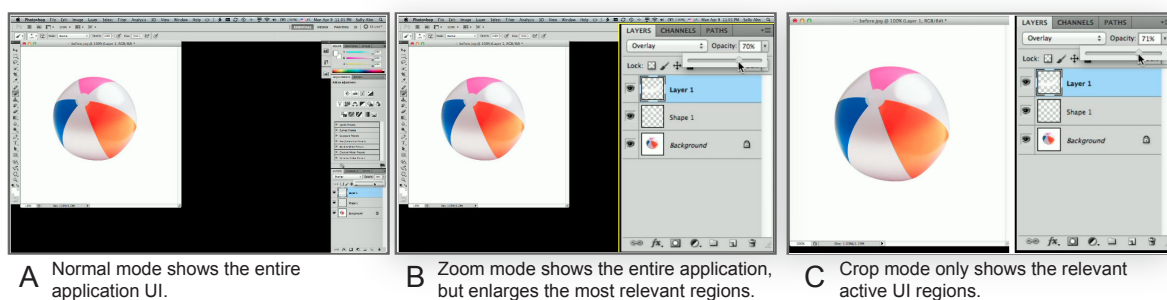


Figure 4.6: MixT offers three video playback options: Normal mode (A), zoom mode (B) and crop mode (C).

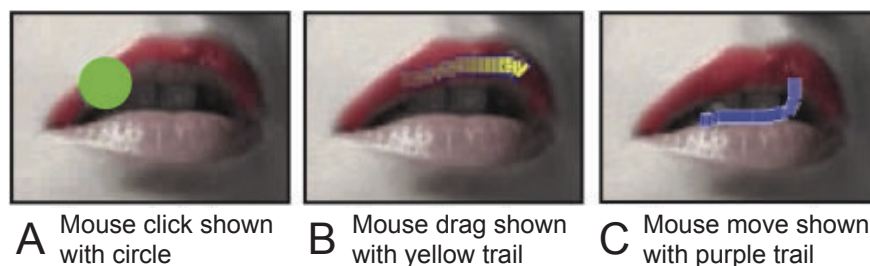


Figure 4.7: Mouse visualization distinguishes moving and dragging.

a focus-plus-context view that makes important regions easier to see at a given video resolution [80]. Commercial screencasting software commonly includes a pan-and-zoom technique to make interactions legible in small videos. However, testing early prototypes of MixT suggested that such a technique is not appropriate for brief, single-step videos, as it is hard to establish application context in such short video segments.

Crop mode does not show the entire application — it only shows the currently active area (a tool bar, dialog, main menu, or a panel) and the canvas if being changed (Figure 4.6C). This offers the unique benefit of showing both a user interface manipulation (e.g., moving a layer opacity slider), and the effect on the image (e.g., parts of the image becoming transparent), while minimizing all other visual distractions. Like the zoom mode, cropped videos are very compact since only the relevant portions of the UI are shown.

Mouse visualization: To help video viewers understand interactions with the canvas, MixT can render a trace visualization of the mouse (Figure 4.7). These traces show a fading path of the most recent positions of the cursor and encode mouse state using color: click events are shown in green (mouse down) and red (mouse up), while mouse *movement* events are shown in purple, and mouse *dragging* events are shown in yellow. Commercial screencasting software also includes mouse visualization techniques. However, they are usually limited to clicking, and the visualizations are typically rendered into the final video. In contrast, MixT emphasizes dragging because such interactions are especially relevant for image manipulation. Furthermore, our trace visualizations can be toggled on and off interactively in real-time. By default, the visualizations are enabled for all steps. However, users can change this behavior through an option in the video player.

Automatic Generation Pipeline

To generate mixed media tutorials, tools can either help authors create new tutorials (e.g., [91]), or they can reformat existing video tutorials into step-by-step videos (e.g., [174]). Our system adopts the former approach and extends Grabler et al.’s system for generating static tutorials from demonstration to include videos [91]. Producing mixed media tutorials requires three steps (Figure 4.8). First, MixT captures an application command log, screencast video and an input device event log and synchronizes them. Second, MixT generates appropriate media files and descriptions: it transforms the log into text instructions, segments the video into steps, and identifies active UI regions, which inform the different video playback modes. Finally, MixT composes the text, images,

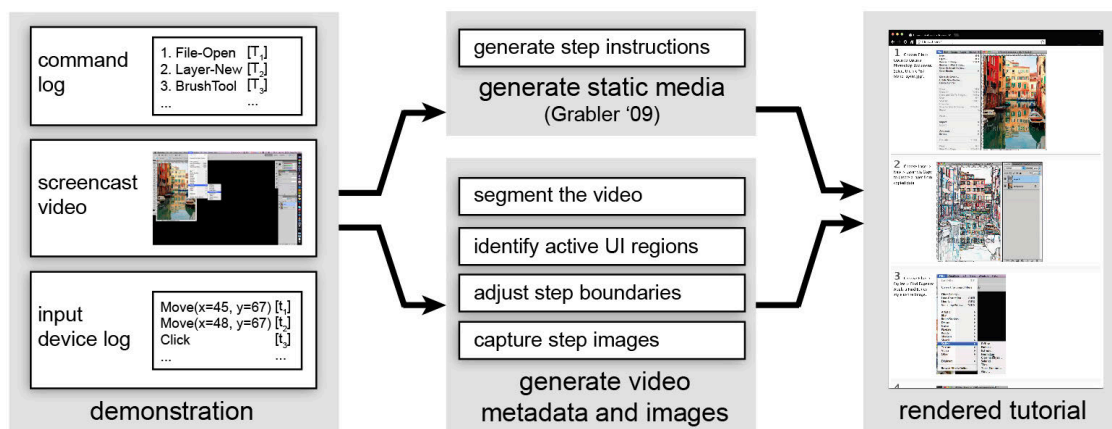


Figure 4.8: MixT generates tutorials from video and log files.

and video into one document and adds mouse interactions as visualizations on top of the videos. For each step in the tutorial, MixT produces the three video formats and one representative step image.

Recording the Demonstration

MixT records three different time-synchronized data streams during a demonstration: a history of executed application commands; a trace of mouse events; and screencapture video of the entire application interface. To capture application commands, such as opening a file, selecting a region, or hiding a layer, we use Tutorial Builder², a freely available Photoshop plug-in that records commands and transforms them into text instructions through text templates. To synchronize Tutorial Builder output with the screencast video and mouse streams, we timestamp the command log during the user demonstration. We obtain mouse event traces on Apple's OS X operating system through the Event Taps³ API, which observes system-wide input events. We record full-screen video with the commercial Camtasia application⁴.

Generating Video Metadata and Step Images

After acquiring the time-synchronized data, there are three technical challenges:

1) Segmenting the video into steps. MixT segments the screencapture video into individual steps based on the timestamps for individual commands in the command log. We map each command Cmd_i with timestamp T_i to a video segment that starts at T_i and ends at T_{i+1} .

2) Identifying active UI regions for zooming and cropping. MixT uses zooming and cropped side-by-side views to preserve legibility for videos at small video frame sizes. To create these specialized views, MixT needs metadata describing the relevant pixel coordinates for each step.

² <http://labs.adobe.com/technologies/tutorialbuilder/>

³ <http://developer.apple.com/library/mac/#documentation/Carbon/Reference/QuartzEventServicesRef/Reference/reference.html>

⁴ <http://www.techsmith.com/camtasia.html>

Each command in the command log contains information about the logical UI regions that are involved (e.g., the toolbar, a dialog box, or the canvas). However, many commands can be invoked in multiple ways. For example, the *New-Layer* command can be accessed through the application menu, or an icon on the Layer Palette. Therefore, MixT must find and select the correct UI regions from a set of candidates. MixT first uses pixel-based template matching [174] to locate these areas on the screen. MixT then identifies the active UI region by inspecting the mouse event log to see which candidate region received a mouse click at the recorded timestamp T_i of Cmd_i . If there are no mouse clicks detected (e.g., a command invoked via keyboard shortcut), MixT treats the entire frame as the active UI region to ensure the application response is visible in the video.

3) Adjusting segmentation boundaries. While segmenting the video based on the command log timestamps produces a reasonable rough alignment between steps and video segments, there are some cases where a command is recorded after important UI events have taken place. One typical case is menu navigation. For example, to use the *Replace Color* operation, the user moves from the *Image* menu through the *Adjustment* sub menu to the *Replace Color* option. The entire traversal sequence is relevant information as it explains how to reach the menu item. However, the command log only records Replace Color when the operation is invoked, which means that a video segment generated based solely on command timestamps would not include the menu traversal.

MixT adjusts step boundaries by leveraging template matching and the mouse event log. To compute the start time of the tutorial step for Cmd_i , our algorithm starts at the command timestamp T_i , looks backward for all mouse clicks that occur within any visible candidate UI region for the command (e.g., menus, panels, toolbar) between T_i and the recorded T_{i-1} of the previous command, and sets the adjusted start time of the step T'_i to the time of the earliest mouse click. Adjusting step boundaries in this manner ensures that the step video clip shows all the relevant actions associated with the command.

4) Capturing screenshots and after images. Finally, in order to generate a representative static image for each step, MixT selects the most informative frame of UI interaction from the video. For example, if the step changes the layer blend mode, we show an image of the expanded blend mode menu where the appropriate option is highlighted. If the command involves a dialog box, we show the final frame when the dialog is visible. To do so, MixT selects the last frame that includes any candidate UI region within the duration of the step $[T'_i, T'_{i+1}]$ and crops the frame to the UI region to produce the representative static image. Furthermore, MixT also captures the state of the canvas from the last frame of the step as an “after” image that helps viewers understand how the canvas was affected.

The MixT video analysis is implemented in MATLAB. In our dataset of nine test tutorials (see Table 4.2), the average duration of a step video is 12.4 seconds – our analysis takes an average of 6.3 seconds to process each step.

Composing the Mixed Tutorial

The step-by-step mixed-media tutorial with text instructions, structured screencast video, and step images is composed and presented in a web interface. The MixT tutorial viewer is implemented with standard Web technologies (HTML5, CSS3, and JavaScript) so it can be easily deployed

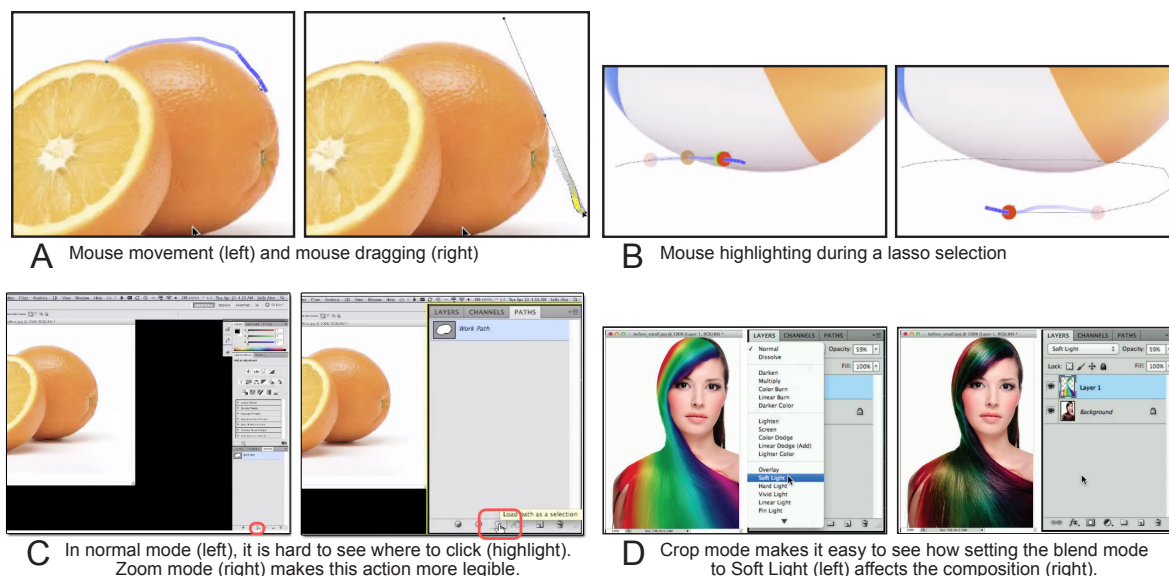


Figure 4.9: Automatically-generated MixT results.

online. In particular, all video compositing and matting is done in real-time using HTML5 video. Just-in-time compositing enables users to change viewing options *on the fly* without pre-rendering multiple video presentations. For example, during the video playback users can choose to show the enhanced mouse movement inside its frame for additional detail of a brush stroke, or disable the effect and focus on the incremental change on a canvas in real-time.

Results

We gathered nine different tutorials and recorded working through each tutorial in Photoshop. We then used MixT to automatically generate mixed media tutorials from these demonstrations. This section describes this corpus. The following section then evaluates the generated tutorials quantitatively and qualitatively.

Our tutorials came from both online and book sources: two from “Adobe Photoshop CS5 Classroom in a Book,” five from photoshopstar.com, one from makeuseof.com, and one from icanbecreative.com. All are popular resources for Photoshop learners. Two of the tutorials were also used in the formative user study. The selected tutorials had a total of 165 steps and covered all five command types: they contained 15 brushing/drawing operations, 14 control point manipulations, 30 parameter adjustments, 67 UI navigations, and 39 layer operations. The demonstrations were recorded on three different laptops running Photoshop in full screen with native resolutions of 1680x1050, 1440x900, and 1280x800 pixels.

Overall, the MixT tutorials that we generated exhibit the desired characteristics that we identified in our formative study: scannable steps, small but legible videos, visualized mouse operations, and user control over presentation format. We highlight some interesting results generated by MixT and

also refer readers to the provided video figure, which has additional examples.

Scannability. The step-by-step layout of our tutorials makes them easy to scan. For example, at a glance, we see that the tutorial “Turning an Image into an Old Photo” involves several adjustment layer operations, while the tutorial “Creating Artistic Effects” involves more parameter adjustments and brushing commands.

Mouse visualization. Our mouse visualizations help clarify several interactions. They clearly communicate the difference between clicking and dragging, a distinction that is fundamental to operations such as path manipulation but hard to glean from screen capture video. For example, Figure 4.9A shows the difference between moving around the contour of an object without drawing a path (left), and dragging a Bézier handle to adjust a path segment (right). Mouse trails and click markers were also useful for showing the trajectory of lasso selections (Figure 4.9B).

Zoom and crop modes. For many steps, the zoom and crop videos offer clear legibility benefits over the normal video mode. In our corpus, zoom mode was especially valuable for highlighting actions on small buttons that occurred near the frame boundaries, e.g., in the layers palette (Figure 4.9C right). Such operations are easy to miss in a normal, scaled video (Figure 4.9C left). Crop mode was useful in showing the effect that parameter selection has on the canvas. Figure 4.9D shows two successive frames that illustrate how changing a layer’s blending mode affects the image. Enlarging the canvas in these modes also helps users see the details of effects, such as applying the *eraser tool* on the canvas to enhance the underlying layer (Figure 4.7).

4.5 Evaluation

To evaluate MixT, we measure the performance of our automatic tutorial generation pipeline and gather user feedback on the effectiveness of the resulting MixT tutorials.

Expert Inspection of Generated Results

We examined the segmented and cropped videos for each step of our nine converted tutorials and recorded the following errors. If a clip does not include all actions of the current step, we record a *segmentation error*. If the screenshots or zoom/crop videos do not show the appropriate application regions, we record a *region finding error*; if the system fails to identify the active region and shows the overall UI instead, we record a *region finding miss*; if they show some relevant regions but omit others, we record an *incomplete region*.

Table 4.2 shows the results of our inspection. On average, MixT correctly segmented steps around 92% and found relevant regions of complete views 92% of the time. These error rates suggest that our automatic generation pipeline performs reasonably well for a variety of real-world tutorials, but there is room to improve our segmentation and region-finding accuracy.

Task (time)	Steps	Segmentation Error	Region Error	Region Miss	Incomplete Region
T1 (2'30)	19	15.8%	0.0%	0.0%	10.5%
T2 (3'19)	21	19.0%	0.0%	14.3%	0.0%
T3 (6'02)	30	3.3%	3.3%	0.0%	10.0%
T4 (2'44)	16	12.5%	0.0%	0.0%	0.0%
T5 (4'37)	9	0.0%	0.0%	0.0%	0.0%
T6 (6'10)	21	4.8%	0.0%	0.0%	0.0%
T7 (2'58)	21	4.8%	0.0%	0.0%	4.8%
T8 (3'59)	16	5.6%	11.1%	5.6%	5.6%
T9 (1'41)	12	0.0%	0.0%	0.0%	8.3%
AVG (3'46)	18	7.3%	1.6%	2.2%	4.4%

Table 4.2: Error rates for automatically generated tutorials.

User Experiences: Working with MixT Tutorials

We conducted a small user evaluation with four participants (2 males and 2 females, aged 25-29, with 5-12 years of Photoshop experience) to gather feedback on the usability of our MixT tutorials. We selected four of our nine evaluation tutorials with features such as the *brush* tool, *pen* tool, *puppet warp* tool, and *gradient warp* tool — commands that led to many video views in our formative study (Table 4.1). These test tutorials were generated with an earlier version of MixT that did not use the mouse event log to refine the step segmentation and active UI region finding as described earlier. This previous implementation relied solely on the command log and computer vision, which resulted in lower segmentation accuracy (84%) and region-finding accuracy (90%).

Participants were introduced to the MixT system and then asked to work through the set of tutorials, analogously to our formative study. We asked participants to comment on their process using the think-aloud method, and afterwards asked open-ended questions to elicit additional detail.

Successes

The participants found the same benefits in automatically-generated MixT tutorials as earlier participants found in manually-created tutorials. Participants commented that videos helped them understand steps that were complicated or text that was ambiguous or did not contain explanations why certain steps were taken: “*Videos were convenient when trying to get a sense of a complex operation.*” / “*I tended to watch the videos when the text wasn’t clear.*” Examples included making a selection from a path (1.75 views per user) and the *puppet warp* tool (2.75 views). Note that a clip might be viewed more than once by a single user. Participants also watched videos to confirm their results because “*The photo/screenshot (...) wasn’t as actively helpful in guiding what I needed to do or confirming that I was doing the right thing.*” Multiple participants commented positively on the utility of automatically segmented videos that focus on short step clips about the task at hand: “*I didn’t have to sit through 5 mins of intro to get a video description of the task I was interested in*” / “*What I liked the most about the mixed tutorial was the ability to only watch short segments of video that was relevant. Often with video tutorials I find myself sitting and waiting for the content that I need. Because of this, I tend to avoid video tutorials in favor of text. The mixed tutorial was a nice way to achieving the best of both worlds.*”

Shortcomings

Our participants identified useful suggestions for improving our tutorial design. Currently, static images do not provide sufficient *information scent* about the contents of the video – it was hard to judge how long each video was and whether there were remaining important actions in a clip. Therefore, participants sometimes skipped important information that resulted in editing errors (e.g., not adjusting the pose after placing pins using the *puppet warp* tool). In addition, the minimal play/pause interface was deemed insufficient: *“Navigating the videos was difficult [...] It was also hard to go back in the video to observe missed steps. Adding standard playback controls might help.”* One approach to remedy this would be to analyze the video and provide thumbnail frames of the video clip that highlight the clip’s content and length.

As mentioned earlier, our automatic tutorial generation pipeline computes the correct video segments and finds the right spatial regions to highlight for most steps. However, the few segmentation and region finding errors that users encountered sometimes caused important information to be hidden in crop or zoom mode. As a result, participants referred back to the normal video mode more often than we expected, even though the crop and zoom videos were typically more legible. For the 41 video segments that were watched, the average view counts per step were 0.66 for crop mode, 0.24 for zoom mode, and 1.8 for normal mode. Participants commented on the impact of segmentation errors: *“Sometimes the video doesn’t line up — and you have to go to the step before to see what’s going on.”* We consider this as an opportunity to include the tutorial authors in the loop to modify computer-generated tutorials.

4.6 Conclusion

This chapter introduced MixT, a system that automatically generates step-by-step mixed media tutorials from user demonstrations. We motivated the design of MixT through a formative study that suggested that step videos help users understand complex direct manipulation operations. MixT’s architecture uses a command log, an input device log, and screencapture video to generate tutorials. It applies video compositing techniques to focus on salient information, and highlights interactions through mouse trails. Our informal evaluation suggests that automatically generated MixT tutorials were as effective in helping users complete tasks as tutorials that were created manually.

The current MixT implementation has some important limitations that should be addressed in future work. One missing yet interesting component is the audio content, such as a tutorial author’s narration in the video demonstrations. Spoken explanations of the demonstrated actions can help viewers understand the rationale behind a sequence of steps. However, narration and interactions may not always occur in synchrony and it is an open problem to segment combined audio and video tracks appropriately into steps. MixT also does not provide opportunities for the tutorial creator to edit a demonstration. To maximize the benefits of mixed media tutorials, we are interested in exploring ways to provide an editing interface for tutorial authors to easily examine and modify automatic results, and to add annotations that can provide rationale in a lightweight way before sharing their demonstrations.

Chapter 5

DemoWiz: Visualization for Software Demonstration

Showing a live software demonstration during a talk can be engaging, but it is often not easy: presenters may struggle with (or worry about) unexpected software crashes and encounter issues such as mismatched screen resolutions or faulty network connectivity. Furthermore, it can be difficult to recall the steps to show while talking and operating the system all at the same time. An alternative is to present with pre-recorded screencast videos. It is, however, challenging to precisely match the narration to the video when using existing video players.

In this chapter, we introduce DemoWiz¹, a video presentation system that provides an increased awareness of upcoming actions through glanceable visualizations. DemoWiz supports better control of timing by overlaying visual cues and enabling lightweight editing. A user study shows that our design significantly improves the presenters' perceived ease of narration and timing compared to a system without visualizations that was similar to a standard playback control. Furthermore, nine (out of ten) participants preferred DemoWiz over the standard playback control with the last expressing no preference.

5.1 Introduction

Performing a software demonstration can be an effective way to communicate with the audience during a live presentation. By illustrating actions within a working system, presenters can guide the audience through an interaction flow and show results in real time. However, it is not always easy to perform an effective live demo. Problems such as software crashes, network connectivity issues, and configuration changes (e.g., screen resolution) may break a demonstration. Furthermore, talking while interacting with the system creates a high cognitive load on presenters. In addition, the stress of public speaking, especially during a high-stakes presentation, makes it difficult for presenters to deliver effective messages in a timely manner without forgetting to cover a set of core values of the system. An alternative is to present with pre-recorded screencast videos that capture

¹ This work was published at CHI 2014 [48].

the correct flow and information. Even though technical problems are less likely to occur with a video, it is challenging for presenters to talk over a video with appropriate timing because they have to mainly rely on their memories for the sequence and timing of interactions. Such a “canned” demo can often result in a less understandable or engaging presentation when a video is not tightly prepared to attract the audience’s attention to anticipate the results [53].

The presenter view in PowerPoint or Keynote attempts to help presenters during slide show presentations by showing notes along with an upcoming slide. A teleprompter, commonly used for news programs or political speeches, prompts presenters with an electronic visual text of a speech or script. With this, speakers can appear to be speaking spontaneously as they look at the audience while reading the script. Inspired by these tools, we built DemoWiz (see Figure 5.1), a system that assists presenters in giving software demonstrations with a screencast demo video during a live presentation. DemoWiz augments a screencast video with visualizations, enabling presenters to *anticipate* the video content rather than react to it; overlaying glyphs to guide presenters to the next action along with the time remaining before the action occurs.

DemoWiz supports the entire authoring process from capturing a screencast video; to rehearsing it and adjusting timings; to performing live presentation of the demo. During the recording phase, DemoWiz captures the screen pixels and logs input events, including event types and locations with timestamps. This event information is then processed and provided to presenters in the form of an adjustable timeline of events. During the rehearsal phase, presenters can speed up or slow down specific segments while navigating through the video recording using the timeline. In addition, they can add *pause markers* and short *text notes*. During the presentation, similar to current presentation tools like PowerPoint and Keynote, DemoWiz shows two views—one for the presenter and the other for the audience. The *Presenter View* is augmented with timed notes and a visualization of the captured events to help presenters synchronize their narration with the video.

To explore the effectiveness of the DemoWiz system, we performed a user study, comparing it with a version similar to a conventional video player. Our results show that, with DemoWiz, participants anticipated upcoming actions better and rated themselves as having narrated the video better. Moreover, 9 out of 10 participants preferred DemoWiz to a system without visualizations.

The contributions of this work are:

- An interactive video playback interface to help presenters control demo videos during a live presentation. It is combined with visual augmentation of screencast videos to enable presenters to anticipate upcoming actions and to be better aware of timing for narration.
- A lightweight workflow for presenters to record, rehearse and edit, and present demo videos. To support automatic video segmentation, we employ a hybrid approach to combine screencast videos and input event logs.
- Evaluation of the overall effectiveness of DemoWiz, incorporating visualizations into the presenter view of a video, across the workflow.

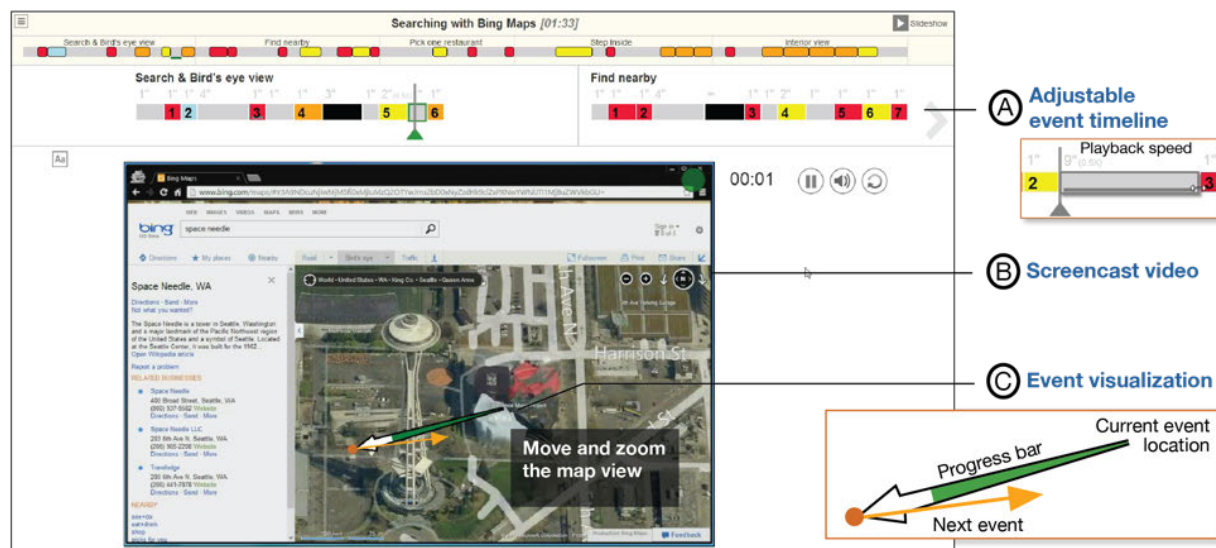


Figure 5.1: DemoWiz visualizes input events in a screencast video to help presenters anticipate the upcoming event for narrating a software demonstration in a live presentation.

5.2 Related Work

Modern presentation tools have supported embedding video recordings and animation. Recent research has proposed advanced designs for content creation and navigation beyond simple slideshow composition, including: tools that help presenters compose content in a large canvas [88] as a path [138] or a directed graph [194] derived from zoomable graphical interfaces [22]; structure slides using markup languages [70] or sketching [137]; and define animation programmatically [224]. There has also been work on analyzing slide content for search and reuse [25, 188] and comparing revisions in a design process [69]. Our work shares similar goals of structuring a presentation based on event inputs that can be navigated and edited. However, we focus more on presentation enhancements of video content specifically for software demonstrations rather than on the authoring experience of the presentation itself.

Research on presenting information that can be perceived at a glance [149] helps presenters recall the content during a presentation, such as a callout to show finer resolution of an overall view [20]. Closely related, Time Aura provides ambient cues of a pile of slides using colors and a timeline for pacing [144]. Recent research shows that people like to have better control of the presentation even though it requires more effort [133], and earlier studies suggest that designing an integrated presentation tool for complicated tasks could be challenging [112]. These findings inspired our design on revealing content of a demo video with information that can be perceived with minimum attention.

5.3 Design Guidelines

To motivate and inform the design of a tool to support live presentations, we collected preferences for software demonstrations using an online survey. We describe the three design goals derived from the survey results.

Understanding Demo Preferences

To understand both presenters' and audiences' preferences for performing and viewing system demonstrations, we conducted an online survey in a software company and a university research lab. Our goal was to collect people's feedback on giving and seeing software demonstrations during live presentations. We received 73 responses from researchers, graduate students, software engineers, and designers. Their main research areas include human-computer interaction (64.4%), software engineering (21.9%), and machine learning (20.6%); 66.7% were male. Among all the respondents, 35.6% indicated that they were very experienced at giving software demos to the audience during a live presentation; 46.6% had demoed at least once; 13.7% had not demoed but attended talks that showed a software demonstration.

We asked respondents who had demo experience ($N = 60$) how they preferred to perform a demo. Their answers were: a live demo (25 out of 60), pre-recorded videos (15), a mixed format of a live demo and videos (12), static screenshots (4), and other (4). In Table 5.1, we list the top 2-3 reasons for their preferences. Giving a *live demo* can be more engaging with a working system and match the audience's interests, but presenters can encounter unexpected problems and forget to show important features within a given time constraint. On the other hand, presenting with a *demo video* avoids such problems by extracting the most important parts, and can allow visual highlighting (labeling or zooming), but can be less engaging. In addition, it is hard to narrate.

We were also interested in reactions as an audience member. For respondents who had seen software demos ($N = 70$), we asked how they preferred to see the demonstration performed. We found a slightly different preference: a live demo (36 out of 70), a mixed format of a live demo and videos (24), pre-recorded videos (7), and other (3). However, the reasons were well aligned with presenters' concerns. A *live demo* shows a working system and can be more engaging, but the audi-

	Presenters		Audience	
	Advantages	Disadvantages	Advantages	Disadvantages
Live Demo	<ul style="list-style-type: none"> • More engaging (88.3%) • Show a working system (86.7%) • Easy to adjust a demo based on audience's interests (45%) 	<ul style="list-style-type: none"> • May encounter unexpected system problems (86.4%) • May forget to show important features (33.9%) • Hard to control time (35.6%) 	<ul style="list-style-type: none"> • Show that it's a working system (97.1%) • More engaging (72.9%) 	<ul style="list-style-type: none"> • May need to wait for problems to be solved (78.6%) • Presenters may end up rambling (37.1%)
Demo Video	<ul style="list-style-type: none"> • Avoid system problems (95%) • Work with a partially working system or a mockup (51.7%) • Can edit to remove mistakes or add highlights (51.7%) 	<ul style="list-style-type: none"> • Less engaging (57.6%) • Hard to match their narration to the video content (30.5%) 	<ul style="list-style-type: none"> • Avoid problems (81.4%) • Show the most important parts with visual highlights (60%) • Work with a partially working system or a mockup (45.7%) 	<ul style="list-style-type: none"> • Hard to tell which parts are real (62.9%) • Want to see how the actual system works (44.3%) • Less engaging (37.1%)

Table 5.1: Survey of software demonstration preferences from presenters' ($N=60$) and audience's ($N=70$) point of views.

ence might need to wait for system problems to be resolved or sometimes see presenters rambling. A *demo video* can show the most important parts, sometimes assisted by visual highlighting, but it can be hard to tell which parts of a demo are real, and can be less engaging to the audience.

Design Goals

From the survey results, we understand that giving a live demo is often more preferable than showing demo videos. However, we cannot, in general, address some of the main concerns with giving a live demo – that is, stability of the software system and variations in the presentation environment which can cause the demo to fail. Therefore, we instead aim to address some of the drawbacks with demo videos while preserving their advantages. More specifically, our goal is to make demo videos *more engaging* by assisting presenters in adjusting their narration to guide the audience through the material. We describe our design goals to support more effective demo video presentations.

G1. Show what’s coming next, where and when it will occur.. To engage the audience with the demonstration, it is important for presenters to guide the audience’s attention to the right place at the right time. To do so, presenters should be fully aware of upcoming actions – specifically *what* actions will happen, *where* they will occur on the screen, and *when* they will happen.

G2. Minimize required attention or interpretation.. While it is our desire to help presenters understand and anticipate impending events, we should not overburden a presenter who is already narrating a specific set of talking points. As a tradeoff between providing more information and minimizing cognitive load, any augmentation of the video needs to be offered in a glanceable fashion, i.e., information can be interpreted quickly and without the presenter’s full attention.

G3. Support light-weight editing during rehearsal.. Different presentations may require more or less extensive explanations, and when first recording a demo video, it may not be possible to perform the demo at the same rate necessary for a live presentation (e.g., typing can be difficult or system response times may be variable). In addition, it should be easy to review, practice, and modify the pace for a particular presentation. For all these reasons, lightweight editing and rehearsal are necessary. Using these principles as a guiding rubric for our design, we iterated on several versions of the DemoWiz system.

5.4 Computer-Generated Visualization

Augmented Workflow

For presenters to narrate “live” over a video recording, we propose augmenting a typical workflow from capturing a screencast video; to rehearsing it and adjusting timings; and finally to live presentation of the demo video (Figure 5.2).

DemoWiz first captures a screencast video and input events during a software demonstration from a user-defined rectangular region. Once the recording is done, DemoWiz analyzes the low-level event stream and transforms it into higher-level events such as mouse clicks, double-clicks, and drags. DemoWiz then allows presenters to edit the timing and notes while practicing their

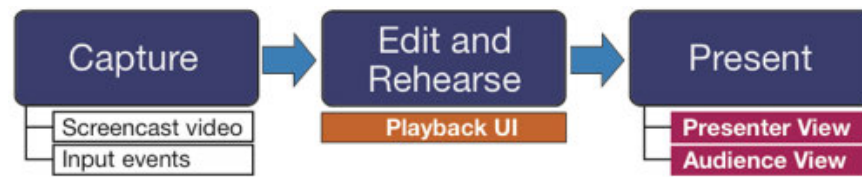


Figure 5.2: DemoWiz workflow: Presenters capture a software demonstration, edit the video recording while rehearsing with our playback UI, and present the edited video to the audience using a presenter view.

presentations with the presenter view equipped with an adjustable event timeline (Figure 5.1A). Finally, presenters can give a live presentation using the same UI (i.e., presenter view) and show the audience view without visualization to the viewers.

Visualizations

To enable presenters to focus on their narration and the original video contents, DemoWiz augments the screencast recording by automatically overlaying simple glyphs.

Input Event Glyphs

DemoWiz overlays visual annotations of events on the screencast recording in a graphical way where the events happen. For example, in Figure 5.1, the presenter clicks and drags the map view to the right. DemoWiz uses the following simple, distinctive glyphs to differentiate event types as Figure 5.3 shows:

- Mouse click: a red circle with a radius of 20-pixels,
- Double-click: a green circle with a radius of 20-pixels,
- Mouse drag: a thin, orange line with a dot at the start point and an arrowhead at the end point,
- Mouse scroll: a thin, yellow line, 80 pixels long, with an arrowhead, and
- Keystrokes: text in blue.

At any given time during the video playback, DemoWiz shows the current event and the upcoming event on the video. We tried to show more than two events within a fixed time period in our initial prototypes. However, we noticed several issues. First, the view becomes too cluttered to understand at a glance, especially when the original video is visually complex. Second, it is not easy to convey the order of the events. Third, it is difficult to observe when multiple events are spatially close. Therefore, we provide minimum but essential events for recall.



Figure 5.3: DemoWiz visualizes input events in a graphical way. From the left to right we show a mouse click, double-click, a drag, a mouse scroll, and keystroke events. These glyphs are overlaid on the video recordings.

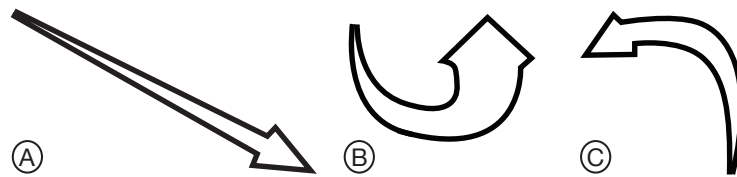


Figure 5.4: Three types of motion arrows in DemoWiz that guide presenters to the next event of different distances at a far (A), nearly the same (B), and near location (C).

Visual Guides to the Next Events

In order to help guide the presenter’s attention, DemoWiz overlays a motion arrow between the current and upcoming events on the demo video (Figure 5.1C). This is inspired by storyboard design used in filming where an arrow effectively shows the movement of a camera or an actor in a single shot [84]. We expand the idea of guiding attention for a specific purpose: the arrow in DemoWiz shows the movement from one action (e.g., click a checkbox) to another action (e.g., click a button). By overlaying this motion arrow, the visualization matches the flow of a presenter’s attention when they observe the video content.

Since the distance between two consecutive event segments vary, we created three visual designs to make sure the arrows are visible to lead a presenter’s attention:

- For two events that are located far away (e.g., clicking an “OK” button after selecting a checkbox on a page), apply a *straight* arrow (see Figure 5.4A).
- For events that are nearly at the same location (e.g., click the “Next” button twice to navigate a list of selections), apply a *round* arrow that points to the current location (see Figure 5.4B).
- Otherwise, apply a *curved* arrow (see Figure 5.4C).

Sense of Timing

DemoWiz provides a sense of timing for an upcoming action so that presenters can adjust their narration. First, DemoWiz embeds a *progress bar* in the motion arrow to show relative time (Figure 5.1C). The green bar shows the proportional time that has been passed before reaching

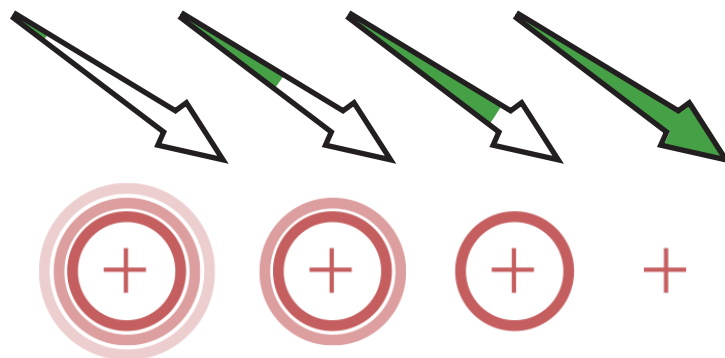


Figure 5.5: A progress in time guides the presenter from the current event (left) gradually to the upcoming action (right) using relative timing with a progress bar (top) and absolute timing (bottom).

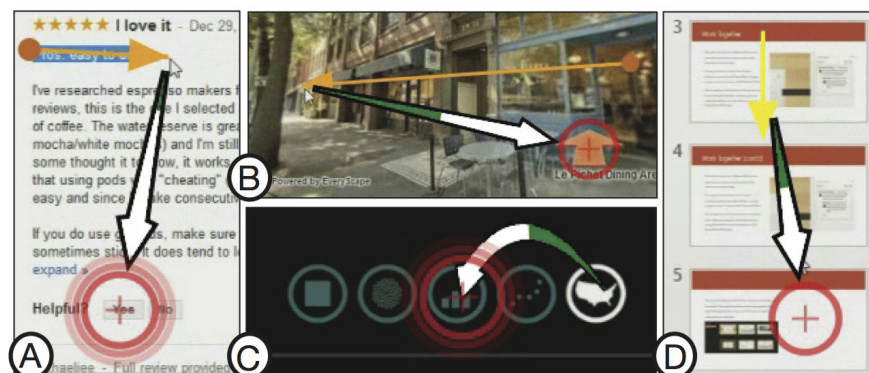


Figure 5.6: Examples of DemoWiz visualizations with four different systems and input event sequences.

the next event (Figure 5.5 top). When a motion arrow is filled up with green, it fades away and guides the presenter to the next action. We were concerned that people may associate the length of an arrow to the length of time. Therefore, we also incorporated a *countdown* visualization where circles will fade out in the last three seconds before the next action starts (Figure 5.5 bottom) to convey absolute timing.

Visualization Examples

Figure 5.6 presents examples of DemoWiz visualizations with four different systems. The glyphs effectively show the start and end points of mouse drags and the locations of mouse clicks. Motion arrows help direct the presenter's attention between events, such as start the end of the drag event to clicking a button (Figure 5.6A and B), clicking between several options (Figure 5.6C), or selecting a specific slide after scrolling down (Figure 5.6D).

Lightweight Editing During Rehearsal

During rehearsal for their demonstration, presenters can modify the video timing and add reminder notes for their narration. DemoWiz shows the type and length of each event in a sequence in a timeline (Figure 5.1A). Each segment is shown as a block whose width indicates its length in time. To simplify the timeline and avoid fine-grained adjustment, lengths of event blocks are rounded to the second. Presenters can modify the playback speed of a segment by dragging the boundaries of a segment on the timeline. For example, presenters can speed up to shorten long text inputs, and slowdown for fast mouse drag inputs that select multiple objects.

Sometimes a change in the playback speed may result in an awkward effect that is noticeable to the audience, especially when showing a UI transition. Therefore, DemoWiz supports two special time control markers to enable breaks in the narration. Presenters can add an adjustable *pause* segment, at which the system will pause at the last frame of the previous segment for the specified length of time. If presenters prefer full control on pause length, a *stop* marker ensures the video stays paused at the last frame of the previous segment and will not proceed until presenters manually resume the playback of the video.

DemoWiz enables presenters to add a short text note (such as the reminder “*Move and zoom...*” in Figure 5.1) so that they could remind themselves of upcoming actions at a higher level. The note can be positioned manually at any location on a video so that it does not block important video content, and will be shown for 3 seconds before the associated event. For every edit that is associated with time changes (including playback speed and pauses), DemoWiz computes and updates the total presentation time as well as updating the progress bar and countdown to provide accurate timing.

Presenter View

During presentation, DemoWiz shows two views in separate windows. Presenters can observe visualizations using the presenter view, while the audience will see the audience view with a full-screen video that has no enhanced information. DemoWiz synchronizes the videos in both views based on presenters’ editing decisions to ensure the same playback speed and time.

As with a conventional video player, presenters can control the video, to pause and play at any time. In addition, when a video is paused (or stopped), presenters can hover the mouse over the demo video in the presenter view to point out an important area, as many presenters currently do in a live demo. DemoWiz then simulates and synchronizes a mouse cursor in the audience view to help the audience follow the demonstration.

Implementation Details

During recording, DemoWiz captures the screen within a specified region and logs low-level system input data with *timestamps* (with an accuracy of 0.1 seconds) from the operating system, including:

- *Mouse events* (mouse downs, mouse ups, and mouse wheel movements) and their *positions* (in x-y coordinates relative to the screen-captured region).
- *Key-press events* (keyboard input).

Once presenters finish their demonstrations, DemoWiz analyzes the low-level event stream and transforms it into high-level event metadata. For mouse events, we pair each mouse down and up into mouse *clicks*, *double-clicks*, or *drags*. We group any consecutive mouse wheel events within a time threshold of 2 seconds to one *scroll* event and any key-press events within the same threshold to one *keystroke* event (e.g., combine keys d-o-w-n-t-o-w-n to “downtown”). For each high-level event, we log the *start* and *end time* (timestamps of the first and the last low-level event).

Based on the start and end times of these high-level events, DemoWiz segments the screencast video recording into *event segments*. Any gap between two consecutive input events is marked as an *inactive segment*, which may include mouse hovering, UI transitions of the demo system, or static frames with no visual changes. DemoWiz adjusts the boundaries of these event segments to avoid any short visual effect that cannot be observed. DemoWiz examines segments in a linear order to ensure each segment lasts at least t_{min} seconds long, which is set as one second based on our early testing. For an event segment S_i of time (t_{start}, t_{end}) that $t_{end} - t_{start} < t_{min}$, DemoWiz expands 0.5 second forward and backward if S_{i-1} and S_{i+1} are inactive. If the adjusted S'_{i-1} and S'_{i+1} are shorter than t_{min} , DemoWiz merges it to the shorter neighbor segment. Currently, DemoWiz does not analyze these inactive segments, but techniques including computer vision and video analysis [14, 46] can be applied for finer segmentation.

The capturing program is implemented in C#. Two APIs were used: 1) the Windows Event Log API for mouse and keyboard hooks and 2) the Expression Encoder 4 API for screen recording running on Microsoft Windows 7. The recorded metadata (stored in a JSON object) and screencast video (in MP4) are read by the Presenter UI, which is implemented using standard Web technologies, including HTML5, CSS3, JavaScript, and jQuery. In particular, the visualization is rendered on the canvas element on top of the video object on the fly based on the video playback time. The audience view is generated by the main browser window of presenter view for video control.

5.5 Evaluation

To evaluate the DemoWiz design, we conducted a controlled experiment in which participants recorded and edited a demo video, and gave a presentation with the edited video. Specifically, we wanted to see if presenters would evaluate their own performances higher with the support of our augmented visualizations and control of timing.

User Study

Baseline Condition: DemoWiz without Visualization

Since DemoWiz allows for rapid editing of the video, it would have been unfair to compare it with a conventional video player without supporting any editing during the rehearsal phase. We therefore modified our system to serve as the baseline condition, providing participants with the same lightweight editing of the video in each condition. However, during presentation, the baseline condition was similar to a conventional video player that shows only the video without event timeline and augmented visualizations. It also did not support the *stop* markers and *text notes*, i.e., participants could only adjust playback speed of each segment and add variable length *pauses*. During presentation, participants only saw the video with a traditional timeline. They could, however, pause (or stop) and resume the video manually at any time during playback.

Study Design

We conducted the study as a within-subjects design in a usability room. After recording and editing a video using the same system, each presenter gave a presentation with both systems to an experimenter. To control the effect of order and learning, we prepared two tasks that included similar interaction flows and counterbalanced the order of the two systems—DemoWiz and Baseline—but we fixed the order of tasks. Even though presenting to a single audience member in a usability room is not the same as using the system with a large conference audience, it is important to control the tasks and presentation as closely as possible to understand the relative benefits of the system in comparison with a baseline condition.

For each condition, we observed and coded the *timing* of narration that matched the video content and noted the time in seconds when an event was described *before*, *at*, or *after* the action happened in the demo video. We also marked obvious *breaks* between narrations, *errors* when the narration was not about the current or following events (e.g., discussing actions in a different order than they actually occurred), and *misses* when an important action was not mentioned. To avoid unconscious bias that might influence the coding of the videos, we neutrally named the recordings and coded them all in a batch. We focused on objective timing measurements as much as possible, measuring deviation from specific video events and their corresponding narrations down to a second. Finally, we gathered qualitative feedback through satisfaction and preference questionnaires.

Participants

We recruited 12 participants (10 males and 2 females) from a software company. However, we excluded the data from two participants (1 male and 1 female); one was due to a software bug during one condition and another was because the participant requested to restart a presentation in one condition. The average age of the effective 10 participants was 37.3 ranging from 24 to 64 years of age. We recruited participants who had experience at showing a software demonstration to an audience such as giving a presentation at a conference. Four participants were native English speakers and the rest were fluent in English. The expertise of participants included audio processing,

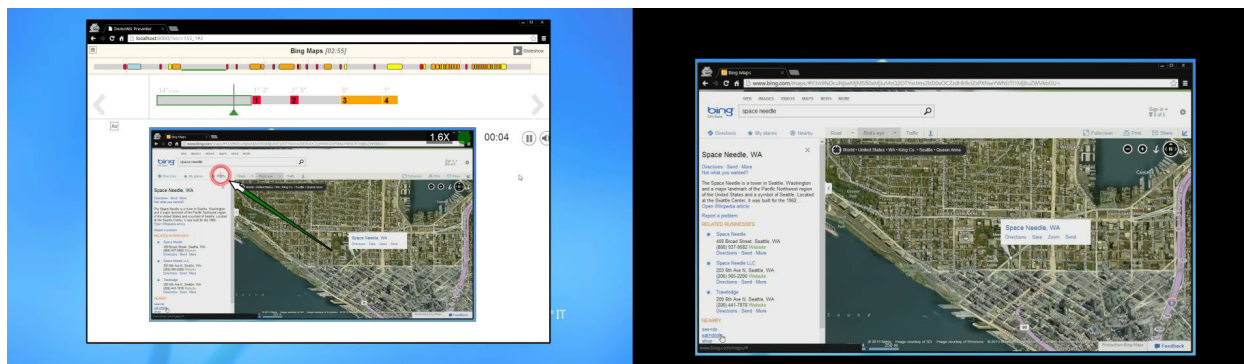


Figure 5.7: Participants saw the presenter view, shown on the left, while giving a presentation in the study. The audience view on the right was shown in the other display with synchronized playback.

computer graphics, human-computer interactions, machine learning, networking, and software engineering. Each participant was compensated with lunch coupons worth \$20.

Procedure and Tasks

Each session consisted of one training task and two experimental tasks. For the training task, to introduce the common features for recording and editing the video, we designed a simple workflow of five steps to demonstrate editing of a slide using PowerPoint. The experimenter briefly demonstrated an example and then introduced the recording program that captured the screen. Participants were then asked to practice and record using the recording program.

The two tasks consisted of a similar sequence and interactions: 1) searching with Bing Maps to show the 2D map view and the Bird's Eye view, looking for a restaurant, and navigating to the interior view of a specific restaurant; and 2) searching with Google Shopping to show the search results with the Grid view, filtering and voting for reviews, and navigating the 3D product view of an espresso machine. For each task, we provided a specific scenario along with a list of subtasks. The experimenter walked through this list with participants to ensure that they could easily find the features that needed to be demonstrated. Participants were then asked to practice (3-5 minutes), record (about 2 minutes), and rehearse and edit (5-10 minutes).

To help simulate a conference setting where participants would not be able to present immediately after having recorded a demonstration, we inserted an intentional 1-minute gap between rehearsal and presentation. During this gap before giving the presentation, we asked participants to watch a conference showcase video. Participants were then asked to stand up and gave a 2-3 minute presentation to the experimenter in a usability room.

After each task, participants filled out a questionnaire of 8-10 questions asking about their experience (8 for the Baseline condition, and 10 for the DemoWiz condition). At the end of the session, an online questionnaire was provided for them to present overall preferences and leave comments. Each session lasted about 1.5 hours.

Experiment Setup

Each participant used a desktop computer running Windows 7, Expression Encoder 4 for screen recording, and a web browser for the DemoWiz user interface. A regular mouse and keyboard were provided, along with two 27-inch displays, one for editing (during rehearsal) and showing the audience view (during presentation, see Figure 5.7 right), and the other for the presenter view (see Figure 5.7 left) on a stand-up table. The resolution of both displays was 1920×1200 pixels. The average captured screen area was 1311×857 pixels. In the presenter view, the video resolution was within 1000×600 pixels; in the audience view, the screencast videos were resized to fill the entire display with wide border in black. During the study, the experimenter stayed in the room, providing instructions and sitting behind the participants during the recording and editing phases.

Results

Ten participants successfully recorded, rehearsed, and gave a demo with both systems.

Subjective Preference

Figure 5.8 shows the average subject responses (on the 7-point Likert scale) from presenters for both systems. We analyzed these subjective responses using a Wilcoxon signed-rank test. We found significant differences in responses for ease of narration (DemoWiz $\mu = 6.2$ over Baseline $\mu = 4.5$, $p = .018$) and ease of presentation (6.4 over 5.2, $p = .048$). We also found marginally significant differences in participants' overall satisfaction with their presentations (5.5 over 4.7, $p = .062$). Participants also tend to agree that DemoWiz helped them interpret timing (6.1 over 4.4, $p = .067$).

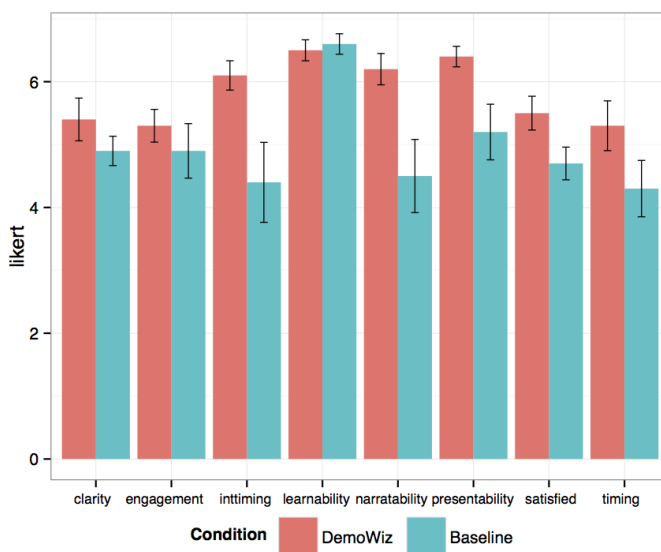


Figure 5.8: User feedback from questionnaire on the 7-point Likert scale.

In addition, 9 out of the 10 participants preferred DemoWiz to the system without visualization and would choose to present with DemoWiz if they were asked to give a public software demo; the remaining participant indicated no preference for both questions. The general feedback was also encouraging. For example, P1 commented “*Awesome system. I’d use it today.*” and P5 “*felt more confident in being able to present what I wanted to.*”

Visualization as a Supportive Cue

Participants answered that they were able to understand DemoWiz visualization of input events ($\mu = 6.0$) and found it supportive for their presentations ($\mu = 6.3$). They also commented that the DemoWiz visualization supported the presentation in various aspects: “*the visualization reminds of the order of the content*” (P1), “*Really liked the ability to know what was coming up*” (P2), “*It provides better insight of the progress of the video*” (P6), and “*viz gave me an idea about timing or something I was going to forget to say*” (P9).

Narration Timing

We coded the 20 recordings of participants’ final presentations to observe the timing of narration of each action in correspondence with the video content (11 key events for both tasks). With DemoWiz, participants tended to *anticipate* the upcoming events rather than talk afterwards, where the average timing was -0.1 seconds with DemoWiz (i.e., narrated the action before it happened) and 0.4 seconds with the Baseline condition (i.e., explained the action after it was shown). We found a significant difference in the number of times that events were anticipated by the narration, co-occurred, or occurred after the fact ($\chi^2(2,220) = 8.6, p = .01$, see Figure 5.9).

In general, this supports our suspicion that DemoWiz would help in anticipating an event as opposed to talking about it after it occurred. More important though, was how often a narrator spoke about an event within several seconds of when the event actually occurred. By defining *better*

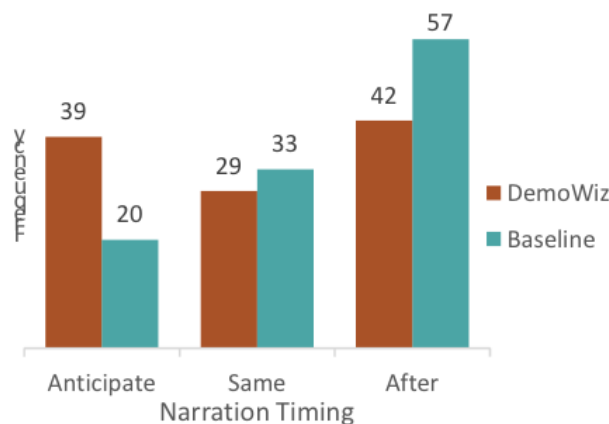


Figure 5.9: The number of times events were anticipated by the narration, co-occurred, or occurred after the fact.

timing as when a presenter's explanation came within 2 seconds of a shown event (either prior, exact, or after), there was marginal significance by condition ($p = .089$ with DemoWiz performing better). In addition, with the Baseline condition, the timing of narration was less consistent and off more, varying from 6 seconds early or 10 seconds late with a variance of 3.9 seconds, in comparison to the DemoWiz condition with at most 3 seconds early to 3 seconds late and a variance of 1.9 seconds.

Five participants had an obvious *error* (forgot the next action or incorrectly narrated another action), had a long *break* (waiting for more than 2 seconds until the action was made), or *missed* an action (did not explain an important feature) when presenting with the Baseline condition. On the other hand, in the DemoWiz condition no errors were made, and there were only one long break and one miss from two different participants, respectively.

Participants' comments also support the fact that DemoWiz helped presenters anticipate the upcoming events. P7 explained, *"(I) felt better able to time my speech to coincide with visual events, rather than trailing after them. Without the event visualizations, I felt like I was talking about what the audience had just seen, rather than having my words and visuals combine to a single message."*

Editing Experience

We collected comments on the workflow. Participants found it easy to record ($\mu = 6.4$) their demonstrations with DemoWiz. For editing features, they found it easy to edit in general (6.6), including controlling the playback speed (6.5) and adding pauses and stops (6.5), but it was less easy to add text notes (4.8); only two participants used this as reminders.

Although using different strategies, all of the participants adjusted the playback speed for matching their narration. Some sped up whenever possible and added stop markers for transitions; some slowed down the repetitive actions (such as drags) to demonstrate effects. P6 said, *"I really liked being able to add 'stop' events so I could 'fake' my demo better."* DemoWiz made it easy for participants to separate the capturing and presentation preparation as P5 explained, *"Overall, recording was very easy. In fact, as I got to the second task, I realized that I really don't need to think about the words as I record because later on I will be able to slow down and speed up time ..."*

On average, the length of demo videos was 2'09" before editing and 2'05" after editing, and the presentation was 2'38" long. Each participant spent 7.5 minutes on average to edit. For each demo of 44 segments on average, participants adjusted 3.15 segments for speedup and 4.25 segments for slowdown, and added 0.55 pause markers. In the DemoWiz condition, 1.2 stop markers and 0.2 text notes were added.

Discussion

New Tool to Present Video in Real-Time

DemoWiz is an attempt to make demo videos more engaging by helping presenters anticipate the upcoming events rather than reacting to them, leveraging a refined workflow with augmented visualizations. Overall, participants liked the DemoWiz visualization, finding it supportive rather

than distracting. For examples, P4 said, “*Event visualization was very powerful – definitely the way to go*”. and P2 (who first experienced Baseline) was originally skeptical when he first saw the visualization but immediately found it helpful and not distracting. This corresponds with our goal of designing the visualization with a minimal cognitive load.

Editing Capabilities

Lightweight editing during rehearsal not only makes it easy to edit the recorded video but also lowers the burden of the initial recording. Presenters do not have to prepare a complete script for exact timing. They also do not have to repeat recording many times to grab the best recording.

Some participants appreciated our design choice of providing only minimum but essential editing capabilities to make the process as light as possible. P2 mentioned that “*Ironically, I think it’s better to have limited editing feature set -- this system was very easy to learn/use*”. A few participants expressed the need for more editing features: P1 explained, “*(I wish the system could be) cutting events in parts so that I can slow down/speed up/remove portions of, e.g., a mouse trajectory*”; P3 wanted to “*flip segments around*” and P8 thought “*break up or merge blocks*” would be helpful. We found these interesting as the system enabled more possibilities, but there is a tradeoff between providing a powerful tool and lowering the burden in editing. We believe that this is a design choice that needs to be balanced.

Our system does not support combining two or more video clips for a presentation. Sometimes, presenters may also want to update part of the existing material to show new features of their developing systems. For example, P4 explained that he would like to see “*the ability to record multiple clips and insert them in a timeline.*” This would be straightforward future work because the current DemoWiz framework is designed to be able to implement this.

Editing can still be limited to support fine timing control of narration. P10 explained, “*The length of narration changes each time I present, and it is difficult to perfectly align the*” timing. Automatically navigating a video based on presenters’ performances could be an interesting avenue of exploration, similar to scenarios of following a tutorial [174] or performing music [134]. However, we decided not to pursue this approach because it would present its own form of risk relying on unreliable speech recognition during a live presentation. Also, considering the time constraints presenters usually have, we chose to provide full control for presenters rather than trying to intelligently update a video.

Study Audience Engagement

In our user study, we gathered presenters’ opinions as to how engaging their presentation was, and we explored the relative timing of the narration to events in the video. Ultimately, however, our goal is to help increase audience engagement. Measuring audience engagement is an ongoing topic of research, and we would like to explore ways of quantifying the relative impact of the DemoWiz system, but that work was out of scope for this project.

Enhance the Audience View

Some participants commented that it would be helpful to highlight certain input events for the viewers to observe subtle changes. For example, P10 wanted to enable, “*visualize mouse events such as clicks and scrolls for the audience so they know what is going on.*” The current DemoWiz framework makes it easy to achieve this goal by highlighting the audience view *only* when the event happens. In other words, presenters and audience will see different visual effects, where the former observe events in advance and the latter see a visualization synchronized with the demo video content.

Beyond Software Demonstrations

Although our current implementation is focused on software demonstrations, we argue that it is possible to expand our system design to more advanced inputs. By defining event types that a system recognizes (e.g., a pinch gesture on a multitouch device or a specific pose detected by a 3D sensor), it is possible to log the events and align them with the captured video for later use. In addition, the enhanced presentation mode can be potentially applied to other domains where knowing the timing and the sequence of events is crucial, such as narrating over animated presentation slides with dynamic graphical objects. DemoWiz is an important first step towards validating this general approach and we believe our work could inspire future research in these directions.

5.6 Conclusion

This chapter introduced DemoWiz, a system with a refined workflow that helps presenters capture software demonstrations, edit and rehearse them, and re-perform them for an engaging live presentation. DemoWiz visualizes input events and guides presenters to see what’s coming up by overlaying visual annotations of events on the screencast recording where the events happen in a screencast video. It also provides lightweight editing for presenters to adjust video playback speed, pause frames, and add text notes. A user study showed that DemoWiz was effective in helping presenters capture timing and narrate over a demo video.

Chapter 6

DemoCut: Instructional Videos from Demonstration

Amateur instructional videos often show a single uninterrupted take of a recorded demonstration without any edits. While easy to produce, such videos are often too long as they include unnecessary or repetitive actions as well as mistakes.

In this chapter, we introduce DemoCut¹, a semi-automatic video editing system that improves the quality of amateur instructional videos for physical tasks. DemoCut asks users to mark key moments in a recorded demonstration using a set of marker types derived from our formative study. Based on these markers, the system uses audio and video analysis to automatically organize the video into meaningful segments and apply appropriate video editing effects. To understand the effectiveness of DemoCut, we report a technical evaluation of seven video tutorials created with DemoCut. In a separate user evaluation, all eight participants successfully created a complete tutorial with a variety of video editing effects using our system.

6.1 Introduction

Do it yourself (DIY) instructional videos show viewers how to carry out physical tasks, such as craft projects, home improvement, repair, or cooking [206]. The availability of free video-sharing sites like YouTube and Vimeo has led to an explosion in user-generated video tutorials online [130]. Effective instructional videos use a range of video editing techniques, including subtitles, annotations, and temporal speed up effects, to concisely communicate physical procedures. However, producing high-quality videos requires significant time investment and expertise. In addition to recording possibly many takes, authors must review and cut the footage and then apply the appropriate editing effects [157]. Instead of investing this effort, many amateurs instead create videos that simply show a long uninterrupted recording of a demonstration. While such videos are easy to produce, they often include a lot of unnecessary footage (e.g., pauses, mistakes, long repetitive actions) that makes it difficult for viewers to focus on the most important steps and actions.

¹ This work was published at UIST 2013 [50].

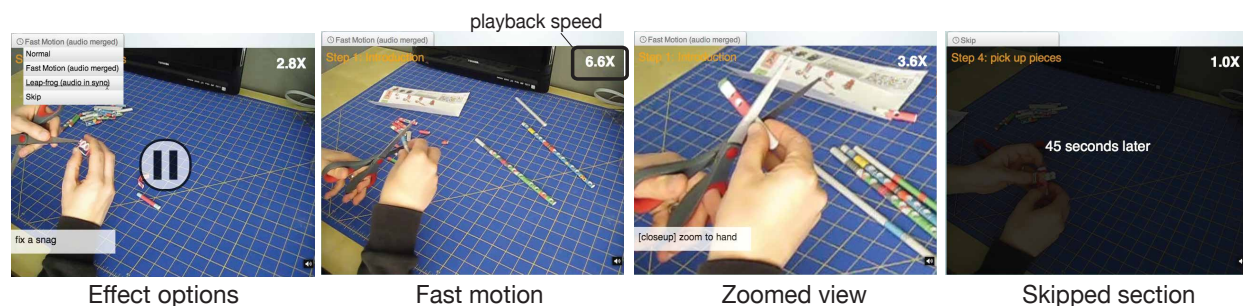


Figure 6.1: DemoCut automatically segments a single-shot demonstration recording and applies video editing effects based on user markers (A), including subtitles, fast motion (B), leap frog, zoom (C), and skip (D).

The goal of our work is to help amateur users produce effective instructional videos. We analyzed existing DIY videos and interviewed video authors to uncover key challenges in creating high-quality video tutorials: organizing long, single take recordings into meaningful steps; removing/condensing unnecessary or repetitive actions; and adding effects that emphasize important details in the demonstration. To address these challenges, we introduce DemoCut, a semi-automatic video editing system that generates concise instructional videos from recorded demonstrations (Figure 6.1).

With DemoCut, users record a single take of a narrated physical task demonstration and then roughly annotate the recording with markers that indicate high-level steps, important actions, supplies and mistakes. Based on these annotations, the system uses a combination of video and audio analysis to automatically organize the recording into meaningful segments and apply editing effects that make the tutorial more clear and concise. DemoCut supports both temporal effects that increase playback speed or skip segments, as well as visual effects, such as zooming, subtitles, and visual highlights. DemoCut also provides an interface that allows users to quickly review and edit the automatically generated effects.

We used DemoCut to create seven video tutorials in five different DIY domains: electronics, crafts, art, repair and food. The generated videos were concise in terms of video length and descriptive instructions with low effect error rates. We also conducted a small user study where participants used our system to record and edit their own video tutorials. All participants successfully created a complete tutorial that included a variety of video editing effects, and the qualitative feedback on DemoCut was very positive. The participants felt that DemoCut enables a convenient workflow for creating concise video tutorials and that the automatic editing effects are particularly useful for speeding up repetitive actions.

In summary, the main contributions of this work include:

- A light-weight annotation-based interface for editing instructional videos.
- A set of marker types for annotation derived from our formative work. Markers represent different types of moments that lead to different editing effects.

- A semi-automatic approach for editing DIY video that combines user annotation with audio and video analysis.
- A working implementation of this approach and a preliminary evaluation with both novice and expert video editors.

6.2 Design Guidelines

Understanding Current Practice

To gain insight into the editing decisions that go into effective demonstration videos, we analyzed a set of 20 highly-rated videos on YouTube and interviewed six of the authors of these videos about their recording and editing processes.

Video Analysis

To cover a range of topics, we chose five different DIY domains (electronics/science, craft, home/repair, art, and food) from a popular DIY website². We selected the first four videos on YouTube for each domain that satisfied a set of criteria chosen to ensure they were effective, including:

- Produced video: evidence of editing through cuts
- Camera angle: 1-2 static camera viewpoints
- Content: 1-2 instructors with audio narration
- Popularity: a minimum of 1000 views, with less than 10% dislikes from the total like-or-dislike ratings.
- Experience: authors with >5 published how-to videos.

20 videos from 20 distinct authors were coded in a week. Figure 6.2 presents selected examples; Appendix B preserves detailed information. The average length of these videos is 5 minutes and 5 seconds ($max=9'08''$, $min=1'54''$), and the average view count is 269,426 ($max=4,004,613$, $min=1,156$). Although these tutorials cover various topics and tasks, we observed several common characteristics of the videos:

- **Narration.** All of the videos include narration that explains what is happening in the tutorial. Most authors seem to narrate during the demonstration (70%), while fewer authors record a separate voice-over track.

² <http://makezine.com/>



Figure 6.2: We analyzed 20 DIY instructional videos. Examples included (clockwise from top left): Microcontroller circuit design, tablet screen replacement, custom shoe painting, and creating latte art.

- **Speed-up effects.** Most videos (60%) include editing effects that speed up repetitive actions, such as screwing in fasteners or chopping vegetables. In many cases, authors break the sync between the audio and video tracks in these sped-up sections so that the narration plays continuously at normal speed with no long silences while the video plays at a faster speed.
- **Annotations.** Many videos (65%) include titles that add relevant information about the task, including descriptions of depicted objects or actions, measurements, elapsed time, and details that are not shown in the demonstration. Some videos also include other annotations (e.g., arrows, rectangular highlights) that emphasize important details.

This analysis suggests that authors apply a common set of editing techniques. Since the final videos convey limited information about the recording and editing processes, we interviewed several authors of the selected videos. We hope to learn what kind of footage they omitted and how much time they spent on editing. We also wanted to understand the rationale behind the edits.

Interviews with Tutorial Authors

We contacted the 20 YouTube account holders of the analyzed videos, and interviewed the first six who responded (all males, ages 17 to 48). One of the YouTube user accounts corresponded to a 3-person team, which we counted as a single participant (P6). Among the participants, two were professional tutorial makers, while the rest were amateurs. For editing, three used Apple Final Cut Pro, two Corel VideoStudio, and one Adobe Premiere. Table 6.1 summarizes other information about the experience of the authors.

ID	Category	Experience	Videos	Sample Project
P1	electronics	professional	48*	Body-mounted camera
P2	home/repair	amateur	162	Powder coating aluminum
P3	science	professional	45*	Develop caffenol film
P4	home/repair	amateur	717	Snowblower repair
P5	home/repair	amateur	33	Paintcan setup
P6	electronics	amateur	5	On-beat disco light

Table 6.1: Background information about interview participants. * *Numbers of videos published on personal YouTube channels, excluding those on the professional channels.*

Capture

Except for the team (P6), all of the participants record demonstrations individually without any assistants. P2–P6 use a single video camera, while P1 uses an extra camera to capture closeup shots. To keep the recording process simple, the amateur authors tend to capture demonstrations in one uninterrupted take, narrating the action as they go. Naturally, such recordings often include mistakes (e.g., walking out of frame to retrieve a forgotten tool) and long, repetitive actions. In contrast, the professional authors create a script beforehand and record the narration separately.

Editing

All of the participants mentioned the importance of editing the final video tutorial down to a reasonable length (5–10 minutes). The goal is to provide enough information to understand the demonstration, but at the same time keep the video lively and interesting. P2 described his strategy as follows: “*If you can get rid of it and the video content still gets through, get rid of it*”; “*The way to make your video sizzle is to have good cuts, good points.*” As a result, authors spend much of their editing time deciding on cuts, segmenting the video, removing and merging shots, and adding visual effects to speed up repetitive actions. They also take time to add subtitles and annotations. As P4 explained, “*the filming is the easiest part; it is the editing that’s the challenge.*” Overall, participants reported that filming time takes from one hour up to one day, and editing time typically takes 6–12 hours, depending on the the complexity of the project.

Design Implications

Based on our analysis of existing video tutorials and interviews with tutorial authors, we identified a few key aspects of the tutorial creation process that have important design implications for DIY video editing systems.

Working with single take, single camera footage.

Most amateur authors record demonstrations in a single take with a a single camera. As a result, the captured footage often includes mistakes and long, repetitive actions.

Making concise videos.

The most important design principle for creating effective DIY videos is to make them concise without sacrificing clarity. To this end, authors remove/condense unnecessary or repetitive actions so that the resulting video only contains salient footage.

Retiming audio and video tracks separately.

One common technique for speeding up a video involves breaking the synchronization between the audio and video tracks so that they can be retimed separately. In cases where the narration refers to specific visual events, the tracks should remain aligned.

Emphasizing important information.

Most effective DIY videos include titles, annotations and/or closeup views to emphasize relevant information and highlight key details.

Focusing on high-level editing decisions.

Amateur users often struggle with low-level manipulation of cut points and timing in general-purpose video editors: A system should reduce the editing efforts and enable authors to focus on making simple choices for the final production.

We next describe how these considerations informed the design of DemoCut.

6.3 Creating Video Tutorials with DemoCut

To enable amateur users to produce effective video tutorials, the DemoCut video authoring system semi-automatically edits a long, single take recording into meaningful steps. Early testing revealed that users find it easier to locate specific *moments* in the video than to mark or edit *segments*. Therefore, our Annotation Interface asks users to mark important moments. DemoCut combines the user annotations with audio and video analysis to automatically generate a segmented video with editing suggestions: It removes or condenses unnecessary/repetitive actions and enables flexible

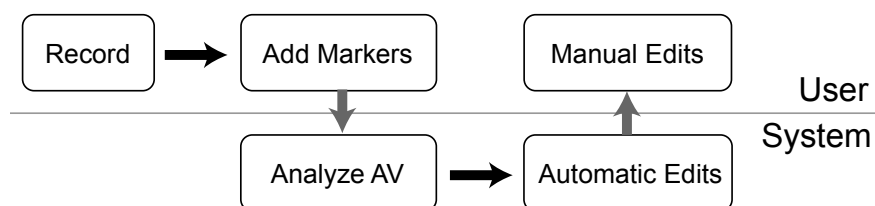


Figure 6.3: DemoCut users first mark their recorded video in the Annotation Interface. DemoCut then segments their recording and suggests video edits, which users can review and change in the Editing Interface.

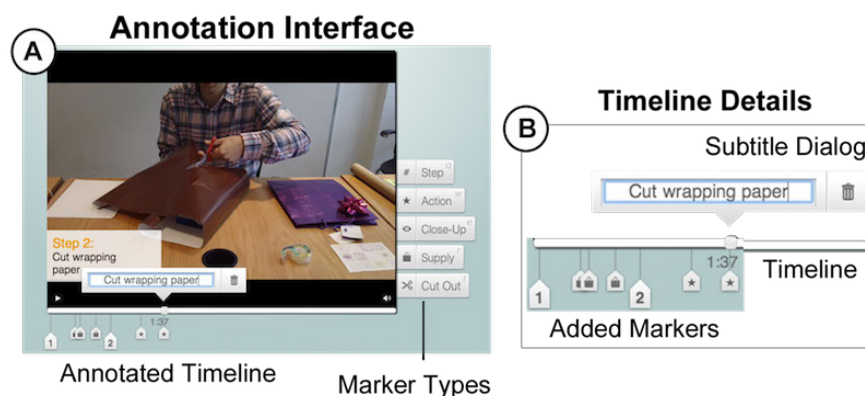


Figure 6.4: With DemoCut’s Annotation UI, users add markers to their recorded video (A). Each marker can be labeled with a descriptive string (B).

synchronization between audio and video tracks. Titles, visual annotations and closeup views are applied to enhance the content. Users can review and revise these decisions in the DemoCut Editing Interface. This section reviews DemoCut from the user’s perspective (Figure 6.3). The following section will describe our video analysis pipeline.

Annotating the Video

The purpose of the DemoCut Annotation UI is to collect high-level information that is difficult to extract automatically but useful in determining how to edit the video. We rely on users to distinguish important from unimportant actions and successful steps from mistakes. The user scrubs through the captured footage and adds markers for distinct moments, such as the instant when he cuts a sheet of paper (Figure 6.4A). DemoCut offers five types of markers for annotating a video:

- *Step*: indicates the start of a major part of the task
- *Action*: marks important moments
- *Closeup*: indicates moments where the action is happening in a small region of the video frame, e.g., for a detailed action such as fastening a small screw.
- *Supply*: indicates a tool or material used in the task
- *Cut-out*: indicates moments of the video that should be removed due to occlusion or a mistake in the performance.

This set of markers was derived from our observations of the structure of effective tutorial videos: actions are treated separately from supplies; zooming can direct the viewer’s attention to a small area of the frame; and step divisions are used to divide actions into meaningful groups. Rather than specify start and end frames, users can place a marker on any frame of an important moment.

Users can add descriptions to markers (Figure 6.4B). These descriptions serve a dual purpose: they are used to generate automatic subtitles, and they are also shown as segment names in the Editing Interface to facilitate navigation. Users can also add visual highlights such as boxes and arrows to any marker.

Automatic Video Editing

Based on the user’s markers, DemoCut automatically segments the raw footage and applies editing effects using the following techniques.

Temporal Effects

We designed four temporal effects to shorten a video. In addition to skipping a segment or leaving it unchanged, we consider the synchronization between the audio and video tracks: People are sensitive to changes in speech playback speed, but video can often be accelerated without loss of clarity. Therefore, our temporal effects accelerate or contract video but keep audio at normal speed.

Fast motion (with merged audio): When a segment includes several sections of narration with intermediate pauses, DemoCut removes the pauses and concatenates the audio segments. Then it speeds up the video so the total video length corresponds to the length of the concatenated audio (Figure 6.5A). This effect is appropriate if tight synchronization between audio and video is not required. For example, an author may describe general strategies for choosing supplies while measuring paper – here audio and video are independent of each other. In this case, DemoCut will accelerate the video to fit the length of the author’s remarks.

Leap frog (with synchronized audio): If synchronization between audio and video is necessary, this effect plays video and audio at normal speed during active audio segments, and skips video in the interstitial segments (Figure 6.5B). Synchronization is important if the author’s face is in the shot (so lip movement and audio match), if actions produce distinct sounds (like cutting paper), or if the narration refers specifically to actions, e.g., when pointing at an object and describing its

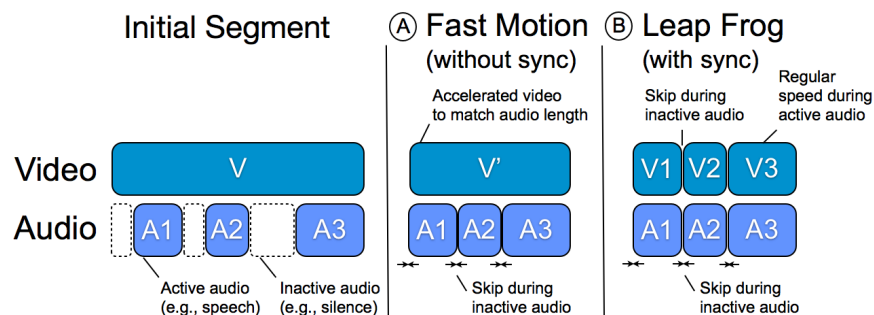


Figure 6.5: DemoCut accelerates playback of video with intermittent audio narration through Fast Motion (A) and Leap Frogging (B).

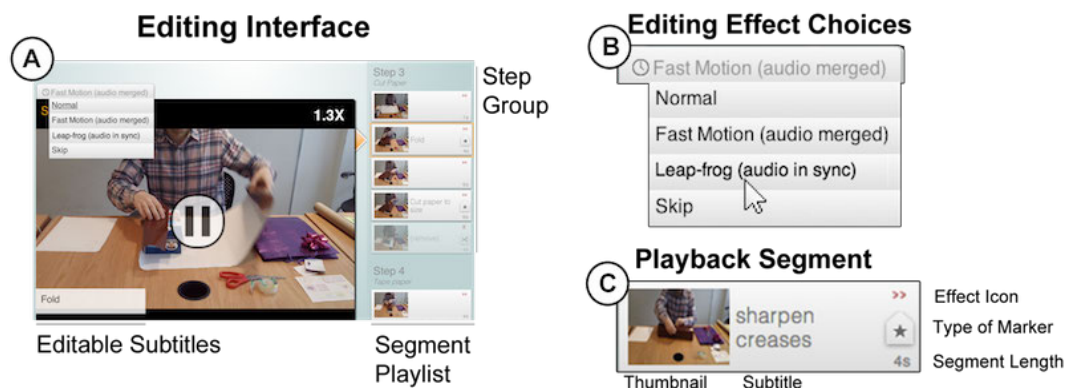


Figure 6.6: DemoCut’s Editing Interface shows automatically generated segments with effect suggestions (A). Users can change the effect (B) applied to each segment (C).

properties. Since DemoCut cannot automatically decide whether synchronization is necessary, it applies the Fast Motion effect by default but offers users control to change that effect.

Skip: Depending on the length of the removed segment, DemoCut either applies a fade through black (for segments up to 15 seconds); or a fade to a title that indicates how much time has passed (e.g., “2 minutes later”).

If these temporal effects are not appropriate, DemoCut plays the audio and video at the captured rate. We call this the *Normal* effect.

Visual Effects

In addition to manipulating time, DemoCut offers three visual effects to structure the video and to provide emphasis. These visuals appear for the duration of the segment DemoCut derived from the user’s marker:

Subtitles: Text entered by the user in the marking phase is converted into automatic subtitles with two levels – a step heading that remains on screen for all segments within a step (e.g., “Wrapping the present”); and a subheading from individual event markers (e.g., “Sharpen creases”).

Automatic zoom: When users create closeup markers, they also specify a rectangular region of interest. DemoCut automatically crops and enlarges this region of the segment.

Visual annotation: DemoCut overlays visual box or arrow annotation specified by the user in the marking stage.

Reviewing and Editing

Since our automatic video and audio segmentation has a limited understanding of the video, it is likely that some editing decisions will be incorrect. For example, DemoCut’s algorithms have no way of inferring whether audio-video synchronization will or will not be required in a given segment. In addition, automatic analysis may also lead to errors: if the narration is not correctly

segmented, speech can be cut off mid-sentence. DemoCut’s editor gives authors the opportunity to review and revise all editing decisions.

In the Editing Interface, the video is visualized as a set of segments (see Figure 6.6C) flowing from top to bottom on the right side of the main video view (see Figure 6.6A). There is no traditional timeline for two reasons: first, editing operations only apply to detected segments (we consciously prevent users from applying frame-level edits to keep with the goal of a semantic editor); second, because segments may come with labels entered by the user, a vertical layout makes it easier to read labels. Users can navigate to any segment by clicking on its thumbnail. Once selected, they can change which effect should be applied to a given segment (see Figure 6.6B). Users can also modify any visual effects, to edit subtitles, resize the cropped region, or add/delete highlights (see Figure 6.7). When satisfied with their choices, users can export a continuous video suitable for online video sharing platforms.

6.4 Automatic Effect Decision Pipeline

DemoCut performs several automated steps to convert the user-annotated input recording into an edited video tutorial. First, the system segments the recording into regions around user-specified markers. This segmentation considers both the similarity of video frames around each marker and the presence of narration in the audio track in order to determine the appropriate segment boundaries (Figure 6.8). DemoCut then automatically applies an temporal and a visual effect to each segment based on the type of the corresponding user marker and the properties of the audio/video content in the segment. The rest of this section describes these steps in detail.

Video Segmentation

Except for the step marker, all of the user-specified markers indicate important moments in the demonstration that correspond to some segment of the recording. In many cases, we can infer the duration of these segments by searching for video frames that look similar to the marked frame.

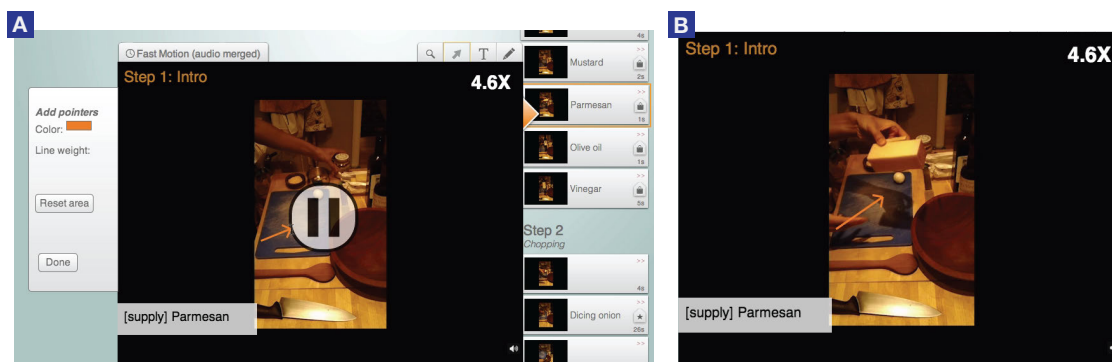


Figure 6.7: Users can annotate a video with visual highlights using the Editing Interface, such as adding an arrow to point out an important area (A). Annotations will be rendered on the fly (B).

For example, in Figure 6.9A, the similar frames before and after a supply marker show the author holding up a bottle of vinegar, and in Figure 6.9B, the similar frames around an action marker show the author grating cheese. For every marked frame T^m , DemoCut uses the following method to compute candidate start and end frames T^s and T^e for the corresponding segment. For the i -th marked frame T_i^m , our algorithm finds T_i^s by comparing T_i^m to earlier frames in the video until it reaches a previous marker at T_{i-1}^m , or until 5% of pixels (in grayscale) have changed by 20%. Similarly, the system finds T_i^e by comparing T_i^m to subsequent frames in the video. To optimize performance, DemoCut compares to frames sampled at 0.5 seconds and ignores overlaps between segments. Segment overlaps are resolved during boundary adjustment after incorporating the audio analysis.

Adjusting Segments with Audio Analysis

Adjacent segments can have different effects that change how video and audio are processed. To prevent such changes from interfering with a video’s narration, DemoCut adjusts segment boundaries to align with audio activity boundaries.

Detecting non-silent sections

Since many DIY videos include prominent non-speech sounds such as chopping noises, power tools, etc., detecting speech automatically is a challenging task. We found that even state-of-the-art speech detection algorithms produce poor results in many cases. As a result, we take a more conservative approach; DemoCut automatically detects non-silent sections in the recorded audio and treats the background sound as part of the narration.

At a high level, our algorithm for detecting non-silent sections works as follows. We compute the “loudness” of each audio window, organize the windows into a histogram based on loudness, and then analyze the histogram to determine a minimum loudness threshold for non-silent windows. We then apply this threshold to categorize all audio windows as silent or non-silent. Finally, we

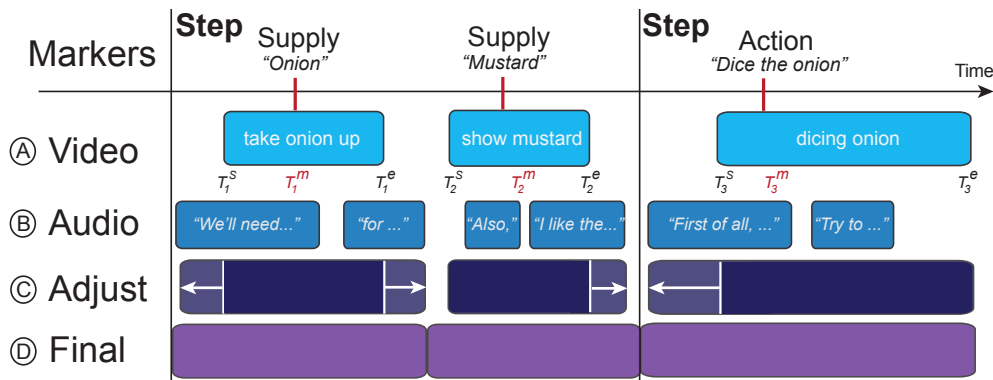


Figure 6.8: Given user markers, DemoCut analyzes both video and audio to segment the demonstration video and apply editing effects.

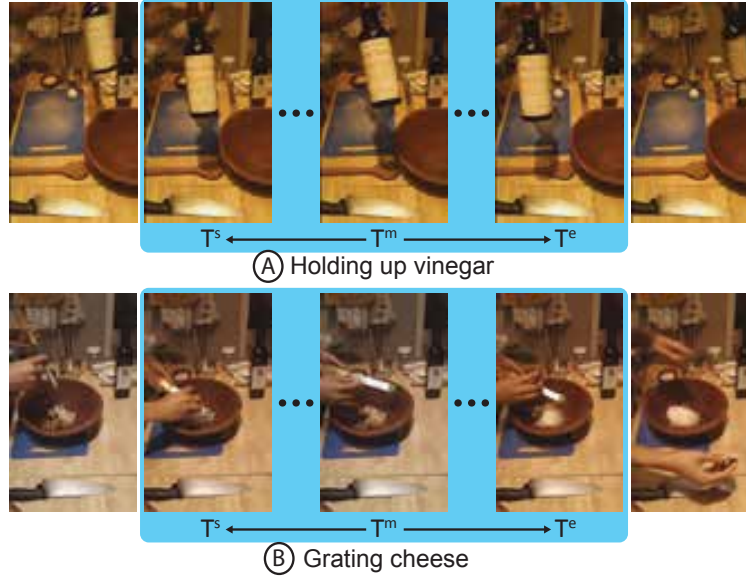


Figure 6.9: DemoCut looks for similar video frames before and after a marked frame T^m to find candidate start (T^s) and end (T^e) frames for the corresponding segment.

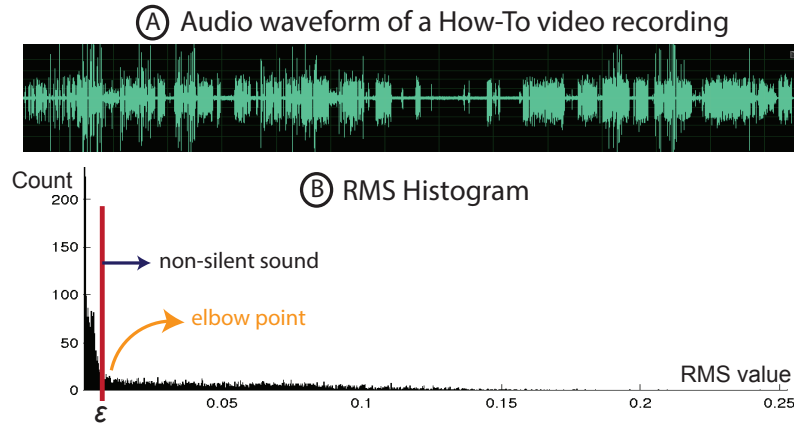


Figure 6.10: We use RMS energy of the audio to find silent and non-silent regions. We determine the threshold for silence by analyzing the histogram of the RMS energy.

filter this categorization to eliminate very short sequences of silent or non-silent samples. Here, we describe these steps in more detail:

Computing loudness. Given an input audio waveform sampled at 44.1 kHz (Figure 6.10A), we estimate loudness by computing the root mean square (RMS) energy [167] across the entire waveform. The RMS energy for a window of size n is $\sqrt{(\sum_n x_i^2)/n}$ where x_i is the value of the i th audio sample in the window. We set window sizes as 0.1 second with $n = 4410$. Prior to computing RMS energy, the audio is normalized and noise-reduced with Adobe Audition.

Computing loudness threshold. After analyzing the RMS energy profiles of several different types of DIY videos, we found that the vast majority of recorded audio represents background sound, which tends to have similar and fairly low RMS energy values. In contrast, user narration varies from medium to high RMS values based on the speaker’s distance to the microphone and the sensitivity of the recording device. Based on this observation, we first compute a histogram of RMS energy for all windows in the audio track; the windows that correspond to background sound form a large mass at the low-RMS end of the histogram (Figure 6.10B). To distinguish these “silent” parts of the recording from the narration, we smooth the histogram with a Gaussian kernel, find the minimum derivative point in the smoothed histogram, and set the loudness threshold ε to be the RMS energy value at this elbow point. Figure 6.10B shows the RMS histogram and loudness threshold for one of our example videos, “How to make salad dressing.”

Categorizing silent/non-silent sections. To partition the audio track into silent and non-silent sections, we first label each window as silent or non-silent based on ε . This initial labeling often includes some very short silent and non-silent sections. Since many short silent sections correspond to short pauses between spoken words, we turn any silent sections that are shorter than 0.4 seconds into non-silent sections. Then, we discard any non-silent sections that are shorter than 0.8 seconds to account for any clicks and pops in the recorded audio. The 0.4 and 0.8 second thresholds for silent and non-silent sections were tuned experimentally, and we used these parameter values for all of our results.

Adjusting segment boundaries

In order to avoid cutting off an author’s narration, DemoCut adjusts the video segment boundaries using the non-silent sections of the audio track (Figure 6.8). First, for any segment we find all of the overlapping non-silent audio sections and then grow the segment so that it completely contains all of these non-silent sections. Next, DemoCut resolves overlapping segments: If any two segments overlap, the boundaries must be readjusted. If the overlap region is silent, the region is split into two equal parts and each is assigned to the corresponding segment. If the overlap region includes a non-silent audio section, DemoCut assigns this non-silent section to the segment that has more overlap with the section. If the overlap for both video segments is the same, DemoCut assigns the section to the smaller video segment. Finally, DemoCut addresses any gaps between segments. If a gap is less than 2 seconds, it is merged to the shorter adjacent segment. Otherwise, DemoCut creates a new segment for the gap. Note that such *unmarked segments* do not have a corresponding marker, but they may still show useful details of the demonstration.

Applying Effects

To automatically apply an effect to each computed segment, DemoCut first detects whether there is motion in the video. A segment is considered to be *static* (i.e., no motion) if less than 1% of pixels in the grayscale versions of consecutive frames have changed by more than 20%. To optimize for performance, the segment is sampled at 0.5 seconds for this comparison. DemoCut chooses effects as follows:

Task	Category	Raw footage length	DemoCut video length	# of markers	# of segments	Incorrect Effects	# of non-silent sections	Audio misses	Audio cut-off	Audio false-positives
A: Xbee tutorial	electronics	7'01"	3'27"	16	30	0%	79	5%	0%	0%
B: Paper pipe robot	craft	10'55"	4'40"	18	30	20%	77	21%	12%	0%
C: Ribbons for straps	craft	10'03"	4'23"	39	46	7%	72	15%	7%	0%
D: Fixing front light	repair	6'32"	2'12"	21	33	9%	40	10%	3%	0%
E: How to make grassy head	art	9'28"	5'29"	29	44	5%	86	8%	2%	0%
F: How to make potato stamps	art	16'38"	4'05"	30	45	7%	119	7%	3%	0%
G: How to make salad dressing	food	14'46"	5'38"	33	39	13%	121	6%	2%	0%
AVERAGE	-	10'46"	4'10"	26.4	38.1	9%	83.5	10.3%	4.1%	0%

Table 6.2: A list of how-to videos we recorded to assess the robustness of the DemoCut system.

1. If the segment includes a *cutout* marker, apply “Skip”.
2. If the segment includes a *closeup* marker, apply “Zoom” to the entire segment.
3. If the segment includes any non-silent audio sections, apply “Fast Motion”.
4. If the segment is silent, static, and unmarked, apply “Skip”.
5. If the segment is silent but not static (either marked or unmarked), apply “Normal”.
6. For any marker with a text annotation, apply “Subtitles”.

Implementation

The video and audio analysis is implemented in Matlab. The Annotation and Editing Interfaces are implemented with standard Web technologies (HTML5, CSS3, and JavaScript). An Apache web server hosts these web pages and sends the user annotations to the back-end Matlab system.

6.5 Evaluation

Evaluating Automatic Effect Decision

To evaluate DemoCut’s analysis engine, we recorded seven how-to tasks from the five categories we selected in the formative user study (see Table 6.2 for detailed information and Figure 6.11 for illustrative frames of these videos). The tasks were recorded by 4 people (all authors of this work) in 7 different locations using a Sony camcorder or an iPad with a video resolution of at least 640x480 pixels. We used DemoCut to annotate the recordings and then examined the automatically generated video tutorials.

Overall, the resulting tutorials³ exhibit many of the desired characteristics outlined earlier in the chapter. The automatically edited videos are concise: 2-5 minutes long and 2.5 times shorter

³ The seven videos used to assess DemoCut are listed in this YouTube playlist:
https://www.youtube.com/playlist?list=PLAq2QZEilgn_zyMFFdw88yKjQLhZvyIDi

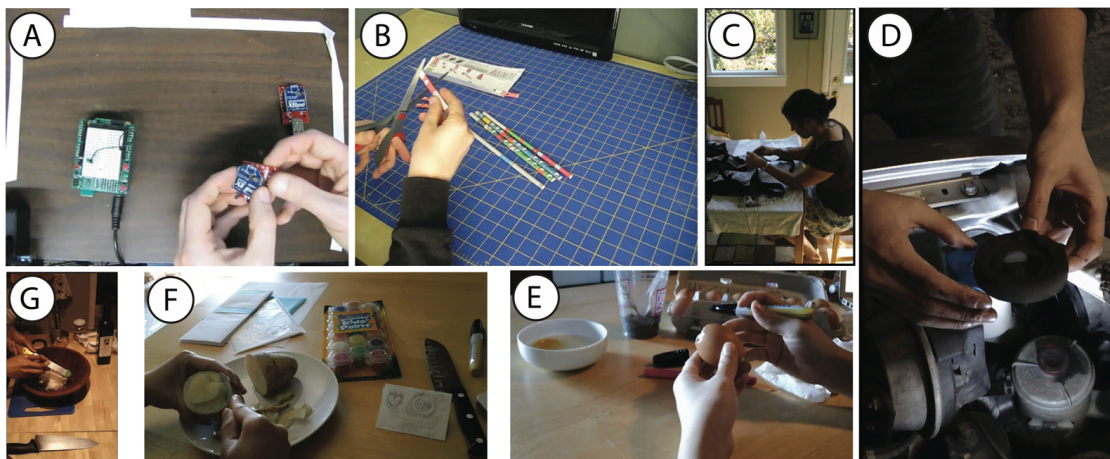


Figure 6.11: Illustrative frames from the seven videos used to assess DemoCut. Labels correspond to task labels in Table 6.2.

than the original footage. In most cases, DemoCut successfully identified segments where the “Fast Motion” or “Skip” effects could be applied to condense the tutorial. For example, the edited salad dressing video uses “Fast Motion” to speed up repetitive actions like chopping an onion and grating cheese, and then skips the segment where the author leaves the frame to toast pine nuts. In addition, the automatically generated titles improve the clarity of the tutorials by adding valuable descriptions of steps, actions, supplies and indicating the elapsed time for skipped segments. In an electronics tutorial, titles like “sending data toggles LED” add important details that are not visible in the video.

There were some situations where the effects were not as successful. To get a more quantitative measure of DemoCut’s performance, we counted several types of errors in the automatically generated videos:

- *Incorrect editing effects.* In a few cases, the “Fast Motion” effect is applied to segments where the audio track should actually be in sync with the visuals. Also, when markers are very close to one another in time, DemoCut sometimes generates very short segments where the editing effects are hard to see. We identify these cases as incorrect editing effects.
- *Audio miss.* We refer to any piece of narration that is not detected as a non-silent section as a miss.
- *Audio cut-off.* We refer to any detected non-silent section that cuts off narration by ending too early or starting too late as a cut-off error.
- *Audio false-positive.* We refer to any non-silent section that is neither narration nor significant activity or background sound as a false-positive.

We report the incorrect edits as a percentage of the total number of segments and the three audio errors as a percentage of the total number of ground-truth narration sections. Table 6.2 shows all

of the results from our analysis. Overall, we found low average error rates (less than 11%) for all of these problems. Also, note that most of these errors can be fixed by changing the automatically applied editing effects in DemoCut's reviewing and editing interface.

User Evaluation

To evaluate the usability and utility of DemoCut, we recruited 8 participants (4 males, ages 20-41) to create how-to video tutorials. We were especially interested in two questions: First, how much *effort* would participants have to invest to mark and edit their own tutorial videos with DemoCut? And second, what are the *qualities of the resulting videos* – both in terms of strengths and shortcomings?

Task and Materials

The participants were asked to create a tutorial for wrapping and decorating a present. We chose this task because it is relatively simple but still involves multiple distinct activities and steps that can be accomplished in 5-15 minutes. Possible steps include: measuring the gift size, cutting paper and ribbons, folding and wrapping, and decorating the present with ornaments. We offered the following supplies:

- *Present*: a rectangular gift box of size $9 \times 2 \times 4.5$ inches.
- *Tools*: scissors, utility knife, ruler, pencil, double-sided tape, transparent tape, and glue.
- *Wrapping paper*: a variety of wrapping paper rolls including plain, patterned, and textured.
- *Decorations*: ribbons (curling and fabric) in multiple colors, gift bows, stickers, and message cards.

To help them understand the context of the study, the participants were asked to watch three videos before visiting our lab. The videos were selected from the formative user study.

Procedure and Environment

The study was conducted in a quiet lab environment with static lighting. We used a tripod-mounted Sony camcorder to record the gift wrapping task (Figure 6.12), and a Macbook Pro running OS X and Google Chrome for DemoCut. The laptop was connected to a 30-inch monitor and external mouse and keyboard. Each study session lasted 60-90 minutes.

Introduction (15 minutes). The participants viewed a web-based tutorial that introduced the goal and procedure of the study. In the tutorial, the participants practiced annotating a one-minute demo video with five types of markers, reviewed a system-generated result, and modified video effects in the DemoCut Editing interface.

Filming setup and practice (5 minutes). The participants were asked to plan their gift-wrapping demonstration with any of the provided supplies. The camcorder was positioned either opposite the participants or behind them on the right and was angled down to capture their workspace



Figure 6.12: Our user study setup.

ID	Editing expertise (years)	Footage length	Demo-Cut video length	Final video length	Annotation time (mins)	# and types of markers used for annotation						Ave text length (words)	# of segments	Review & edit time (mins)	# of effects changed
						Total	Step	Action	Supply	Closeup	Cutout				
P1	5	3'51"	2'14"	2'14"	10	20	3	10	5	1	1	3.2	28	8	1
P2	3	7'16"	4'09"	4'14"	16	29	4	16	4	5	0	4	49	11	4
P3	10	10'57"	6'45"	6'45"	18	36	10	18	4	0	4	2.2	57	10	3
P4	2	9'16"	5'12"	5'38"	25	35	6	20	3	4	2	3.3	56	13	6
P5	0	7'17"	4'13"	4'07"	17	38	5	18	7	6	2	3	56	12	5
P6	0	6'28"	3'20"	2'48"	8	24	3	13	6	0	2	3.5	40	9	9
P7	0	10'08"	6'18"	6'02"	21	66	7	35	12	8	4	3.9	92	14	20
P8	0	8'58"	3'05"	3'03"	8	13	4	6	1	0	2	3.6	21	10	2
AVE	2.5	8'01"	4'24"	4'21"	15	33	5	17	5	3	2	3.3	50	11	6

Table 6.3: Quantitative analysis of the user evaluation.

on a conference table. The participants reviewed the camera's point of view and were told that any activities outside of the delineated workspace would not be captured. The participants were free to plan their task on paper or conduct a practice run.

Filming demonstration (10-20 minutes). The participants filmed their gift wrapping demonstration in a single take. The study moderator initiated and terminated the recording, but did not provide additional assistance.

Annotating and editing (30-45 minutes). The participants annotated their video with the DemoCut Annotation Interface and modified the generated video tutorial using the DemoCut Editing Interface.

Review of the final video and discussion (10 minutes). Finally, participants reviewed the final video, completed a questionnaire, and discussed the process with experimenters.

Discussion

All of the participants successfully created a complete video tutorial of their gift wrapping technique using DemoCut during the study session. The average length of the recorded demonstrations was 8 minutes, and the final generated videos were just over 5 minutes long (45% shorter than the raw

footage). On average, participants spent 15 minutes annotating the recordings and added 33 markers. Table 6.3 summarizes several other quantitative results from the study.

Here, we summarize a few key points from the responses to the questionnaire and the end-of-study discussions:

DemoCut interface and workflow. We received strong positive feedback about the DemoCut interface as a whole and the editing workflow that it enables. All participants agreed or strongly agreed that it was easy to annotate a recording using the Annotation Interface, and seven of them found it easy to use the reviewing and Editing Interface. P1 explained, *“This is very simple for beginning users and takes out some of the guess work around learning how to use different layers, speed effects, etc.”*, and P2 described the workflow as, *“super easy and SUPER FAST!”* P6 also appreciated the simplicity of the interface: *“I like this a lot because there aren’t thousands of different buttons to work with.”* In addition, several participants noted how the automated components of the system reduced the amount of effort required to create an edited video: *“I could be lazier and still have a great video cause it did everything for me”* (P8), and *“Pre-segmentation (when it worked well) made it easy to zero in on the portion I wanted to modify”* (P4).

Automatic editing effects. In general, the participants liked how DemoCut automatically removed or condensed parts of their recordings. Their feedback suggests that the automatically generated effects were particularly useful for speeding up repetitive actions like cutting and folding and skipping extraneous actions, such as removing the adhesive sticker from a bow. As P3 noted, these effects were generally successful because DemoCut *“correctly understood parts with no speech but long actions.”* Another participant commented specifically on the fast motion with merged audio effect, and said *“(I) appreciate the automatic speeding up/slowing down of video to match speech.”*

Reviewing and editing. As expected, there were some cases where participants decided to modify the automatic effects. Errors in the audio analysis can cause the narration within a segment to get cut off when fast motion effects are applied. To eliminate these audio artifacts, participants changed the segment effect from fast motion to normal mode. In cases where the narration referred to specific visual events, participants switched from the default fast motion with merged audio effect to leap frog with synchronized audio. Finally, in a few situations, participants decided to skip an annotated segment that they deemed unnecessary or unclear after reviewing the rest of the tutorial.

Quality of generated tutorials. Five of the eight participants said they were satisfied or very satisfied with the video tutorials that they created with DemoCut during the study. The remaining three participants had significantly more video editing experience, and they wanted to further refine their tutorials by adjusting some of the timing and cut points using more traditional low-level editing tools. However, even these participants agreed that DemoCut was *“good for a first pass of editing”* and provided *“helpful “smart” suggestions”* even though the system is *“limited in manual control.”*

Default speed-up effects. The participants noted some limitations with the default editing effects. P5 explained that *“having the speed up of video be the default speed creates a stressful tutorial.”* Some participants pointed out that there are some obvious cases where fast motion with merged audio should not be applied; for example, the effect *“does not work well if the person’s face is showing (the speech and mouth movements would not match up).”* We agree with these comments and plan to use face detection and add an adaptive learner to improve the system.

Annotation guidelines. One observation from the study is that adding too many markers during the annotation phase can hurt the quality of the generated tutorial. Adding markers temporally close together leads to many short segments, and since DemoCut applies a video editing effect to each segment individually, the resulting tutorial may end up transitioning rapidly through several inconsistent effects (e.g., fast motion effects with various playback speeds). One way to address this problem is to make automatic editing decisions that span several consecutive segments. The participants offered a few other suggestions: P4 wonders *“if there are simple tips you could give to the user while recording that would make them more successful,”* and P8 suggested that seeing real-time effects while adding markers might help him understand how best to annotate the video.

6.6 Conclusion

In this chapter, we presented DemoCut, a semi-automatic video editing system that helps users create clear and concise video tutorials of DIY tasks. The key idea behind our approach is to combine rough user annotations with simple video and audio analysis techniques in order to segment the input recording and apply appropriate editing effects. Our small user evaluation suggests that video authors are able to create effective video tutorials using DemoCut, and the qualitative feedback includes encouraging positive reactions to the annotation and editing workflow, as well as the automatic editing effects.

Our implementation is based on several simplifying assumptions that limit generality. We assume a single, static camera position that shows all relevant actions and a quiet indoor environment with constant lighting and little background noise. In order to detect static shots that should be skipped, our video analysis assumes a static background. Our audio analysis assumes that all non-silent sections of audio are narration, but this may not always be the case. Loud non-speech sounds, such as chopping or the sound of a sewing machine, can lead to errors in our editing effect decisions.

As was pointed out by several of our study participants, making effect decisions individually for each segment can lead to inconsistencies in playback speed as the video transitions from segment to segment. A more global approach that looks at all video effects together and enforces smooth transitions between adjacent segments would help address some of these artifacts. In addition to addressing these limitations, we see several promising directions for future work.

Multiple camera footage. We designed DemoCut to work with footage from a single, static camera. One interesting avenue for future work is to consider footage from multiple cameras. Prior work has compared different camera views capturing physical tasks for remote collaboration [81, 177]. Similarly, DemoCut could try to automatically select the best view for each segment based on user annotations as well as the video content (e.g., choosing a zoomed view for closeups, switching to a different view when there are occlusions).

Support viewer’s learning. In this work, we focus on producing well-edited video tutorials. However, we could also imagine generating different output formats, including indexed videos, step-by-step instructions, or mixed media tutorials, similar to those presented by Chi et al. [46]. Another natural extension would be to develop interactive components that monitor user actions and provide

realtime guidance and feedback for general DIY tasks. Follow-up studies to understand viewer's learning experience would be useful for refining the automatic editing effects and interactive design.

Generalize to other instructional video domains. One exciting direction is to explore other areas where our techniques could be applied, such as software learning, music instruction, and video lectures. Each domain may require slightly different analysis and segmentation rules. For example, the system could use a log of executed operations to adjust segment boundaries for software tutorials, or incorporate pitch detection when analyzing music instruction.

Chapter 7

Kinectograph: Body-Tracking Camera Control

A large community of users creates and shares how-to videos online. Many of these videos show demonstrations of physical tasks, such as fixing a machine or demonstrating dance steps. It is often difficult for the authors of these videos to control camera focus, view, and position while performing their tasks. To help instructors produce videos, in this chapter, we introduce Kinectograph¹, a recording device that automatically pans and tilts to follow specific body parts, e.g., hands, of a user in a video with lightweight control. It utilizes a Kinect depth sensor to track skeletal data and adjusts the camera angle via a 2D pan-tilt gimbal mount. Users configure Kinectograph through a tablet application with real-time video preview. We conducted a preliminary evaluations to test the usability of Kinectograph's control interface. All of the participants successfully created instructional videos without assistance. The initial findings suggested that Kinectograph enables instructors to focus on performing their demonstrations, while giving them sufficient camera control at recording time.

7.1 Introduction

Popular online video-sharing websites such as YouTube have enabled the growth of a large community of users who share their knowledge and expertise in video tutorials. How-To videos demonstrate specific skills and procedures for tasks as varied as cooking, building a treehouse, or fixing a machine [206]. These online tutorials help learners observe the manipulations and then put them into practice [205]. However, in recording these videos, instructors often find it challenging to control the camera during demonstration. Working with a cameraman who controls the device and viewpoints ensures that the video captures the movements that the audience would want to see, but it requires having a second person to direct the recording and work closely together with instructors. Many amateur users who mostly work alone, therefore, choose to self-record with one or more cameras. Camcorders can be set on a tripod to capture a static viewpoint, but it is

¹ This work was published at CHI 2013 [44].



Figure 7.1: Kinectograph includes a Kinect camera to track user movement and a motorized dock to pan and tilt the camera so that the user (or their hand) remains centered in the recorded video. Here the device follows the user’s hand while he is illustrating.

hard to make sure whether users’ actions are properly in frame at recording time. An alternative is to wear a head mounted camera to record what the instructors see. This may record unwanted and distractive head movements, making it difficult for the audience to watch. Additional camera views of the overall workspace might be needed to assist learners with understanding the context of demonstrated actions [81].

Seeing these filming challenges, we would like to enable users to gain the flexibility of real-time camera control without requiring a dedicated camera-person. Our goal is to design a device that can automatically track and orient to film the tutorial instructors while providing lightweight manual controls. Existing video conferencing cameras and surveillance tools offer human tracking to provide full or partial automatic viewpoint control. Polycom² designs video conferencing cameras that feature face recognition and voice detection to enable a group of users to talk in an office room setting. This approach assumes people’s faces should be in the frame, which may not be true for instructional videos that focus on actions rather than “talking heads.” Automatic motion tracking is possible to always keep the user in view using visible markers [179] or wearable sensors such

² <http://www.polycom.com/>

as infrared emitters by Swivl³. However, instructors are unlikely to take such approaches to put on visible markers when demonstrating. Researchers have been developing techniques to track specific targets using computer vision, including hands [178], user movements [216], fast-moving objects (e.g., a Ping-Pong ball) [163], or regions in pre-defined spaces [177]. These usually require an expert defining heuristics of space regions or movement classifications ahead of time for the tracking program. On the contrary, we aim at proposing a new approach that does not have these issues, gives users flexibility in a home environment, and provides interactive control over the behavior of the camera tracking.

We propose Kinectograph, a new device that enables semi-automatic camera control for users to self-direct camera orientation for demonstration tasks (Figure 7.1). Kinectograph serves as both the camera and the cameraman. It provides a motorized dock for a Kinect⁴ sensor and a tablet-based user interface (separate from the camera) to switch between tracking regions at runtime based on their needs when demonstrating. Using skeleton tracking to follow the user's movement, Kinectograph automatically pans and tilts the camera in real time. Users can define zoom regions that follow his actions to provide closeup views. Using a Kinect sensor, our system works in a common indoor setting and does not require the user to wear sensors or configure the environments. Kinectograph makes a novel contribution over the prior art in its mixed-initiative approach that offers various levels of automation and control to users at record time.

In Section 7.2, we describe the user experience of recording with Kinectograph. We also describe design and implementation decisions (Section 7.3). Finally, in Section 7.4, we review findings from a preliminary evaluation with 7 participants to study Kinectograph's usability of self-recording tutorials. All of the participants successfully created a demonstration video using our system without assistance and found it easy to interact with.

7.2 Recording Video Tutorials with Kinectograph

Users connect the Kinectograph device to a personal computer and start its server software, then open the Kinectograph mobile control UI (Figure 7.2) on a phone or tablet that they can carry with them during their demonstration. In order to allow How-To tutorial makers as much control over the filming of their tutorial with as little effort as possible, Kinectograph offers the following features on its mobile control UI:

Real-time Video View

In a self video recording, it is convenient for the user to be able to view the current state of their video to make filming decisions. Thus, Kinectograph's UI displays a real-time video feed (currently limited to a low fps) from the Kinect camera.

³ <http://www.swivl.com/>

⁴ <http://www.xbox.com/en-US/kinect>

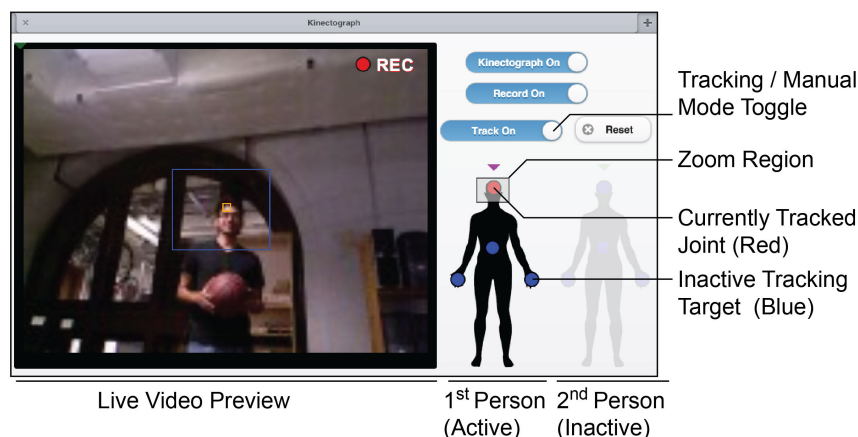


Figure 7.2: Kinectograph UI on a tablet device.

Manual Control

Users can manually control the camera angle by swiping on the preview image to pan and tilt the camera. For example, during a cooking demonstration, users may wish to pan to a shot of the oven to their left or the sink to their right, or tilt down when they open the oven.

Automated Tracking Control

Demonstrations may require users to move around, such as in furniture assembling or personal training videos. Kinectograph’s automated tracking control enables users to focus on their demonstration. Thus, users can switch to automatic camera mode to have Kinectograph track them. Kinectograph can follow one or two actors. Users select which body parts should remain in the frame by tapping on targets of an iconic body outline in the UI - e.g., the head (as in video conferencing systems) or the hands (which may be more important for demonstrations). To keep the entire upper body in view, users can select multiple joints simultaneously. Each selected joint can be deselected by toggling the target button on the iconic body outline.

Early testing showed that because the video feed to the tablet has some latency and a lower framerate, selecting joints on the video feed itself can be difficult if those joints are moving. We therefore chose to display a static, iconic body outline.

Zoom Control

Close-ups of important steps are a common editing technique in demonstration videos. To capture close-ups, users can define a zoom region related to their body by dragging a rectangular area across the iconic body outline - e.g., they can draw a region that captures their hands to focus on their hand motions. Because the camera used has a fixed focal length, our current prototype uses digital zoom (Figure 7.3). By default any joints in the zoomed region are automatically added to the track list.

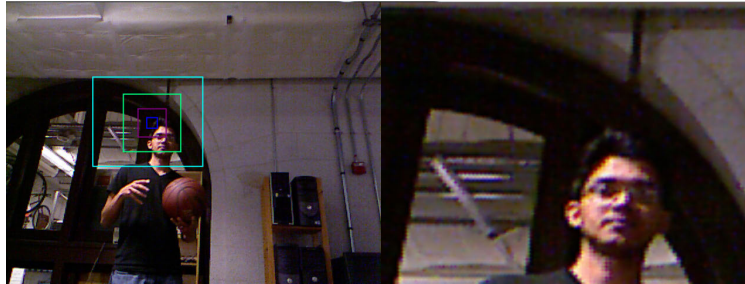


Figure 7.3: Kinectograph tracks and provides a digital zoom view (right) captured from the Kinect camera view (left) in real-time based on user specified area.

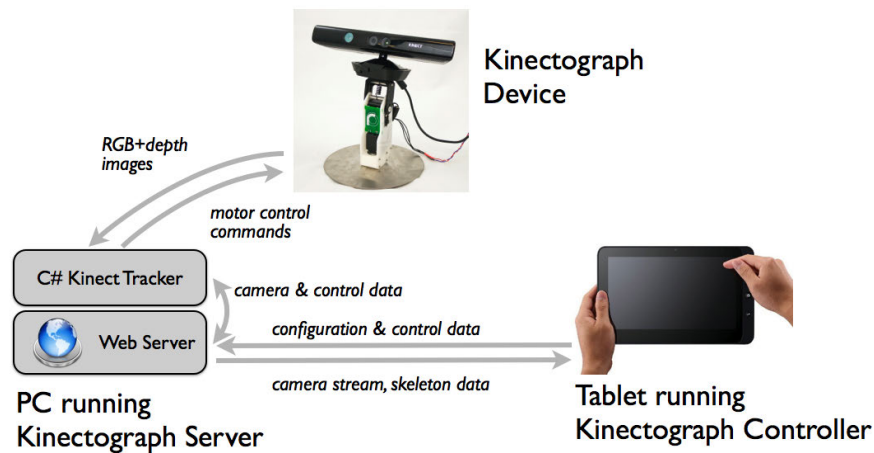


Figure 7.4: Kinectograph architecture.

Reset

Finally, the control UI provides a button to dismiss any joints selected for automated tracking or for target zooming.

7.3 Automatic Tracking Techniques

Kinectograph streams the video view captured from a Kinect camera to a PC. This PC also acts as a web server, publishing the control interface to tablet or phone clients. When a user enables automatic tracking, the PC analyzes user movements using the skeletal data from the Kinect SDK. Based on the user position, it sends appropriate commands to the motorized dock via USB to move the camera (Figure 7.4).

Hardware

To control the camera view, Kinectograph utilizes the motor system and base provided by the Kubi⁵, a dynamic telepresence solution for video calls. This connects to the Kinect with a 3D printed holder (Figure 7.1). The Kubi uses two Dynamixel AX-12 motors⁶, to pan and tilt. Instead of using Kubi's motor control API and microcontroller, we connect Kubi's motors to an external Arduino Mega (ATmega1280)⁷ so we can better customize the control of the Dynamixel motors.

Auto-Tracking Algorithm

Motion Tracking and Servo Adjustment

To track the user position and determine the camera angle, our system analyzes the skeletal tracking data and depth information of a user's body parts received from the Kinect sensor in real-time. Using Kinect enables the user to freely move or turn around, with support of self-occlusion [191]. Currently Kinectograph tracks one or two people. If more people are present, only the two closest are tracked.

Physical Tracking of Joints

Kinectograph angles the camera to position the target, such as a user's hand, in the center of the camera view. When a user is found in the view, it receives the position of the tracked joint located at $\langle pos_x, pos_y, pos_z \rangle$ in the 3D space that represents a x-y coordinate and z as the depth information of the target. The rotation angles are computed in order to align the target joint to the position of the center of view at $\langle center_x, center_y, pos_z \rangle$ on the same X-Y plane as the joint position. The rotation angles θ_{tilt} and θ_{pan} (in degrees) to tilt or pan the camera are:

$$\text{Tilt angle of y-axis: } \theta_{tilt} = \frac{\Delta y}{\Delta z} = \arctan\left(\frac{center_y - pos_y}{pos_z}\right)$$

$$\text{Pan angle of x-axis: } \theta_{pan} = \frac{\Delta x}{\Delta z} = \arctan\left(\frac{center_x - pos_x}{pos_z}\right)$$

Figure 7.5 depicts the geometric relations from the Kinectograph view. Every 10 milliseconds, the top motor turns by θ_{tilt} and the bottom motor turns by θ_{pan} . To avoid extraneous small camera movements, a variable bounding box around the center of the field of view is set such that the Kinectograph only moves once the tracked object leaves the bounding box.

To support multiple target selection, we use the same θ_{tilt} and θ_{pan} formulas as above, except $\langle pos_x, pos_y, pos_z \rangle$ represents the average of all tracked joint positions rather than the position of one tracked joint. If there is more than one actor in the view, users can select joints from the second user. All selected joints are treated equally to be tracked, and thus the camera angle adjustment works as it does in the previous multiple-joint scenario.

⁵ <http://www.revolverobotics.com/>

⁶ <http://www.crustcrawler.com/motors/AX12/index.php>

⁷ <http://arduino.cc/en/Main/arduinoBoardMega>

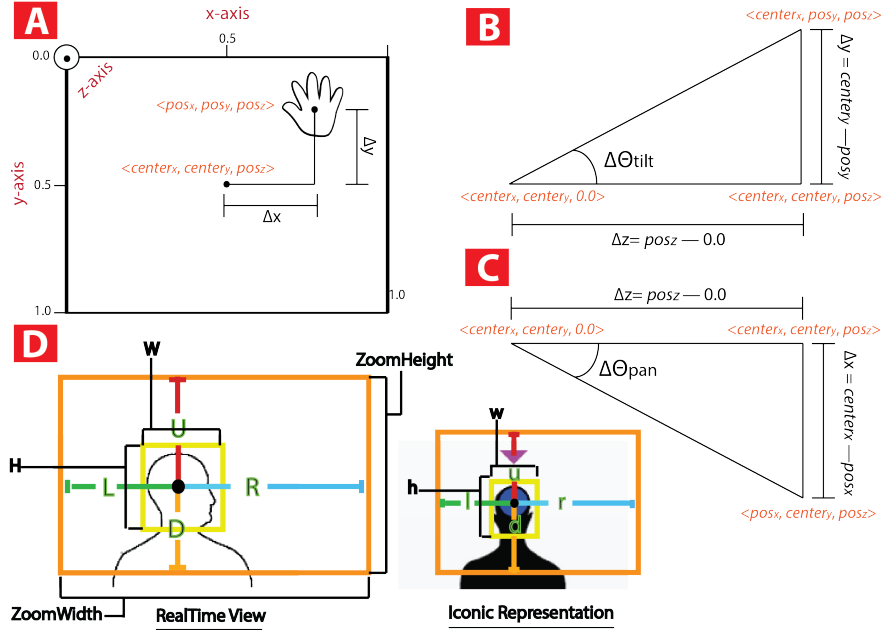


Figure 7.5: Kinectograph tracks the position of the target (A) and computes the tilt (B) and the pan (C) angles in order to center the target. It digitally zooms the camera view based on user specified region on the tablet UI (D).

Digital Tracking and Zooming

Users may find it important not only to center a joint in the view but also magnify it, such as zooming into the hand during a demonstration showing how to stir the ingredients. Thus, Kinectograph allows users to digitally zoom into a specific area with one or more joints. Our algorithm attempts to keep the joint(s) in the same relative position as it was in its initial zoomed in view. This feature works independently of the physical tracking and thus does not require physically controlling the motors or camera. Kinectograph allows the the user to draw a box on the figure in the UI around the joint she wishes to digitally track. The initial dimensions and positions of that translated zoomed region on the live video stream is determined as follows (Figure 7.5D), where:

$$ZoomWidth = R + L = \frac{W}{2} * \frac{l}{\frac{w}{2}} + \frac{W}{2} * \frac{r}{\frac{w}{2}}$$

$$ZoomHeight = U + D = \frac{H}{2} * \frac{u}{\frac{h}{2}} + \frac{H}{2} * \frac{d}{\frac{h}{2}}$$

After the zoomed region's initial dimension has been set, the center of this region is continuously shifted using the following equations: $\Delta X = pos'_x - pos_x$ and $\Delta Y = pos'_y - pos_y$

Implementation

Kinectograph is implemented in C# using the official Kinect API and the Arduino software. Our tablet UI is implemented with the standard Web technologies, including HTML5, CSS3, JavaScript, and jQuery for recognizing touch gestures and communication.

7.4 Evaluation

This section presents user feedback collected from three preliminary studies that we conducted to answer questions: What activities would users find useful to capture using Kinectograph? How well could users create self-directed tutorials with Kinectograph?

Study 1: Demo at an Expo

We demonstrated an initial design of Kinectograph (see Appendix C) at a public exhibition to approximately 60 people. Each participant was allowed to enter our capturing space and experience the device. Based on our observation and conversations collected, we found that people were convinced by the idea as soon as they walked into the scene when Kinectograph started to move along. These questions were often asked: “*How fast was Kinectograph able to follow me?*” and “*Can I switch to track other parts (like my hand)?*”. With the tablet device control, participants soon were successful in controlling the camera. They often walked, ran, and danced to test the tracking. We also learned that people expected the device to provide fast response in various conditions such as turning, rapid change of directions, or partial occlusions (when people were hidden by furniture or large objects).

Study 2: Test on Recording Activities

To understand how Kinectograph can support users in demonstrations, we invited four participants (3 male and 1 female, aged 22-29) who did not join the exhibition to our user study in a home environment. We aimed to explore whether users prefer to watch the video captured by Kinectograph over video recorded with a static camera, and whether Kinectograph can capture complete demonstrations that a static camera cannot achieve.

We first introduced Kinectograph by having participants walk around while the device tracked. We encouraged participants to brainstorm some activities they wanted to record. Once the task was decided, they were asked to set up both a static camera and the Kinectograph with our tablet device and start the recording. There was no time constraint during the study. A short post interview was then conducted, in which we showed the recorded videos from both cameras on a PC.

Table 7.1 shows details of the four tasks and analysis of the recorded videos. We categorized physical activities into three movement types: *Continuous* (user continuously moves around), *Periodic* (user moves, stays, and moves again periodically), and *Occasional* (no clear motion pattern was observed). There were two Continuous and two Periodic tasks that participants designed. The moving range was about 15 feet in a home environment, and participants set the static camera

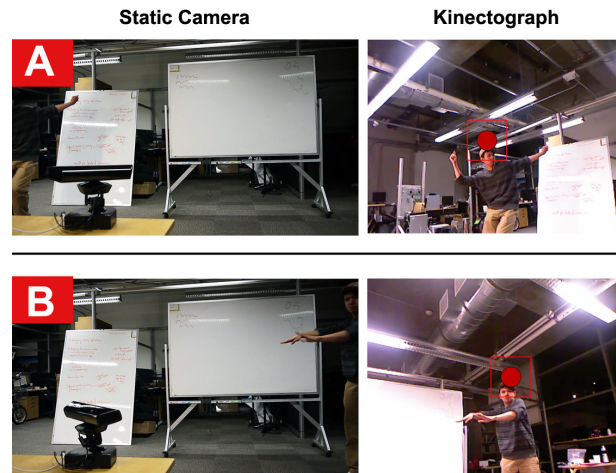


Figure 7.6: Examples of camera views captured by a static camera and Kinectograph at two specific moments in time.

about 8 feet away from the center of their workspace. Participants chose this distance to avoid out-of-frame problems with the static camera: “*The distance was chosen so that all of the activity could be captured*” (P4). Kinectograph was placed 6 feet away on a tabletop by the experimenters to capture the participant’s whole body. Participants were allowed to adjust the camera angle via our tablet UI before recording the demonstration.

All the participants chose to track their heads, but note that their activities involved frequent turning where pure face recognition might fail. Participants did not change this setting during the performance, although they were allowed to. P2 changed to the manual mode for testing, switched back, and then continued the activity. The average video length is one and half minutes long.

All the participants agreed or strongly agreed that Kinectograph captured what they intended to show, while only half of them agreed that the static camera captured as expected. The main reason was the limited static camera angle; in three tasks, participants moved out of the static camera view more than once. Figure 7.6 shows two examples where our system captured what the static camera missed. It was worth noting that although P3 had set and confirmed the viewpoint before recording, he was not aware that he shortly but frequently (9 times) went over the boundaries when he was

User	Recording Task	Location	Movement Type	User Moving Range	Static Camera Distance	Video Length	Out of view from static camera		Out of view from Kinectograph	
							Counts	Length	Counts	Length
P1	Hip-hop dance	Meeting room	Continuous	16ft	7ft	1’45”	3	15”	0	0”
P2	Workouts	Computer room	Periodic	15ft	8ft	1’30”	0	0”	0	0”
P3	PiggyBack ride tutorial	Living room	Continuous	15ft	8ft	2’00”	9	15”	0	0”
P4	Fight scene	Living room	Periodic	15ft	8ft	0’55”	2	5”	0	0”

Table 7.1: Task information and results collected in the preliminary user study.

demonstrating. He explained that he preferred using Kinectograph because it “*kept us in the center of view no matter how we moved around.*” This shows that Kinectograph successfully ensured the activities would be captured and therefore enabled users to focus on their tasks.

Study 3: Self-Recording Activities

Finally, we conducted a study to measure whether users could film an entire demonstration video using Kinectograph with minimal aid. We recruited seven participants (3 male and 4 females, ages 20-33) from a university to record multi-step tutorials in a lab environment. Four had filmed a video before but only one had filmed a video without the assistance of others. Each participant was compensated with a \$10 giftcard. Each session lasted about 30 minutes long. Below we describe the procedure of this study: **Introduction (5 minutes)**. Participants first went through an online documentation to learn the Kinectograph features.

Training (10 minutes). Experimenters guided participants through a series of interactions highlighting each of our core features. Participants were asked to operate each feature through our tablet UI with the support of experimenters.

Testing (5 minutes). We asked participants to film a basketball tutorial using Kinectograph. They were asked to introduce actions including passing, catching, and tossing a basketball. Participants acted as both an actor and a director, i.e., they fully controlled the camera and performed the demonstrations without any assistance. A series of nine subtasks were designed for participants to exercise the following features: manual mode (pan/tilt), tracking mode (to track single and multiple body joints), and zooming. In particular, one of the subtasks involved two users in the view. Experimenter walked in the view for passing the ball when participant invited. To help participants understand these activities, we provided a storyboard with high level instructions for filming (e.g., “zoom into your face”, “pan to the basketball”, or “track your head and walk around”) without explicitly listing which Kinectograph feature to use. During the recording, we captured Kinectograph’s rendered video view.

Questionnaire and Debrief (10 minutes). Finally, we asked participants to watch the recorded video in full and answer a questionnaire, regarding the ease of use of our interface and open-ended questions. We monitored the number of attempts it took to complete each filming task of the tutorials.

Results

All of the 7 participants successfully created a self-directed tutorial using our system. No users failed to complete any of the 9 subtasks. Each participant reattempted at most 2 subtasks, mostly to reselect a zoom area. The average video length was 3.5 minutes. Overall, participants rated the ease of use of the system as $\mu = 4.1$ on the 5-point Likert scale. All participants were able to manually pan and tilt the camera by swiping as intended. Participants stated that this was easy to control with the UI ($\mu = 3.6$, $\sigma = 1.0$). They also successfully enabled the tracking mode and had Kinectograph track their head and hands. Participants stated it was easy to enable tracking ($\mu = 4$, $\sigma = 1.3$) and rated their satisfaction with the system performance as $\mu = 4$, $\sigma = 0.8$. Participants stated that “It

was easy to select a body part of choice” (P6), and that “ (...Kinectograph) could center the screen very well, and accurately tracked the person” (P7). Four users specifically mentioned that zooming was one of the features that worked well. Participants were satisfied with their video recording ($\mu = 4.1$, $\sigma = 0.7$).

We also learned some important shortcomings from the study. Notably, the pan-tilt motors that our previous Kinectograph prototype uses is under-dimensioned, which leads to oscillation (camera shake) when Kinectograph performs large amplitude pan movements. This is less noticeable at further distances, but becomes especially problematic when the users zooms in on small regions. Learning this effect, we have removed this issue with our current use of Kubi’s servos, which provide smooth motor control.

Latency in video streaming to the tablet device hampered usability. As P4 stated: “*The lag made it difficult for me to move the kinectograph smoothly [during manual control]*”. Participants also suggested alternative control modes other than a tablet while engaged in bi-manual tasks: “*Certain demos require the use of multiple hands, meaning that I can’t carry the iPad during the demo if i wanted to change the point of focus on my body. It would be nice if there were gestures i could do to switch the point(s) of focus without having to use the iPad.*” We found this concept interesting and plan to explore in future work.

Chapter 8

DemoDraw: Motion Illustrations from Demonstration

Illustrations of human movements are used to communicate ideas and convey instructions in many domains, but creating them is time-consuming and requires skill. In this chapter, we introduce DemoDraw¹, a multi-modal approach to generate these illustrations as the user physically demonstrates the movements. In a Demonstration Interface, DemoDraw segments speech and 3D joint motion into a sequence of motion segments, each characterized by a key pose and salient joint trajectories. Based on this sequence, a series of illustrations is automatically generated using a stylistically rendered 3D avatar annotated with arrows to convey movements. During demonstration, the user can navigate using speech and amend or re-perform motions if needed. Once a suitable sequence of steps has been created, a Refinement Interface enables fine control of visualization parameters. In a three-part evaluation, we validate the effectiveness of the generated illustrations and the usability of DemoDraw. Our results show 4 to 7-step illustrations can be created in 5 or 10 minutes on average.

8.1 Introduction

In sports, dance performance, and body gesture interfaces, movement instructions are often conveyed with drawings of the human body annotated with arrows or stroboscopic effects [57] (see Figure 8.1 for examples). These *illustrations of human movements* are also used within HCI to convey new user experiences in papers and storyboards [36]. When designed well, these illustrations can precisely depict the direction of motion while excluding unnecessary details such as clothing and backgrounds [57].

We found that both professionals and non-designers create these kinds of illustrations, but the methods they use are commonly time-consuming and not amenable to iteration and editing. The typical workflow is to prepare the physical scene, pose and photograph actors, and create annotated illustrations from the source photos. Even with the photos, producing effective depictions of the

¹ This work will be published at UIST 2016 [47].

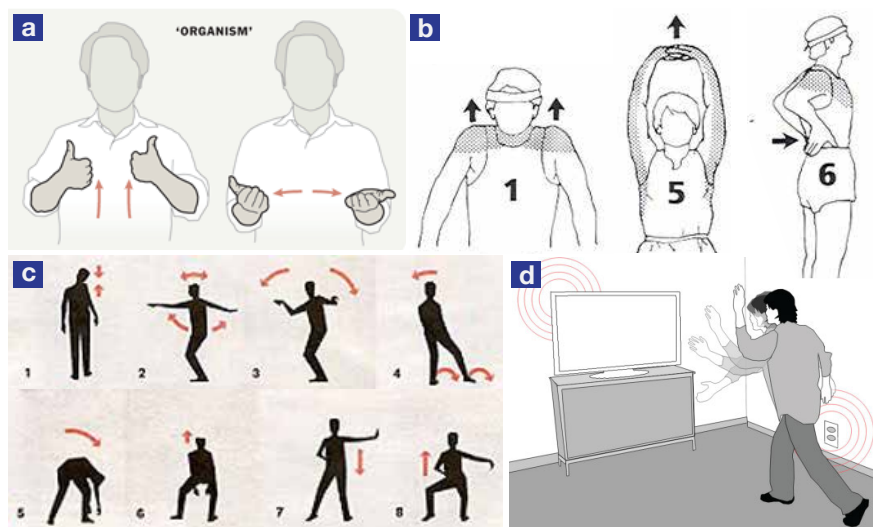


Figure 8.1: Examples of manually generated human movement illustrations: (a) for sign language [56]; (b) for weight training [8]; (c) for dance steps [unknown]; (d) for a gestural interface [54].

actors with integrated motion arrows and/or stroboscopic overlays takes considerable time and skill. Overall, the entire authoring process can take from 10 minutes up to several hours. Moreover, it can be difficult to identify the appropriate pose and viewpoint for the source photos before seeing the resulting illustrations. For example, one may choose to exaggerate or change the orientation of a hand gesture after seeing the illustrated motion. Unfortunately, making such adjustments often requires starting over again with new source photos.

To address these challenges, we propose DemoDraw, a system that enables authors to rapidly create step-by-step motion illustrations through physical demonstration (see Figure 8.2). DemoDraw offers two key advantages over existing workflows. First, our system automatically renders characters and motion arrows based on demonstrations, which significantly reduces the amount of time and effort required to create an illustration. Second, DemoDraw helps users iteratively refine demonstrations to produce effective depictions. In our system, users can quickly add, replace, preview and modify demonstration takes.

Authoring proceeds in two modes: *Demonstration*, performed using body motions and voice commands; and *Refinement*, which uses a desktop interface. The user first physically demonstrates desired motions in front of a Kinect RGB-D sensor. As in current instructional practice, they simultaneously speak during important parts (e.g., teaching dance moves with “one, two, three, four”). The motions are then mapped to a 3D human avatar rendered as a black-and-white contour drawing, a common style identified in our survey of illustration practices. An algorithm analyzes speech and motion streams to segment motions into illustration figures with key frames. Salient joint movements are automatically identified and rendered as motion arrows overlaid on the stylized body drawing (Figure 8.4c). With this *Demonstration Interface*, segmented motions can be reviewed and re-recorded using speech commands. In addition, the annotation style and placement can be

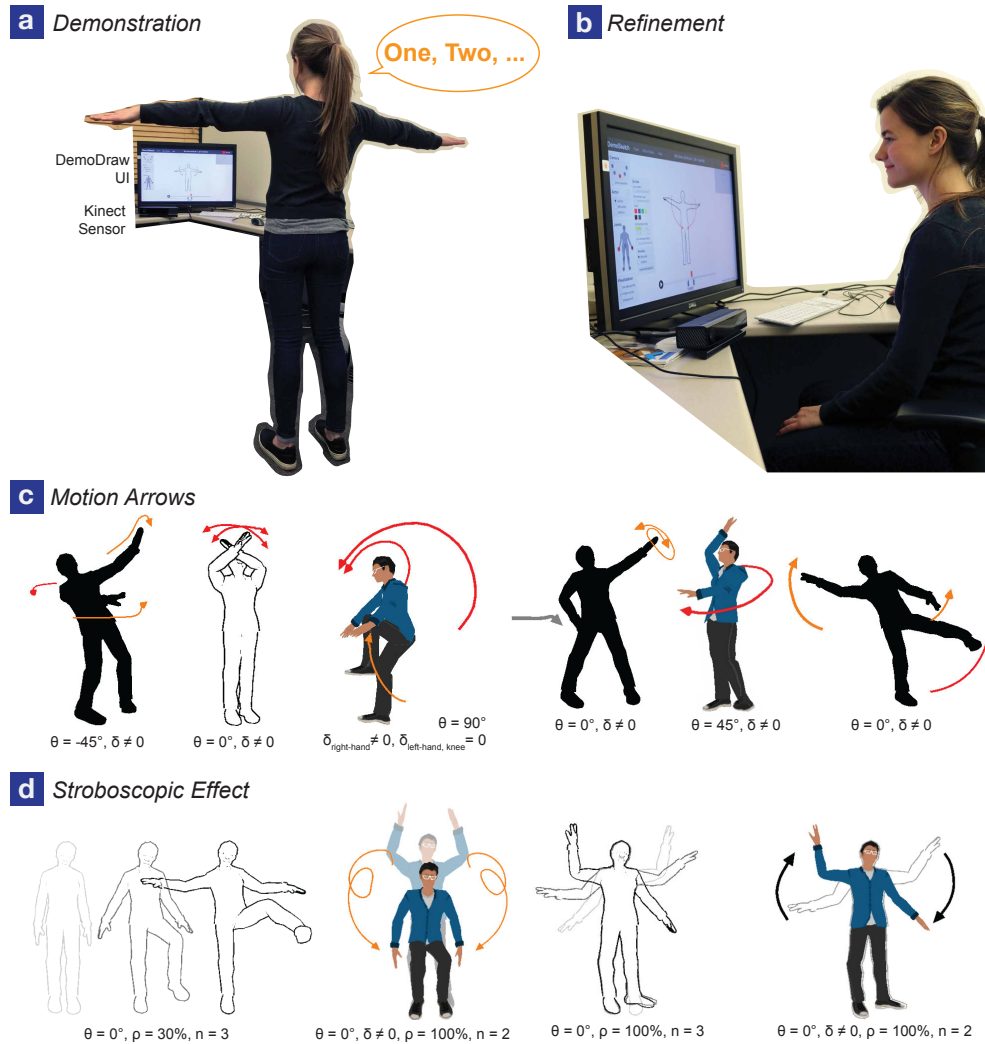


Figure 8.2: DemoDraw’s authoring interfaces and results: (a) multi-modal *Demonstration Interface* to capture motion, verify results, and re-perform portions if needed; (b) conventional *Refinement Interface* for refinement and exploring other visualization styles; (c-d) examples of illustration styles (annotated with camera viewing angle θ , motion arrow offsets δ , stroboscopic overlap ratio ρ , and numbers of intermediate frames n).

adjusted, camera angles moved, and alternate visualization styles explored in a mouse-driven GUI *Refinement Interface* (see Figure 8.5a). A three-part evaluation with 14 participants shows that DemoDraw’s illustrations are effective and amateur authors can use the Demonstration Interface and Refinement Interface to proficiently create motion illustrations with various levels of complexity.

DemoDraw integrates physical demonstration and authoring into one interactive workflow. Our work is the first to generate human movement illustrations by demonstration, refinement, and editing as an iterative process. Our work includes the following specific contributions:

- An approach to generate human movement illustrations by direct physical demonstration and interactive rendering.
- Multi-modal interaction techniques to record, review, retake, and refine demonstration sequences.
- Methods to automatically analyze 3D motion data with speech to generate step-by-step annotated illustrations.

8.2 Related Work

Our work is related to research in authoring by demonstration and motion visualization techniques.

Demonstration-Based Authoring

User demonstrations have been harnessed to generate explanatory, educational or entertainment media in domains including software tutorials [24, 91], animation [16, 105], 3D modeling [223], or physical therapy [220]. For many of these systems, captured demonstrations are treated as fixed inputs that are then processed using fully or semi-automated techniques to produce a visualization. Work that falls into this category includes: generating step-by-step software tutorials from video or screen recordings with DocWizards [24], Grabler et al.’s system [91], and MixT [46], and automatically editing and annotating existing video tutorials with DemoCut [50]. This workflow is similar to graphics research that transforms existing artifacts into illustrations or animations. Examples include: using technical diagrams to generate exploded views [136], mechanical motion illustrations [154], or Augmented Reality 3D animations [155]; using short videos to generate storyboards [84]; creating assembly instructions by tracking 3D movements of blocks in Duplo-Track [98]; and closely related to our work, using existing datasets of pre-recorded motion capture sequences to generate human motion visualizations with systems by Assa et al. [10, 11], Choi et al. [51], and Bouvier-Zappa et al. [31].

Animation is one domain where demonstration is often incorporated into the authoring workflow in a more interactive manner. For example, GENESYS [12], one of the earliest computer animation systems, allows users to perform motion trajectories and the timing of specific events with sketching and tapping interactions. Performance-based animation authoring remains a common approach, and recent work shows how physical props can be incorporated to support layered multi-take performances [65, 97] and puppetry [16, 105].

While the primary goal of performance-based animation systems is to accurately track and re-target prop motions to virtual characters, DemoDraw focuses on the mapping from recorded body movement demonstrations to static illustrations conveying those motions. Some previous systems have also mapped body movement to static media: BodyAvatar [223] treats the body as a proxy and reference frame for “first-person” body gestures to shape a 3D avatar model and a Manga comic maker [141] maps the body pose directly into a comic panel. Systems using interactive guidance for

teaching body motions are essentially the inverse of DemoDraw. Examples include YouMove [7] that teaches moves like dance and yoga, and Physio@Home [201] that guides therapeutic exercises.

Motion Visualization

Several of the systems above focus on developing automated algorithms to visualize various dynamic behaviors, such as mechanical motion [136, 154, 187], motion in film [84], molecular flexibility [33], and human movements [10, 31, 51]. Much of this work is inspired by formalizing techniques and principles for hand-crafted illustrations [4]. Bouvier-Zappa et al.’s [31] automatic approach visualizes large collections of pre-recorded motion capture sequences. We support many of the same visualization techniques, including motion arrows, overlaid ghosted views, and sequences of poses, but we introduce an interactive approach for authors to create illustrations for particular motions to share with others. Since such demonstrations often involve mistakes and repeated takes of the motion, DemoDraw supports interactions to help authors review and retake portions of their demonstrations. Moreover, the interactive nature of DemoDraw enables more fine-grained controls for adjusting visualization parameters and compensating for idiosyncratic characteristics of automated algorithms.

8.3 Principles and Methods

To understand motion illustration design and production, we surveyed related literature, studied collected examples, and interviewed individuals who create such illustrations.

Design Principles

Cutting [57] argues that superimposing vector-like lines, often called “actions lines”, on an image satisfies four important criteria: it evokes a feeling of motion, the object undergoing motion is clearly represented without deformation, the direction of motion is clear, and the magnitude of motion is conveyed with reasonable precision. To complement this metaphoric representation, Cutting also argues for the more literal method of multiple stroboscopic images, which satisfies all criteria except clear motion direction. McCloud [151] provides further arguments and examples for using these methods in the field of comic illustration, and notes communication benefits when they are combined.

To examine how professional illustrators use motion lines and stroboscopic images, we gathered examples from sources like user manuals, gesture-based games, safety guides, illustration compendia (e.g., [153]) and how-to books (e.g., [92]). We found Cutting’s notion of vector-like lines are almost always rendered with an arrowhead in a variety of styles (heads, weights, colors) with strokes typically two-dimensional, smooth, and offset to avoid occluding the object. Stroboscopic images can be overlapping or spatially distributed, and change in transparency or shading to convey time. The most common style for depicting the object undergoing motion is a simplified black-and-white contour drawing, but filled silhouettes and flat-shaded colour can also be found – using full color

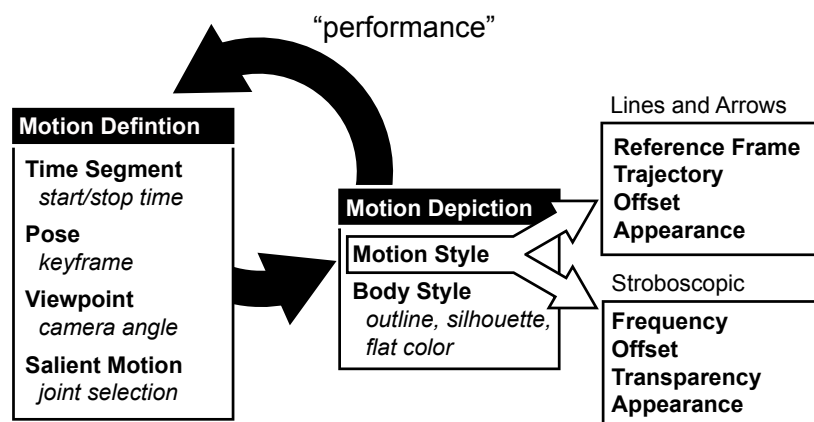


Figure 8.3: Canonical authoring workflow consisting of a Motion Definition task then a Motion Depiction task. Design decisions associated with a task are shown in bold with design parameters in italics.

photographic detail is rare. By carefully removing extraneous details, such techniques help readers focus on only the salient motion information.

Interviews: Methods Used In the HCI Community

Conveying movement for interaction is common in HCI publications. We found 100 motion illustrations in 58 recent papers. To understand current creation methods, we conducted video interviews with six Human-Computer Interaction researchers with experience creating motion illustrations.

Findings. All interviewees used a similar methodology to create motion illustrations: they took still photographs of people performing actions, traced outlines using Adobe Photoshop (4/6) or Illustrator (2/6), then added graphic annotations to convey motion. All mentioned that it was time-consuming to set up scenes and poses, take and trace photos, then add details like arrow placement while maintaining a consistent style. Typical creation times were estimated between 10 minutes to a few hours. They also noted how difficult it was to make adjustments: changing the pose or viewpoint essentially meant starting over again with new source photos and re-tracing. Yet, identifying the best pose and viewpoint ahead of time is difficult and it often took several iterations to yield an illustration suitable for publication.

Design Space Goals and Workflow

Based on the observations above, we derive a canonical workflow to motivate our system's central design goal. Authors face two primary illustration tasks (Figure 8.3): *defining the motion* for portraying movements like the view of the body and salient moving joints; and *exploring a style of motion depiction* by choosing styles like lines-and-arrows or stroboscopic, then adjusting related

style parameters. These tasks and the underlying design parameters are highly interdependent, so authoring motion illustrations is necessarily an iterative process. This means that changes to one task parameter often leads to re-evaluating and changing the other. The problem with current methods, is that movements are mostly “performed” using a time-consuming process of taking photos and manually tracing them. Therefore, the central design goal of our system is to make motion definition low effort and iterative via interactive demonstrations.

Designing a system to capture interactive demonstrations of *any* body movement also poses an input challenge. Since body movements form the demonstration itself, also issuing application commands with a body gesture introduces ambiguity. Using a hand held device, touch screen, or any conventional input is not ideal since performing requires open space and full freedom of movement. For these reasons, we use a multi-modal voice and gesture interaction style traced back to Bolt’s Put-That-There [29]. Like Bolt, we use voice for commands like “*start*” and “*stop*” with body movements providing command parameters in the form of the recorded demonstration, and for setting parameter context with utterances like “*one, two, three, four*” to label step-by-step segments.

8.4 Creating Illustrations with DemoDraw

DemoDraw is designed for non-experts who cannot effectively or efficiently create concise motion illustrations motions using existing tools. To provide an overview of how the system works, we present a scenario in which a motion illustration author, Marie, creates instructions for an 8-step dance tutorial.

In her living room, Marie begins using DemoDraw with the Demonstration Interface shown on her television by standing in front of a Kinect. In the center of the display, an avatar follows her movements in real-time (Figure 8.4a). This avatar is shown as an “outline” figure, but she could always change to different rendering effects like “silhouette” or “cartoon,” or select a different 3D human model later using our Refinement Interface (Figure 8.5a).

Recording. Marie starts recording her physical demonstration with the voice command “*Start.*” After a 3-second countdown, DemoDraw captures the position, orientation, and depth distance of her body (using Kinect’s simplified 25 body joint model). While demonstrating dance moves, Marie verbally indicates the count of each step with “*one, two, three, and four,*” just like she does when teaching a dance. The specific utterance is not constrained, Marie could use words like “*right, left, shake, and clap.*” A speech recognition engine captures these labels with timestamps and displays them in the interface (Figure 8.4b). Marie finishes recording by saying “*Stop*”.

Reviewing and Re-Recording. After recording, DemoDraw automatically segments the motion around the speech labels and identifies salient joints. An illustration of the first step of Marie’s demonstration is rendered with motion arrows, showing the path of the most salient joints. Figure 8.4c presents an example illustration that shows how her right hand waves from bottom to the top, and the left on the opposite direction. She also notices three panels emerged: A timeline below shows the start, end, and key frame points used to generate the current illustration, a side panel shows the visualized joints; an step-by-step overview of step snapshots is created and added to a motion sequence list.

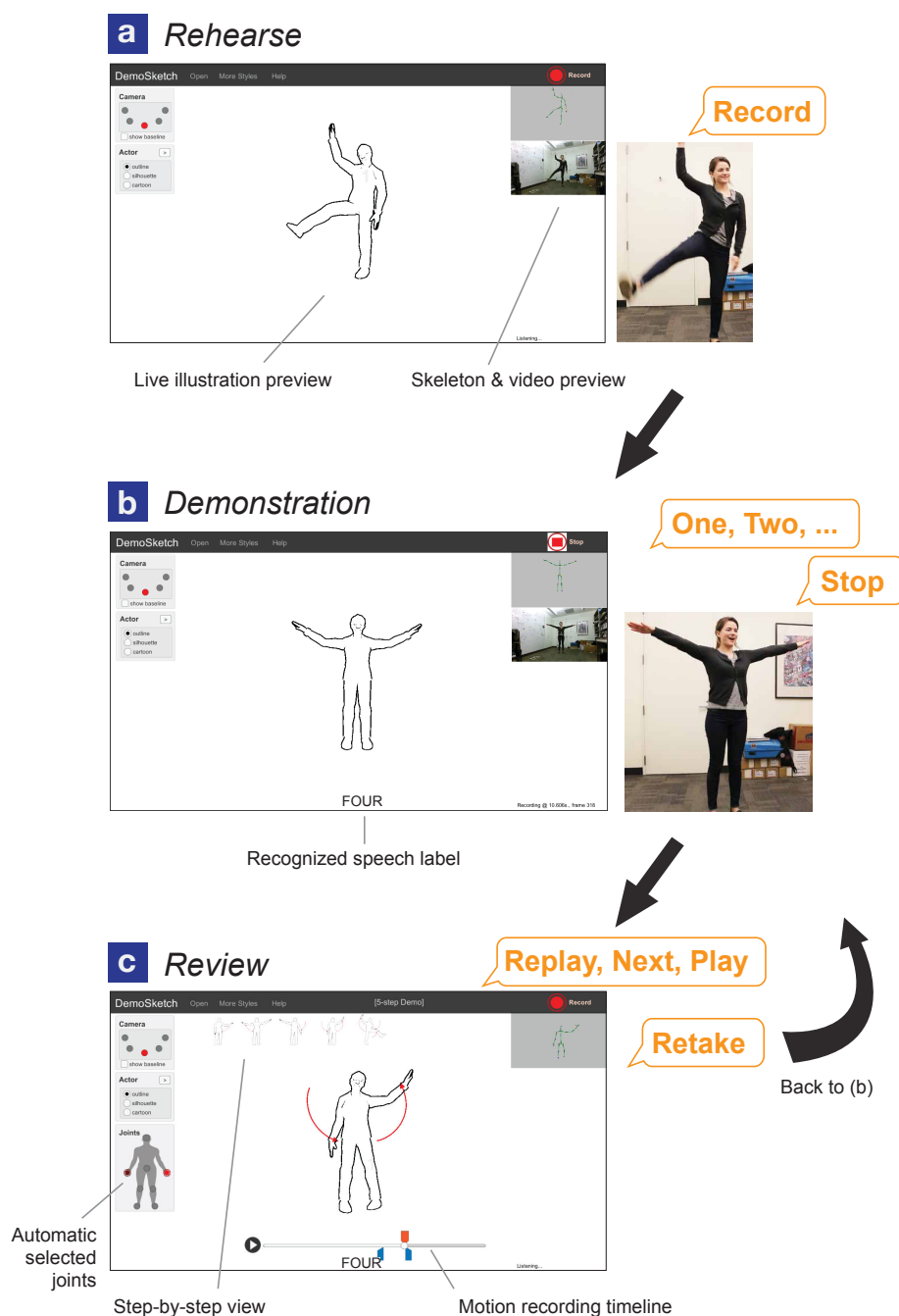


Figure 8.4: DemoDraw authoring UI: Using the Demonstration Interface, an author sees an avatar following her real-time movement (a). During recording (initiated by voice command “Start”), real-time feedback shows the speech labels (b). Once a recording is completed by voice command “Stop”, the motion visualization and a timeline are immediately available (c) for the author to review, and a step-by-step overview will be generated.

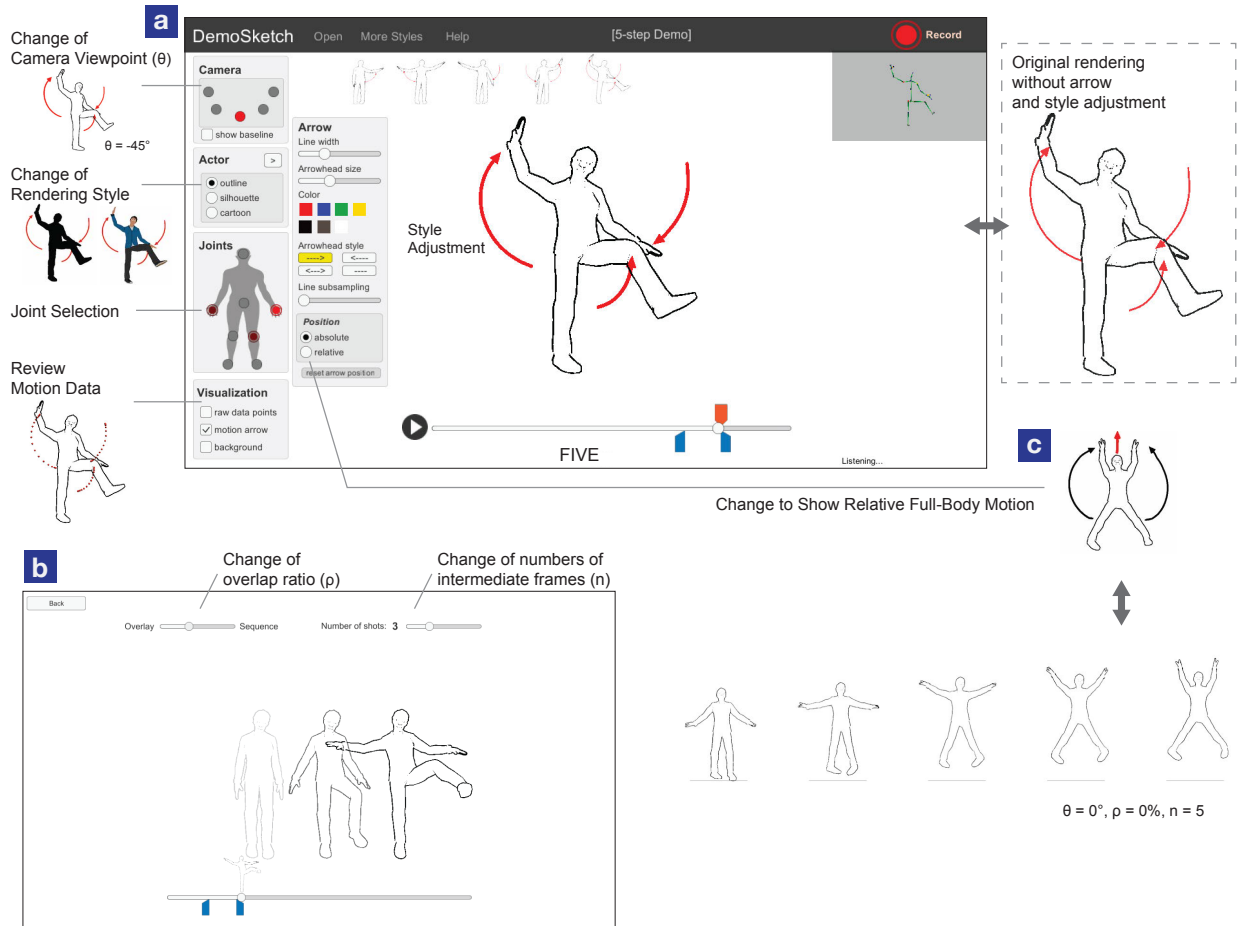


Figure 8.5: Using DemoDraw’s Refinement Interface, the author can refine the visuals (a) and explore more illustration effects (b, c).

Marie can navigate to other illustrated steps by either saying “Next” or “Back”, or repeating one of the words she said during recording (like “three”) to skip to that corresponding step. To play an animation showing her continuous motion, she can say “Play” to play the current step only, or “Replay” to play the entire motion recording with each step visualization highlighted.

Once Marie reviews the steps, she realizes she should have exaggerated the hand motion in step 4. By saying “Retake Four,” Marie can re-record a partial sequence of movements including that step (e.g., redoing and saying “Four” and “Five”). When she ends the re-recording with “Stop”, the old illustration for that step is replaced with a new one (step four in this example) generated using the new motion recording.

Motion Depiction Adjustments. Once Marie is satisfied with her demonstration, she walks out of the capture area to her desktop computer. The system automatically switches to the Refinement Interface by revealing post-processing panels in a standard graphical user interface (Figure 8.5a). Using this interface, Marie can adjust several design parameters: the arrow appearance can be

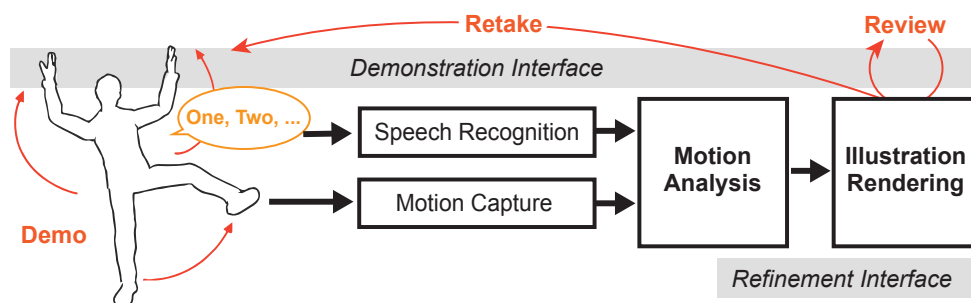


Figure 8.6: DemoDraw system components and pipeline.

refined, including line width, arrowhead size, and color; the arrow offset can be adjusted with direct manipulation dragging; the camera viewpoint can be adjusted by orbiting the camera to a side or three-quarter view; the joints used for motion paths can be added or removed using a panel; and the smoothed motion trajectory can be toggled on and off. She could also select a different key pose and adjust the start and end times of a motion segment by dragging the markers on the timeline. In addition, Marie could explore other illustration styles like stroboscopic rendering by selecting numbers of intermediate frames and how they render in one diagram (Figure 8.5b). These results can be exported to image files containing the final motion illustrations.

8.5 Generation Pipeline

DemoDraw has four main components (Figure 8.6): a *motion capture* engine to record joint data from the author’s demonstration and apply it to a 3D avatar; a *speech recognition* engine to process speech input for commands and motion labels; a *motion analysis* algorithm to partition recorded motion and identify salient joint movements for each illustration segment; and an *illustration rendering* engine to visualize the avatar and motion segments with different effects. These components combine into an interactive and iterative system pipeline to translate demonstrations into motion diagrams. A notable technical contribution is our motion segmentation algorithm combining speech labels and joint motion streams.

DemoDraw is implemented using C# in Unity 5. It runs interactively on a Macbook Pro with Windows Bootcamp (2.5 GHz Intel Core i7 processor and 16 GB memory). Below we describe the design and implementation of each component.

Motion Capture

In support of our design goal to enable low-effort iteration within tasks, the motion capture component provides real-time feedback during demonstrations so authors can monitor their performance accordingly. We capture position and joint angles of a simplified 25-joint skeleton using a Kinect2

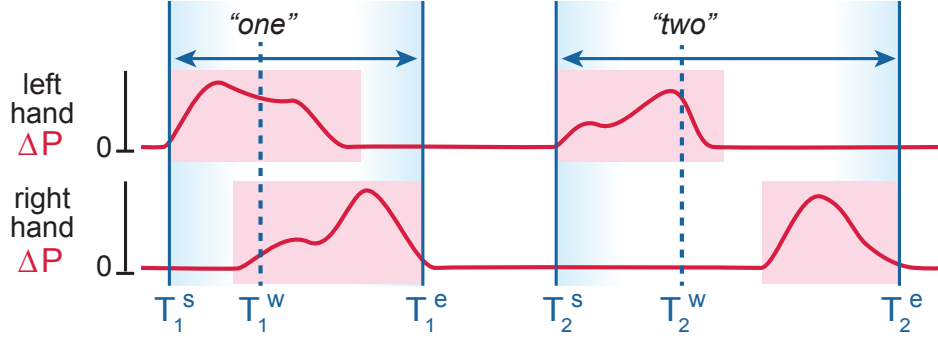


Figure 8.7: Illustration of motion analysis algorithm (two joints shown due to space): significant moving periods of joint movements (pink) are mapped to speech labels to define motion segments (blue). Note the right hand period is mapped to “two” because it begins shortly after the left hand period.

sensor and the Kinect SDK 2.0. The real-time joint data is applied to a generic 3D human model (an “avatar”) using forward kinematics enabled by a modified Unity asset².

Speech Recognition

Speech is used when recording a demonstration to label motions (e.g., “one, two, ...”) and for recording and navigation commands (e.g. “Start, Stop, Retake” or “Replay, Next, Play”) – see Figure 8.4 for the speech commands that DemoDraw supports. We recognize both types of speech using the Microsoft speech recognition library³ to process audio captured by the Kinect microphone array. During recording, the start time, duration, and confidence of each motion label are logged for use in the motion analysis algorithm.

Motion Analysis

Our motion analysis algorithm translates a multi-part demonstration recording into a sequence of labeled time segments, each with one or more salient joint motions and a keyframe of joint positions for a representative body pose (see Figure 8.7 for an illustration of the approach). Formally, given a set of n speech labels $\{w_1, w_2, \dots, w_n\}$ that end at latency-corrected times $\{T_1^w, T_2^w, \dots, T_n^w\}$, our algorithm associates each speech label w_i with a *motion segment*, of which the start and end time are denoted as $[T_i^s, T_i^e]$ where $T_i^s \leq T_i^w \leq T_i^e$. Each motion segment includes a set of k salient joints $\{j_i^1, \dots, j_i^k\}$ and keyframe time T_i^{key} between $[T_i^s, T_i^e]$. It is then sent to the Illustration Rendering engine to create a motion illustration in a multi-part sequence.

Human motion segmentation and activity understanding has been well studied in computer vision and graphics [2]. We adopted a spacetime approach to identify salient motion sequences in

² <https://www.assetstore.unity3d.com/en/#!/content/18708>

³ <https://msdn.microsoft.com/en-us/library/hh361572>

3D space. However, in our scenario such as dancing, movements may not necessarily encode a semantic meaning for automatic recognition, such as “walking” or “throwing (a ball)” in previous research. Therefore, our approach combines the user’s speech labels, similar to a scene segmentation method used in DemoCut [50]. We make two assumptions about the synchronized data streams of speech labels and joint movements: 1) authors make short pauses between motions to be grouped, i.e., $T_i^e < T_{i+1}^s$, and 2) the speech label utterances overlap or closely occur with at least one joint motion. These assumptions are practical since authors often pause for a moment to prepare for demonstrating the next movement in a step-by-step sequence.

Motion Segmentation. To determine a motion segment of $[T_i^s, T_i^e]$ for each speech label w_i that ends at T_i^w , we begin by identifying all *moving periods* of significant joint movements (pink rectangles in Figure 8.7) for 8 joints J : the 5 end-effectors (head, hands, feet), 2 knees, and the body root. To filter jittery movements, joints are considered moving if smoothed inter-frame differences in absolute Euclidean distance are greater than a threshold. Specifically, for each joint $j \in J$ of a frame r at time t , the average difference in position between two adjacent frames $\Delta P = |P^r - P^{r-1}|$ is computed over the subsequent half second (15 frames). If this moving average is greater than $0.05m/s$, then joint j of a frame is labeled as “moving”, marked as m_j^r . This is repeated on all frames and all joints. Next, of the entire motion recording for joint j , we combine all the consecutive $\{m_j^r, m_j^{r+1}, \dots\}$ into a joint moving period M_j .

Once a list of moving periods $\{M_j^1, M_j^2, \dots\}$ for joint j is determined, we begin labeling each M_j^m at $[T_m^s, T_m^e]$ to map to a speech label w_i at time T_i^w where $T_m^s \leq T_i^w \leq T_m^e$. In other words, the speech utterance occurs during or near to a joint movement (illustrated as dashed lines crossing pink rectangles in Figure 8.7). After all moving periods are mapped to speech labels for all major joints in J , the start and end time $[T_i^s, T_i^e]$ of the motion segment for label w_i are set to the minimum start time and maximum end time across all mapped joint movement periods.

Joint Salience Identification. The salient joints $\{j_i^1, \dots, j_i^k\}$ are defined by the set of all joints that were mapped based on significant moving periods.

Key Pose Selection. A key pose is used to represent a motion segment in an illustration. Based on our informal experiment, it is often the end state of movements as motion arrows are pointed toward this end goal (see the Figure 8.6 for example). Therefore, we set a key pose at the end of a motion segment, i.e., $T_i^{key} = T_i^e$.

Motion Retake. When retaking a partial demonstration with one or more speech labels $\{w'_i, w'_{i+1}, \dots\}$, the full motion analysis algorithm is run on the new recording. New motion segments then replace the original segments by mapping w'_i with w_i .

Illustration Rendering

The Illustration Rendering engine generates a motion illustration for each motion segment of speech label w_i (bounded by $[T_i^s, T_i^e]$). There are two related rendering tasks: the body pose and the motion depiction style.

Body Pose. The body pose is determined by all joint positions at keyframe time T_i^{key} . We use standard Non-Photorealistic Rendering (NPR) [87] techniques to render the 3D human model in a

stylized manner that abstracts away distracting details. Specifically, we support contour-only, filled silhouette, and flat-shaded rendering styles (see Figure 8.5a left for examples).

Line and Arrow Depiction Style. Based on Cutting’s criteria [57] and our survey of motion illustrations, we use lines with arrowheads as the default depiction style for visualizing joint movements. This style is rendered as follows: For each salient joint of a motion segment, the absolute joint positions in world space over the period $[T_i^s, T_i^e - \epsilon]$ are used to construct a 3D poly-line using Catmull-Rom interpolation. Rather than visualizing the entire path, we set ϵ to be 0.5 seconds to visually point the arrow toward the key pose at T_i^e . Two 3D cones are positioned collinear with the last two polyline positions to form arrowheads for both the beginning and the end of a line. Although the poly-line is 3D, it is shaded to appear 2D. All arrows are colored red by default to contrast with the avatar, a common technique for layering information [209].

For some motions, visualizing absolute joint positions might not be suitable. For example, for a two-foot jump with a two-hand waving motion (see Figure 8.5c), our algorithm will mark all major joints as salient and generate multiple arrows showing the jump movement, but fail to convey the hand waving. Authors can choose to visualize joint motions *relative* to the spine instead, triggering the same motion analysis algorithm described above to be re-run using relative motion. In this way, the same movements would be shown more concisely with a single up arrow (for the overall jump direction) and two curve arrows (for the hand movements).

Other Adjustments. Authors can review the results using the Demonstration Interface or Refinement Interface. With the latter, line weight, arrowhead sizes, and color can be adjusted and re-rendered in real-time using graphical widgets (see Figure 8.5a). Arrows can also be re-positioned to increase the offset (δ) by direct manipulation dragging. Considering some movements cannot be easily seen from the default front camera viewpoint (such as those parallel to the XZ plane, see Figure 8.2c top-right), our UI enables the selection of four other camera angles (θ), including three-quarter front views (45° and -45°) and profile views (90° and -90°), all at the eye level. These discrete choices simplify control, but of course it would be possible to select any viewing angle given the 3D avatar and joint information. By default, 8 main joints are analyzed and illustrated, but any of the 25 body joints can be explicitly selected for illustration using the interface.

Stroboscopic Depiction Style. Cutting [57] noted stroboscopic effects are also effective, and we found examples of illustrations with a sequence of overlaid semi-transparent body poses in our survey. Therefore, authors can select a stroboscopic depiction style in the Refinement Interface (see Figure 8.5b). The style is rendered by compositing multiple semi-transparent renderings of intermediate body poses between T_i^s to T_i^e behind a rendering of the representative pose at keyframe time T_i^{key} . Authors can adjust the number of intermediate poses n (the default is 3 poses) and the horizontal overlap ratio ρ between intermediate pose renderings can be adjusted to stack them up ($\rho = 100\%$) or spread them out ($\rho = 0$ is the default).

Results

The DemoDraw pipeline is capable of generating expressive and clear motion illustrations. In Figure 8.2c, motion arrows show the upper body motion (top left), hand waving back and forth (top middle), and hand circular motion (bottom right). Whole body motions can also be visualized

(bottom left), and can be especially helpful when motions are best viewed from a different angle, such as the side view (top right). In Figure 8.2d, stroboscopic effect depicts the transition from the start pose to the end pose, which can be rendered as a sequence (top left) or in one combined pose (bottom left). A combination of this effect with motion arrows creates a compact, integrated illustration (top and bottom right).

8.6 Evaluation

We evaluated the capability and usability of DemoDraw in three lab-based studies. The first study with 10 participants tested the effectiveness of illustrations generated by DemoDraw (*How well do users understand static motion illustrations?*). The second study with the same participants evaluated the Demonstration Interface for recording motion demonstrations (*Can users generate step-by-step illustrations with our system?*). The third study with 4 different participants evaluated the Refinement Interface for editing a pre-captured recording (*Can users refine illustrations with our system?*). Recall that our survey found current methods (using software like Adobe Illustrator) are time intensive, require design expertise, and make iteration difficult. For these reasons, we did not include a baseline in our evaluations.

Study 1: Illustration Effectiveness

We hypothesized that learners can understand and re-perform motions after reviewing step-by-step illustrations generated by DemoDraw. To validate, we recruited 10 participants (5 females), aged 18 to 33 years ($M=24.3$), from a university and an IT company. Six participants had previously created illustrations (from 5 to 50 diagrams, typically using Adobe Illustrator), but none involved body motion. We first showed the illustrations in Figure 8.1 to introduce the context, then we presented two sets of printed diagrams generated by our the experimenter using our system. There were 16 highlighted joint movements in 8 steps in total (see Figure D.1). For each set, participants interpret the illustrations and performed the movements in front of a video camera.

Measures and Results. We coded each joint movement using the video recordings as follows: (M1) the user moved the correct joint; (M2) the start and end positions were approximately correct; and (M3) the movement was performed correctly (e.g., moving hand straight). Out of 160 re-performed motions across all 10 participants, 85% motions were completely correct (i.e., M1, M2, M3 all correct). On average, each participant performed 87.2% of the motions correctly ($sd=7\%$). No users intentionally moved non-annotated joints. Table 8.1 shows 6 motions that resulted in errors. These motions fall into two classes: temporal sequencing of multiple joints moving in a single step; and cyclic actions (e.g., waving). Some participants could read and re-perform these more complex motions, but there may be limits to what motion diagrams can convey.





		Error type and rate	Intended motion	Performed motion	Participant Explanation
Set I Step 2		M3 (2/10)	Waving hands back and forth	Waving one way (crossing hands)	Didn't catch it
Set I Step 3		M3 (4/10)	Hands circling out and in	Hands circling in and then out (opposite direction)	Didn't catch it
		M1 (7/10)	Squatting	N/A (missed)	(6/7) focused on hand motions and did not see the down arrow; (1/7) noted the arrow but thought it referred to the hands
Set II Step 1		M2 (5/10)	Moving right hand from lower left to upper right	Moving right hand from lower right to upper right	(4/5) Didn't notice the start position; (1/5) assumed the starting pose is a stand position

Table 8.1: Incorrect movements performed by participants in Study 1.

Study 2: Demonstration Interface

We hypothesized that amateurs can efficiently create motion illustrations using DemoDraw's multi-modal Demonstration Interface. Immediately after Study 1, the same 10 participants completed this study (total time was 45 to 60 minutes).

Study 2 began with training where participants were introduced to DemoDraw and shown how to create the second set of illustrations from Study 1. Participants then recorded the same motions and reviewed the results (training was 5 to 10 minutes). Then, four tasks were completed in sequence:

1. The experimenter demonstrated 4 moves for a gestural interface with their right hand: waving, circling, a reversed V shape, and right swipe (see Figure D.2-1). Participants were asked to record these motions and review the captured results. Once satisfied, they rated the generated illustrations.
2. Similar to task 1, but with 8 dance moves (see Figure D.2-2).
3. The experimenter introduced the retake operation and asked participants to choose one step from task 2 to revise. Participants re-performed the motion and reviewed.
4. Participants were asked to perform any 4 to 8 moves they could imagine and retake them until they were satisfied with the results (within a time limit of 5 minutes).

The system was displayed on a 30-inch monitor (with mouse and keyboard) and the Kinect sensor was placed 3-feet above the floor, capturing 8×8-feet of clean space.

Measures. In tasks 1 and 2, participants rated each step along five dimensions: (Q1) “*The visualization accurately captured/described my motion*”, (Q2) “*The visualization shows all the*

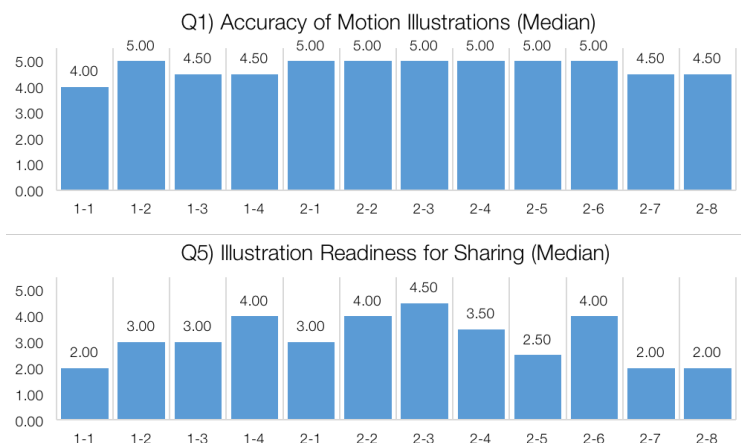


Figure 8.8: Study 2 median ratings for Q1 and Q5 by illustration step.

important joints of movement”, (Q3) “It shows at least one extraneous joint”, (Q4) “The key pose was appropriately chosen”, and (Q5) “This figure needs more (manual) editing before I would share it with others.” The scale for Q1 was a 5-point Likert scale from “1: Strongly disagree” to “5: Strongly agree” and Q5 was from “1: Definitely needs edits” to “5: Very comfortable to share as is”. The answers for Q2 to Q4 were “Yes”, “No”, or “N/A”. Other comments and qualitative feedback were also collected.

Study 2 Results

On average, participants completed task 1 in 5 mins with $\mu = 2.3$ takes, task 2 in 10 mins ($\mu = 2.8$ takes), task 3 in 3 mins ($\mu = 2.3$ takes), and task 4 in 5 mins ($\mu = 2$ takes). Figure D.2 and Figure D.3 provide examples of illustrations created by participants. Below we discuss participant feedback on generated illustrations and system design.

Ratings and Accuracy. Overall, participants thought the illustrations accurately described their motions (Q1 median ratings of 4.5 for task 1 and 4.88 for task 2). However, on average participants rated 4 of the 12 steps as requiring further editing to share with others (Q5 median ratings below 3 for both tasks). Figure 8.8 shows ratings for each step. Participants commented, “This figure represented the overall motion well (...) In particular, it captured all key poses, and the motion lines are easy to follow” (P8), but also “the system picked up really small movements in my other joints that were not relevant to the motion I was trying to depict (such as a small motion in my wrist or elbow)” (P1).

Across all 120 illustrations created all 10 participants, 99% showed all the important joints (Q2), and 80% precisely selected only the salient joints without extraneous movements (Q3). Participants commented: “the picture correctly represents my stance and body position. the arrows are easy to see and follow” (P1), “the lines were very accurate” (P2), and “the arcs are gorgeous and represent the intention of my motion really well” (P5).

Several participants appreciated how DemoDraw smoothed the motion arrows, especially when their demonstration was not perfect or there was capture noise. During the debrief session, participants were shown illustrations from different camera viewpoints and several noted the advantage: *“I love the multiple camera angles for the wiggle arm motion I did in step 1”* (P9).

Key Pose Selection. The answers to Q4 showed 94% of key poses were selected correctly. Example comments include: *“The key poses are very descriptive of the motion”* (P2), and *“The key frames were just right”* (P5).

Authoring Workflow. Participants found DemoDraw easy to learn (Median 5 out of 5) and easy to create illustrations with (Median 4.5). All participants were able to author and navigate using the speech interface. For example, *“The voice command allows people be able to control the system remotely. Without the voice capability, the system can be impractical in the single-person use case”* (P10).

Participants were especially impressed by how fast authoring could generate a step-by-step diagram: *“Surprisingly fast to make some really cool full-body motion demonstrations. There is no way I could do this in higher-fidelity than a napkin sketch in the same time”* (P5) and *“the system saves significant amount of time creating illustrations”* (P10). We also asked participants to estimate the time required if they were to generate a similar 8-step diagram without using DemoDraw, four participants answered that they would not be able to create them manually, while others responded that it would take 90 to 160 minutes based on each single figure taking 10-20 minutes.

Improving a Demonstration. The immediate visual feedback during the capturing phase was effective in helping authors review, adjust, and retake their performances. P4 explained, *“I learned how to exaggerate the important aspects of motion without being explicitly told to.”* In task 3, when we introduced the retaking capability, participants commented that this function would be especially helpful for a long motion sequence. All but one participant chose to retake step 2-7, which involved a holding position with one foot. Three participants later used the same technique for task 4. P8 also noted that it was helpful as *“I could improve this by retaking that step and moving smoothly”* when referring to a specific pose they thought needed additional work.

Study 3: Refinement Interface Effectiveness

To understand how users refine automatically-generated results and generate different visualization styles in Refinement Interface, we conducted an informal study with 4 participants from an IT company (all males, aged 23 to 32 years, $M=28$). The same apparatus as Study 2 was used. First, experimenters guided participants during a 5-minute training phase to load one motion recording and create an illustration by manipulating a set of visual parameters. In the evaluation phase, participants were given two motion recordings, each with three illustrations created using our system. We presented the printed figures one by one and asked them to reproduce them. Participants then used DemoDraw to physically perform and record two specific motions given by the experimenter, and create illustrations to best convey each motion.

Results. Participants actively experimented with visual parameters and styles provided by the Refinement Interface, especially when creating stroboscopic effects. For example: number of intermediate frames and offset, dragging to reposition the arrows, and arrow color and width. In

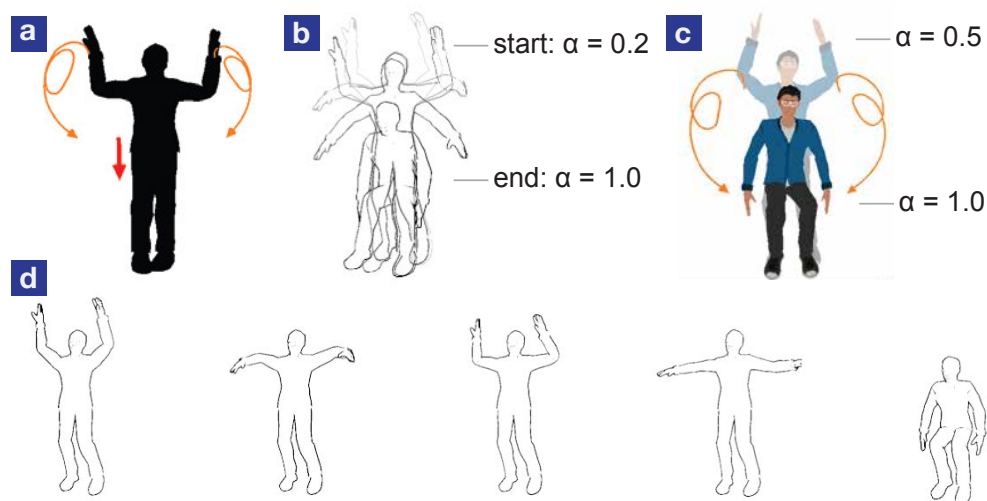


Figure 8.9: Different illustration effects conveying the same motion recording using DemoDraw's Refinement Interface: a and c are created by the authors of this work and a was used in Study 1; b by Study 3-P1 using 4 intermediate frames with zero offset; d by Study 3-P2 using 5 frames, positioned as a sequence.

addition, participants formed strong preferences for styles once given these visualization options. For example, P4 said the stroboscopic effect in task 3 *“is exactly what I looked for – It clearly conveys the start and end poses.”* P2 preferred the cartoon renderer over the silhouette since *“this character looks just like me!”* All said they could not create similar illustrations without DemoDraw (Median 2 out of a 5-point Likert scale). This indicates detailed editing with the Refinement Interface was effective and expressive for various motion types.

Discussion

Participants were clearly excited about the overall experience using both the Demonstration Interface and the Refinement Interface. Some explicitly pointed out their enjoyment: *“it accurately captures how much fun I had making it. :)”* (Study 2-P9), and *“For professional artists, the system not only increases their productivity, but also brings joy and fun to this kind of tasks”* (Study 2-P10). Participant feedback suggests motion illustrations generated by DemoDraw are expressive enough to depict their demonstrations. Our multi-modal interface with motion analysis and rendering algorithms enabled users to quickly create step-by-step diagrams.

Support of Various Styles. In Study 1, motion arrows successfully conveyed the majority of movements. When arrows alone are not adequate, stroboscopic could be combined to clarify details in the start, intermediate, and end poses. For example, the simultaneous hand movements and squatting action of Step 3 in Study 1 are difficult to convey using only arrows (see Table 8.1), but Study 3 participants chose a stroboscopic effect to convey the same motion (see Figure 8.9). These findings align with existing instructional design principles [153], which suggest designers combine

illustration styles based on the context. An authoring interface (like Refinement Interface in our system) must make it easy to adjust styles and parameters within a style (i.e., an outline figure with motion arrows).

Iterative Creation Process. Our studies verified our earlier findings that creating motion illustrations is an iterative procedure, where (re-)performing, reviewing, and refining are necessary components (see Figure 8.3). In Study 2, we observed how retaking a partial demonstration could be useful for long step-by-step motion sequences; Study 3 suggested that once moving into a refinement phase, authors focused on detailed adjustments of captured movements. Ensuring a high-level review of capturing results during demonstrations is therefore important.

Animation vs. Illustration. As DemoDraw captures the continuous motion sequence in 3D from a demonstrator, our system also generates animations showing the dynamic movements. In the warm-up task of Study 2 that captured the second motion set in Study 1, some participants explained that the playback animation of the recording clarified the motion where they incorrectly interpreted the start position. We propose that as motion arrows can efficiently and effectively express most of the motions, a mixed-media version can be created, where viewers can selectively review part of a static diagram with in-place animation playback. Such format has been shown to be useful for clarifying step-by-step instructions [46]. In addition, the 3D reconstruction also makes it possible to review motions from different viewing angles. All in all, our technology enables both instructors and viewers to interactively create and review motion illustrations in multiple ways.

Design Implications. In this work, we focus on body motion diagrams. More broadly, we believe our work also provides findings and techniques that apply to other demonstration-based systems. First, our multi-modal interface enables authors to perform critical tasks without leaving the demonstration context. Second, separating demonstration capture from detailed refinement allows users to focus solely on performance at demonstration-time while preserving the ability to fine-tune results later on. Third, clear, real-time illustration previews of performed motions enable fast iteration at demonstration.

8.7 Conclusion

We introduced DemoDraw, a multi-modal system for generating human motion illustrations by physically demonstrating desired movements. It translates speech and 3D joint motion into a segmented sequence of key poses and salient joint movements, which are used to automatically generate a series of motion illustrations in effective and understood illustration styles. A multi-modal Demonstration Interface enables authors to record, review, and retake physical movements, and later refine and explore different motion visualizations with a Refinement Interface. We believe this “demonstrate-refine” pattern will generalize to other demonstration-based authoring systems. The primary motivation of this work is to provide users with domain-appropriate authoring tools that free them from tedious low-level tasks – allowing them to focus their effort on both communicative and aesthetic aspects. We look forward to applying the same approach to other instructional materials and illustration types in the future.

Limitations and Future Work

Like any system, there are limitations imposed by architectural decisions and limits in available technology.

Limited Interactions in Demonstration Mode. Presently, authors can review and retake steps using voice commands, but many fine-grained operations are only available in the Refinement Interface, which requires users to leave the performance area. Future work should investigate if voice commands combined with gestures can expose more functionality like timeline scrubbing to the author, to tighten the feedback loop between performance, context setting, and depiction.

Motion Capture and Segmentation. First, the quality of DemoDraw illustrations is limited by the accuracy of motion capture data. Second, our segmentation algorithms currently assume that motions are separated by periods of inactivity, so we cannot yet capture and segment continuous motions that might be necessary, e.g., in different sports, where interruptions are not possible. Third, retargeting motion from a human performer to an avatar can introduce artifacts when skeletal geometry does not match. Future work could apply retargeting approaches from the computer graphics literature [83] or examine if it is feasible to automatically generate suitable avatars that match performers' anatomy more closely.

Movements Involving Objects and Multiple Users. Many illustrations focus on motions while holding props (e.g., a tennis racket or baseball bat in sports) or the manipulation of objects (e.g., furniture assembly). We do not yet support such motions as the Kinect depth sensor we employed are limited to track skeletons. While the general case seems very hard, using techniques for recognizing objects in video based on a library of 3D models [117] appears promising. Furthermore, recent work has proposed fine-grained 3D tracking of humans and objects [66] and hands [202]. These techniques may reduce the numbers of retakes to obtain an artifact-free performance observed in our studies. Our current implementation is for single user, but we argue that it is possible to include multiple performers by loading and controlling additional avatar models, which would be especially useful in dancing.

Interpretability of Motions. DemoDraw can visualize the trajectories of multiple joints in a single image, but does not yet take the different timing of sub-motions into account. This can make illustrations of complex motions hard to interpret. Future work could provide per-joint timelines and automatically number sub-motions by their start times. In addition, the dynamics of motion are not adequately represented in output images. To address this, we have begun to experiment with mixed-media output formats. Inspired by MixT [46], we can render static illustrations that can replay a motion segment as an animation when clicked.

Chapter 9

Conclusion

We have presented five interactive authoring systems for creating and reviewing instructions from author demonstration. This final chapter restates the contributions of this dissertation and discusses future directions.

9.1 Restatement of Contributions

This dissertation has demonstrated video-based computational approaches that support tutorial creation and consumption. A set of interactive systems that generate concise instructions from author demonstrations are introduced. Design and technical contributions of this work can be summarized as follows:

- New instructional formats that consider learning factors.
 - Mixed-media tutorials composed of step-by-step static instructions and in-place video clips to demonstrate individual operations.
 - Enhanced video playback that contains dynamic glyphs to provide viewers awareness of upcoming interaction events in the video..
- Authoring workflows for amateur users to create effective instructions by demonstration.
 - Methods and user interfaces for recording, reviewing, and editing an instructional task with the support of software and capturing devices.
 - Multi-modal interfaces using motion and voice commands or touch interaction to author step-by-step instructions while performing physical demonstrations.
- Automatic or semi-automatic approaches to produce high-quality instructions using video and audio analysis that includes users in the loop.
 - Algorithms of analyzing video, audio, and motion data using computer vision and signal processing approaches to segment a demonstration.

- Techniques for combining high-level user annotations with content analysis to generate concise instructions.

9.2 Future Directions

Our tools provided evidence how computer technologies could assist amateur authors in creating instructional content, which in turn enhances the learning experience. Based on these design experiences, we propose directions of future research in computational instruction design.

Beyond a Single User and Linear Instructions

The authoring systems presented in this dissertation mainly focus on supporting a single user in a software domain (i.e., MixT and DemoWiz that capture a software application workflow) or physical tasks (including DemoDraw that records an author's movements). The DemoCut system analyzes one single, static video shot. The example videos we tested included only one demonstrator. We argue that our techniques can possibly apply to content that contains multiple demonstrators' activities and narrations, but our authoring interfaces are designed for a single author. Finally, the Kinectograph recording device supports two demonstrators in a scene, and the tablet interface can be possibly controlled by both users at the same time.

Tool Support for a Team. We see the need of expanding our designs to support a group of authors. In specific domains, a demonstration often includes several demonstrators. For examples, furniture assembly tasks are commonly performed collaboratively by two or more people. Demonstrators may have different roles, such as one as the main instructor, while the other(s) serve as supporters or a learner. For another example, dancers may have different moves in a choreography. Researchers have proposed methods of detecting activities of multiple people in a video [66] and video editing [169], but authoring instructions collaboratively is still an open topic. In addition, professional filmmaking is commonly done by teamwork, where a group of people forms a production team and contributes in different stages, including planning, shooting, and editing [172]. We see research opportunities in supporting large projects or complicated activities that involve long time (e.g., a few days to weeks) and multiple collaborators with the same or different roles (e.g., a director controls a recording device to document a demonstrator's making process). What information should be captured to support live or offline authoring? How should we incorporate authors' roles and intentions into a tool support? What interaction modality would be suitable to a multi-user, multi-task scenario?

Tool Support for Human-Robot Collaboration. Robots have become advanced and accessible to end users for personal tasks, such as cooking [41, 195] and fabricating large-scale structures [211]. Rather than performing a complete task individually, these robots or intelligent tools are designed to collaborate with a human user. Researchers have found that a user, who interacts with an unfamiliar robot, may confront with several challenges: First, it can be difficult to understand how to operate a supportive tool or a robot during a task. Real-time guidance might be needed, especially for physical tasks. Research work has shown haptic or visual feedback to be effective for physical

constructions [3, 225, 186]. Second, as tasks could involve creativity and dynamics that a human and a robot work collaboratively together, it could be challenging to foresee what a robot’s next move is. Conversation [43], eye gaze [9], and motion [67, 199] enable better communication to convey robots’ intent while performing a task. These topics open questions to instructional design: How should instructions be presented, at what timing and in what form, when a user is working with another agent? How can a robot express its intent while a human user is paying attention to a task? We see our DemoWiz design could be one visualization method to present a robot’s upcoming motion path. In fact, a recent Augmented Reality application has demonstrated this idea of visualizing a robot’s trail [52], which helps users avoid collision in a space.

Tool Support for Alternatives. The instructions presented in this dissertation are designed to be navigated linearly in a step-by-step order. An intriguing idea is to enable *non-linear* instructions that can be interactively reviewed by a learner. Interactive narrative, or commonly known as “Choose Your Own Adventure”, is an interactive form of dynamically following a storyline through reader’s actions [180]. By carefully designing a fictional world with multiple branches, readers might experience different stories, including characters, plots, and endings, based on their inputs. It has been shown that such type of navigation could be useful for video editing [189], composing personal stories [49], and following educational videos [119]. As instructions involve domain knowledge and experiences, authors have created similar forms via external links to guide viewers to other instructions [212] (see Figure 9.1). This serves several purposes, including to complete a tutorial (e.g., to pick up basic skills), to demonstrate different approaches, results, or effects, and to raise viewers’ interests. In addition, learners could also contribute to the instructional content via comments or edits, making a production process iterative. To support interactive navigation experiences, how would an authoring tool enable alternatives of performing a task? How would learners efficiently preview options and make a choice between approaches? One useful approach is to visualize different workflows for comparison [126] or incorporate learners’ comments [35], but supporting both authoring and collaborative editing for alternatives is yet an open question.

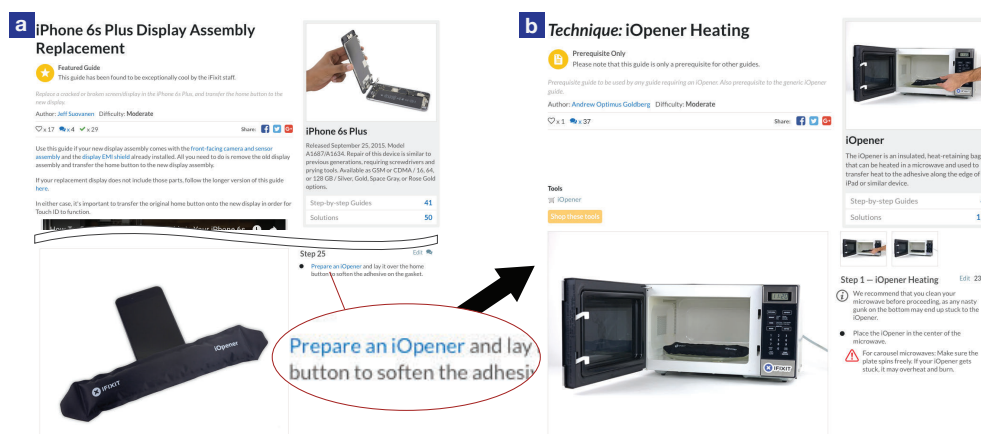


Figure 9.1: Online instructions often include external links (a) to other materials (b), which enhance or expand a step-by-step tutorial. Example by Jeff Suovanen [196], licensed under CC BY 3.0.

Supporting Different Levels of Expertise

Authoring instructions involves expertise from three perspectives: making, teaching, and editing. The first element, *making*, considers how authors master an instructional topic to perform a task. In this dissertation, we support authors who are the domain experts. These professional instructors often have a clear overview of a task and are comfortable with demonstrations. It could be interesting to also support amateurs or novices that are less familiar with a task workflow and want to find alternatives while working on a project. Researchers have proposed interactive methods to enable live authoring personal videos [79] or provide real-time photo taking recommendation from community data [30] during filming. What information would amateur authors find useful when creating a project?

The second element, *teaching*, considers how authors illustrate a task to help learners follow instructions. Our work suggests and automates novel tutorial formats to help authors focus on performing their demonstrations. As instructors have preferred styles of teaching, future authoring tools can include customization with automatic editing effects (e.g., automatically zoom to a region when pointing at an area with verbal or gestural cues) and material reuse (e.g., include the successful footage of other instructions). Learners can also possibly define their preferences, such as automatically slowing down or replaying a video for unfamiliar parts, which have been shown useful for navigating personal videos [45].

The third element, *editing*, considers how authors compose the instructional material into a final form, such as a video or a static tutorial. Our tools focus on automating this aspect for authors who are not professional in editing. We suggest that the results of our work can be used for advanced editing, such as exporting to conventional video or graphics editing tools. By carefully considering different levels of expertise, we believe that creation tools can be flexibly support a wider range of authoring and learning needs.

Machine Learning from Instructions

The interactive systems presented in this dissertation focus on the authoring process for individuals, but one interesting question is what we can learn from the created tutorials. In Chapter 2, we discussed the popularity of online instructions. What are the common techniques or patterns that can be seen from millions of documents? In recent years, machine learning methods have become powerful to analyze and reason big data. Techniques have been applied to various interactive systems, including suggesting lecture videos [119], web designs [127], or everyday activities [73], comparing instructions for image manipulation tasks [168], and identifying dietary or cooking patterns [110, 214]. However, support for authoring instructional content is not yet seen. How do we learn from existing tutorials and user comments to suggest authors adding explanations (e.g., for specific subtasks that people often feel confused), making clarifications (e.g., “is this similar to (another method)?”), or finding references (e.g., to point to a relevant tutorial)? Can we identify instructional topics for authors to brainstorm and contribute to the communities? In addition, Grossman et al. [95]’s design guidelines for GUI systems suggested that it is important to help users understand task flows, increase awareness of features, and locate functionality to improve

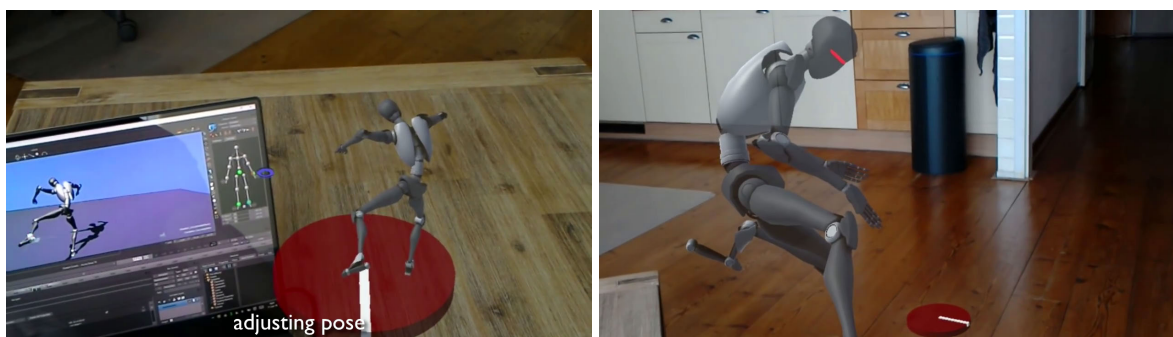


Figure 9.2: A recent Augmented Reality (AR) application enables reviewing character animation beyond a desktop in a room-size environment [32], licensed under CC BY 2.0.

software learnability. As static tutorials describe these key elements with text and images, what usability problems can we automatically identify to suggest developers improving interface design?

Emerging Instructional Space

Augmented and virtual reality systems are becoming available to end users in affordable forms. Commercial devices on the market have provided high-quality displays and motion-based user inputs for VR (e.g., Oculus Rift¹ and HTC Vive²) and AR (e.g., Microsoft HoloLens³). 360-degree or spherical videos showing a complete view of a space can be captured using one omnidirectional camera or a circular array of cameras (e.g., Jump camera rig [89]). In addition, sensing technologies have been greatly improved to live track human actions [66], hands [202], and locations (e.g., Project Tango [90]) in a 3D space. These enable novel applications for providing real-time instructions between people [99, 152] and live editing and testing beyond desktop [32] (see Figure 9.2). However, designing AR and VR experiences requires expertise and skills in computer graphics and 3D modeling. New authoring tools that focus on delivering immersive, in-person experiences are needed. How can a tool capture a real-world demonstration and effectively transfer to another remote learner? What input modality would be suitable for authoring in a physical world? How can a creation tool enable more interactive and iterative design beyond a conventional production pipeline? These future directions in authoring AR and VR content are becoming more important with increasing numbers of amateur developers and authors in recent and the following years.

¹ <https://www.oculus.com/>

² <https://www.htcvive.com/>

³ <https://www.microsoft.com/microsoft-hololens/>

9.3 Summary

In this dissertation, we present video-based interactive approaches designed for amateur users to produce and consume effective instructions from demonstrations. Using video and audio analysis techniques, our tools support recording, editing, and playback in a tutorial production process. We demonstrate results generated from five interactive systems that we designed for several instructional domains, including software applications and physical activities. Our methods increase the quality of instructions created by tutorial authors, which in turn improves learning for viewers who interactively navigate the content.

Appendix A. Materials for the MixT Formative Study

The complete tutorial of Task 2 we provided in the MixT formative study.

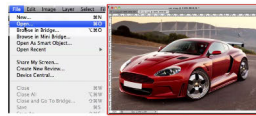
Rapid Movement Effect

How to use Tutorial Builder

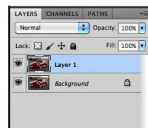
By Adobe CTL; revised by UC Berkeley research team



1 In this tutorial, we will create a rapid movement effect on a photo of a still car. Choose File > Open to open the desired photo in Photoshop.



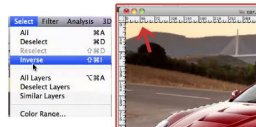
2.1 Duplicate layer with Cmd+J.



2.2 With Quick Select tool, select out the car from the copied layer. You can decrease the brush size (shortcut key: B) for finer details. If you make a mistake, hold down the Option key while brushing to deselect the region.



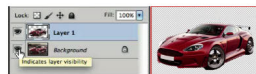
2.3 With the car selected, go to Select > Inverse and press Del. Now the background is removed from this layer.



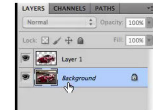
2.4 Choose Edit > Clear to Delete the selected pixels.



2.5 Click on the eye next to the Background layer to hide and see the difference. Click again to show the background.



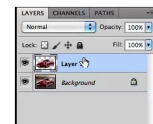
3.1 Select "Background" in the Layers Panel.



3.2 With the background layer selected, go to Filter > Blur > Motion Blur. In the dialog box, change the direction of the blur with the Angle parameter and the strength of the blur with the Distance parameter as desired for your photograph. Make the distance parameter fairly large (e.g. 150); we will add a smaller blur for the car itself, but the background should be more blurred. Click OK to apply the changes.



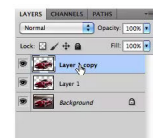
4.1 Select "Layer 1" in the Layers Panel.



4.2 Press Cmd+D to deselect.



4.3 Then duplicate the car layer with Cmd+J.



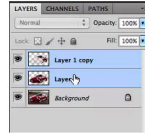
4.4 With the copied layer selected, go to Filter > Blur > Motion Blur. In the dialog box, set the Angle parameter to be the same as you did in step 3.2, but make the Distance parameter much smaller (e.g. 20) than you did in 3.2.



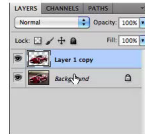
4.5 Select Eraser Tool in the toolbar with soft-edged brush and erase the blurred layer in the hood area.



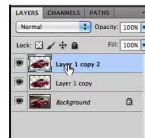
5.1 Shift click the "Layer 1" layer in the Layers Panel to select additional layers.



5.2 Merge layers with car by selecting both layers and pressing Cmd+E



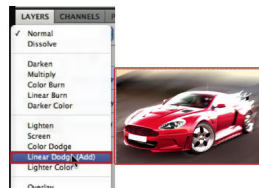
5.3 Duplicate car layer using Cmd+J.



5.4 Go to Filter > Liquify for the copied layer. Use the Liquify tool by clicking and dragging short strokes along the back of the car to create the melting effect as shown in the image. Click okay when you finish.



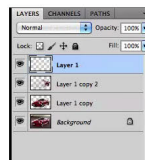
5.5 Change blending layer mode to Linear Dodge (Add).



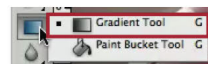
5.6 Select Eraser Tool in the toolbar with soft-edged brush and erase everything from the front side of the car to lessen the brightness.



6.1 Create a new layer on top.



6.2 Go to Gradient Tool in the toolbar



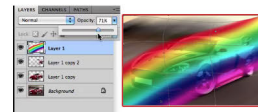
6.3 select Transparent Rainbow (standard Photoshop gradient) and make sure option Linear Gradient is selected.



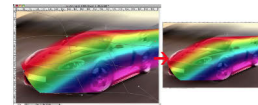
6.4 Fill new layer with this gradient by clicking and dragging the mouse to form a perpendicular line to the desired strip of the rainbow.



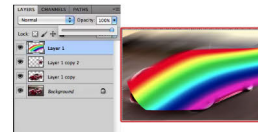
6.5 Go to Edit > Transform > Warp with the rainbow layer selected. Temporarily lower the opacity of this layer so you can see the car underneath.



6.6 Click and drag on the grid to warp the rainbow to fit the front of the car and extend to the edge of the photograph.



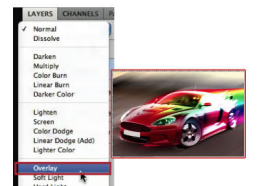
6.7 Change the opacity back to 100%.



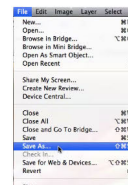
6.8 Select Eraser Tool in the toolbar with soft-edged brush one more time and erase the gradient from the front side of the car.



6.9 Change blending layer mode to Overlay.



7 Choose File > Save As to Save the current document with different options.



Appendix B. Materials for the DemoCut Formative Study

20 YouTube How-To videos we analyzed in the DemoCut formative user study.

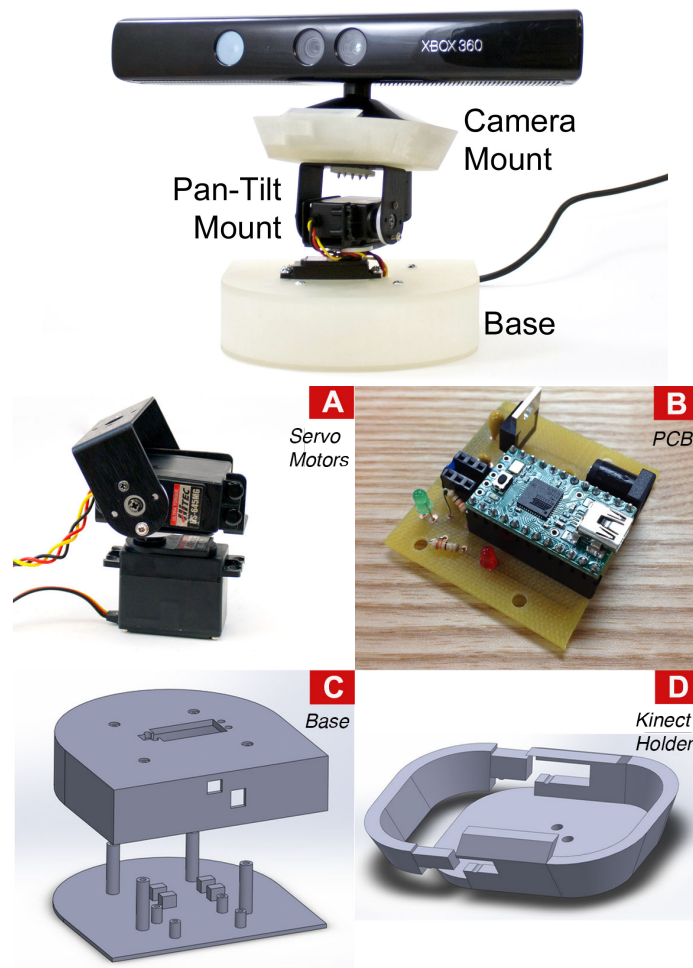
A video playlist is available online at

<https://www.youtube.com/playlist?list=PLAq2QZEiIgn80wbHOp9In4s8IzDnkb3II>

Category	Index	Length	Views	Creation Date	Link
Electronics	1	1:54:00	3,006	Oct 21, 2012	https://youtu.be/watch?v=zOBy6iKjpso
	2	5:32:00	22,211	Sep 16, 2012	https://youtu.be/watch?v=4cuneFm-AG4
	3	6:00:00	71,895	Jun 18, 2012	https://youtu.be/watch?v=hOdu1Zl1lic
	4	5:34:00	130,001	Jun 30, 2011	https://youtu.be/watch?v=zYjOqcbBEco
Home/Repair	5	5:59:00	2,231	Oct 6, 2012	https://youtu.be/watch?v=Y5j55Mlg09s
	6	8:40:00	5,063	Jan 4, 2013	https://youtu.be/watch?v=54k_OgAT_uQ
	7	3:42:00	8,312	Mar 15, 2011	https://youtu.be/watch?v=JL5Q2lJAAdk
	8	7:28:00	32,201	Mar 10, 2012	https://youtu.be/watch?v=7OrXrmFqXv0
Art	9	5:50:00	699,177	April 12, 2012	https://youtu.be/watch?v=RFauBI0zTvw
	10	4:53:00	4,004,613	Nov 18, 2011	https://youtu.be/watch?v=iybQwiJWToM
	11	9:08:00	58,248	Aug 27, 2012	https://youtu.be/watch?v=va0sOYEHRho
	12	2:26:00	6,203	Feb 8, 2013	https://youtu.be/watch?v=NHfPywNg_Vw
Craft	13	6:25:00	12,8741	Jan 12, 2013	https://youtu.be/watch?v=RajOsFVWJ4I
	14	3:09:00	19,303	Jan 22, 2013	https://youtu.be/watch?v=8KP-trX9I5g
	15	4:17:00	1,156	Oct 5, 2012	https://youtu.be/watch?v=ajW6h6JAR04
	16	2:32:00	2,804	Feb 2, 2013	https://youtu.be/watch?v=qBUJtRNUGOA
Food	17	2:16:00	130,752	Jun 4, 2009	https://youtu.be/watch?v=3opfBL9YZ10
	18	8:14:00	30,740	Apr 11, 2011	https://youtu.be/watch?v=DWxUa9LwbSY
	19	3:47:00	22,670	Sep 12, 2011	https://youtu.be/watch?v=l0QHDHMe9oU
	20	3:54:00	9,192	Feb 6, 2012	https://youtu.be/watch?v=5G3OGIN7Cx8

Appendix C. The Initial Design of Kinectograph

The initial design of Kinectograph and its hardware components that we used in Study 1 and 2. A base (c) stabilizes the system and houses the electronics, such as a custom PCB board (b). The bottom servo (a) of the pan-tilt system fastens to the middle of the base, while the Kinect holder (d) attaches the Kinect sensor to the pan-tilt system. Both (c) and (d) are fabricated on a Projet HD 3000 printer.



Our newer system used Kubi⁴ to replace parts a-c for fine-grained pan-tilt control (see Section 7.3).

⁴ <http://www.revolverobotics.com/>

Appendix D. Materials for the DemoDraw User Study and Results

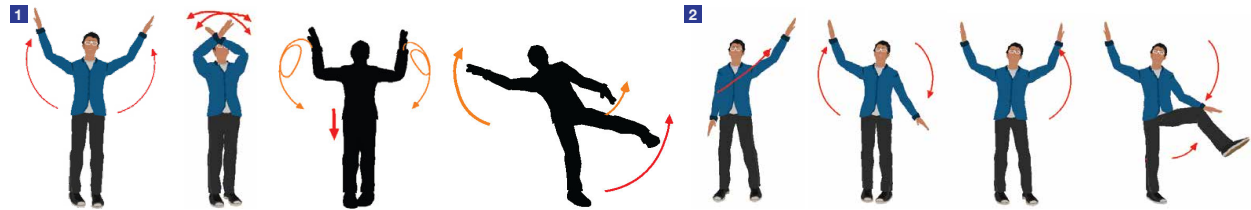


Figure D.1: Tasks provided in Study 1: We showed the printouts of these two sets of 4-step motions generated by DemoDraw using both the Demonstration Interface and the Refinement Interface. We asked participants to re-perform in front of a camera.

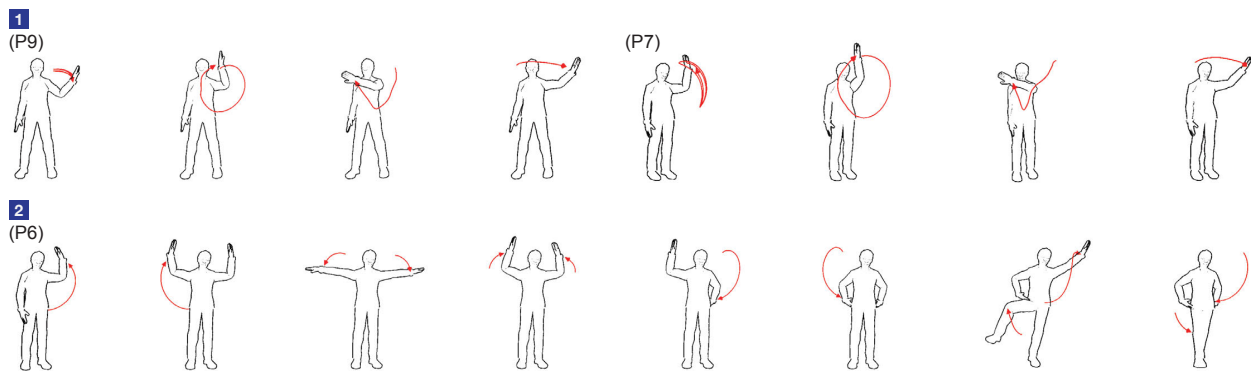


Figure D.2: Step-by-step illustrations generated by participants in Study 2 using the Demonstration Interface: 1) Results from P9 and P7 show the same four gestures of interface control in task 1, and 2) Results from P6 show 8-step moves in task 2.

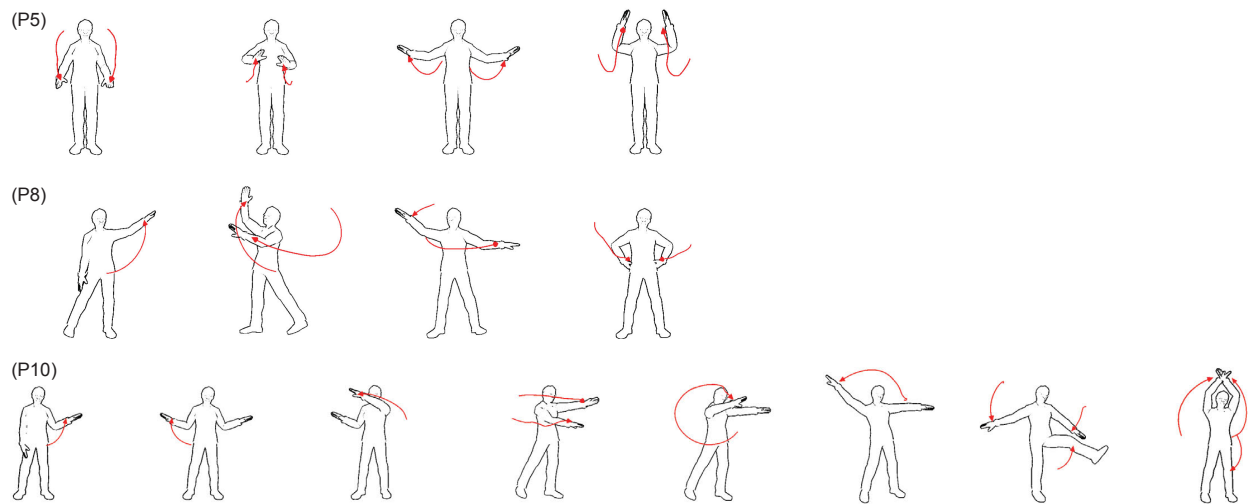


Figure D.3: Selected illustrations from the open-ended task created by three different participants using the Demonstration Interface in Study 2: P5 performed to conduct a 4/4 beat pattern; P8 and P10 each performed four and eight free moves.

Bibliography

- [1] Brett Adams and Svetha Venkatesh. “Situating event bootstrapping and capture guidance for automated home movie authoring”. In: *Proceedings of MULTIMEDIA*. ACM Press, 2005, pp. 754–763.
- [2] J.K. Aggarwal and M.S. Ryoo. “Human Activity Analysis: A Review”. In: *ACM Comput. Surv.* 43.3 (Apr. 2011), 16:1–16:43. ISSN: 0360-0300. DOI: 10.1145/1922649.1922653. URL: <http://doi.acm.org/10.1145/1922649.1922653>.
- [3] Harshit Agrawal, Udayan Umapathi, Robert Kovacs, Johannes Frohnhofer, Hsiang-Ting Chen, Stefanie Mueller, and Patrick Baudisch. “Protopiper: Physically Sketching Room-Sized Objects at Actual Scale”. In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST ’15. Daegu, Kyungpook, Republic of Korea: ACM, 2015, pp. 427–436. ISBN: 978-1-4503-3779-3. DOI: 10.1145/2807442.2807505. URL: <http://doi.acm.org/10.1145/2807442.2807505>.
- [4] Maneesh Agrawala, Wilmot Li, and Floraine Berthouzoz. “Design Principles for Visual Communication”. In: *Commun. ACM* 54.4 (Apr. 2011), pp. 60–69. ISSN: 0001-0782. DOI: 10.1145/1924421.1924439. URL: <http://doi.acm.org/10.1145/1924421.1924439>.
- [5] Maneesh Agrawala, Doantam Phan, Julie Heiser, John Haymaker, Jeff Klingner, Pat Hanrahan, and Barbara Tversky. “Designing effective step-by-step assembly instructions”. In: *ACM Transactions on Graphics (TOG)*. Vol. 22. 3. ACM. 2003, pp. 828–837.
- [6] Fraser Anderson, Tovi Grossman, Justin Matejka, and George Fitzmaurice. “YouMove: Enhancing Movement Training with an Augmented Reality Mirror”. In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST ’13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 311–320. ISBN: 978-1-4503-2268-3. DOI: 10.1145/2501988.2502045. URL: <http://doi.acm.org/10.1145/2501988.2502045>.
- [7] Fraser Anderson, Tovi Grossman, Justin Matejka, and George Fitzmaurice. “YouMove: enhancing movement training with an augmented reality mirror”. In: *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM. 2013, pp. 311–320.
- [8] Robert Anderson and Jean Anderson. “Before and After Weight Training”. In: *Stretching*. Bolinas, California: Shelter Publications, 2010, pp. 210–211. ISBN: 978-0936070469.

- [9] Sean Andrist, Xiang Zhi Tan, Michael Gleicher, and Bilge Mutlu. “Conversational Gaze Aversion for Humanlike Robots”. In: *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*. HRI ’14. Bielefeld, Germany: ACM, 2014, pp. 25–32. ISBN: 978-1-4503-2658-2. DOI: 10.1145/2559636.2559666. URL: <http://doi.acm.org/10.1145/2559636.2559666>.
- [10] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. “Action synopsis: pose selection and illustration”. In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 667–676.
- [11] Jackie Assa, Daniel Cohen-Or, I-Cheng Yeh, Tong-Yee Lee, et al. “Motion overview of human actions”. In: *ACM Transactions on Graphics (TOG)*. Vol. 27. 5. ACM. 2008, p. 115.
- [12] Ronald M. Baecker. “Picture-driven animation”. In: *Proceedings of the May 14-16, 1969, spring joint computer conference*. Boston, Massachusetts: ACM, 1969, pp. 273–288. DOI: 10.1145/1476793.1476838. URL: <http://portal.acm.org/citation.cfm?id=1476838> (visited on 11/02/2009).
- [13] Jiamin Bai, Aseem Agarwala, Maneesh Agrawala, and Ravi Ramamoorthi. “Selectively de-animating video”. In: *ACM Trans. Graph.* 31.4 (2012), 66:1–66:10. ISSN: 0730-0301. DOI: 10.1145/2185520.2185562. URL: <http://doi.acm.org/10.1145/2185520.2185562>.
- [14] Nikola Banovic, Tovi Grossman, Justin Matejka, and George Fitzmaurice. “Waken: reverse engineering usage information and interface structure from software videos”. In: *UIST ’12*. ACM Press, 2012, p. 83.
- [15] Connelly Barnes, Dan B. Goldman, Eli Shechtman, and Adam Finkelstein. “Video tapestries with continuous temporal zoom”. In: *ACM Trans. Graph.* 29 (4 2010), 89:1–89:9. ISSN: 0730-0301. DOI: <http://doi.acm.org/10.1145/1778765.1778826>. URL: <http://doi.acm.org/10.1145/1778765.1778826>.
- [16] Connelly Barnes, David E. Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. “Video Puppetry: A Performative Interface for Cutout Animation”. In: *ACM SIGGRAPH Asia 2008 Papers*. SIGGRAPH Asia ’08. New York, NY, USA: ACM, 2008, 124:1–124:9. ISBN: 978-1-4503-1831-0. DOI: 10.1145/1457515.1409077. URL: <http://doi.acm.org/10.1145/1457515.1409077> (visited on 02/23/2015).
- [17] Barbara Barry. “The Mindful Camera: Common Sense for Documentary Videography”. In: *Proceedings of the Eleventh ACM International Conference on Multimedia*. MULTIMEDIA ’03. Berkeley, CA, USA: ACM, 2003, pp. 648–649. ISBN: 1-58113-722-2. DOI: 10.1145/957013.957152. URL: <http://doi.acm.org/10.1145/957013.957152>.
- [18] Tom Bartindale, Guy Schofield, Clara Crivellaro, and Peter Wright. “TryFilm: Situated Support for Interactive Media Productions”. In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. CSCW ’16. San Francisco, California, USA: ACM, 2016, pp. 1412–1422. ISBN: 978-1-4503-3592-8. DOI: 10.1145/2818048.2819929. URL: <http://doi.acm.org/10.1145/2818048.2819929>.

- [19] Tom Bartindale, Alia Sheikh, Nick Taylor, Peter Wright, and Patrick Olivier. “StoryCrate: Tabletop Storyboarding for Live Film Production”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’12. Austin, Texas, USA: ACM, 2012, pp. 169–178. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2207700. URL: <http://doi.acm.org/10.1145/2207676.2207700>.
- [20] Patrick Baudisch, Nathaniel Good, Victoria Bellotti, and Pamela Schraedley. “Keeping Things in Context: A Comparative Evaluation of Focus Plus Context Screens, Overviews, and Zooming”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’02. Minneapolis, Minnesota, USA: ACM, 2002, pp. 259–266. ISBN: 1-58113-453-3. DOI: 10.1145/503376.503423. URL: <http://doi.acm.org/10.1145/503376.503423>.
- [21] Patrick Baudisch, Desney Tan, Maxime Collomb, Dan Robbins, Ken Hinckley, Maneesh Agrawala, Shengdong Zhao, and Gonzalo Ramos. “Phosphor: Explaining Transitions in the User Interface Using Afterglow Effects”. In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*. UIST ’06. Montreux, Switzerland: ACM, 2006, pp. 169–178. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166280. URL: <http://doi.acm.org/10.1145/1166253.1166280>.
- [22] Benjamin B. Bederson and James D. Hollan. “Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics”. In: *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*. UIST ’94. Marina del Rey, California, USA: ACM, 1994, pp. 17–26. ISBN: 0-89791-657-3. DOI: 10.1145/192426.192435. URL: <http://doi.acm.org/10.1145/192426.192435>.
- [23] Bernard Forest de Bélidor. *L’architecture hydraulique*. v. 1. chez C. A. Jombert, 1737. URL: <https://books.google.com/books?id=I7946H-XdjUC>.
- [24] Lawrence Bergman, Vittorio Castelli, Tessa Lau, and Daniel Oblinger. “DocWizards: A System for Authoring Follow-me Documentation Wizards”. In: *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*. UIST ’05. Seattle, WA, USA: ACM, 2005, pp. 191–200. ISBN: 1-59593-271-2. DOI: 10.1145/1095034.1095067. URL: <http://doi.acm.org/10.1145/1095034.1095067>.
- [25] Lawrence Bergman, Jie Lu, Ravi Konuru, Julie MacNaught, and Danny Yeh. “Outline Wizard: Presentation Composition and Search”. In: *Proceedings of the 15th International Conference on Intelligent User Interfaces*. IUI ’10. Hong Kong, China: ACM, 2010, pp. 209–218. ISBN: 978-1-60558-515-4. DOI: 10.1145/1719970.1719999. URL: <http://doi.acm.org/10.1145/1719970.1719999>.
- [26] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. “Crowds in Two Seconds: Enabling Realtime Crowd-powered Interfaces”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, California, USA: ACM, 2011, pp. 33–42. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047201. URL: <http://doi.acm.org/10.1145/2047196.2047201>.

- [27] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. “Tools for placing cuts and transitions in interview video”. In: *ACM Trans. Graph.* 31.4 (2012), 67:1–67:8. ISSN: 0730-0301. DOI: 10.1145/2185520.2185563. URL: <http://doi.acm.org/10.1145/2185520.2185563>.
- [28] John B. Black, John M. Carroll, and Stuart M. McGuigan. “What Kind of Minimal Instruction Manual is the Most Effective”. In: *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface*. CHI ’87. Toronto, Ontario, Canada: ACM, 1987, pp. 159–162. ISBN: 0-89791-213-6. DOI: 10.1145/29933.275623. URL: <http://doi.acm.org/10.1145/29933.275623>.
- [29] Richard A. Bolt. “Put-that-there: Voice and Gesture at the Graphics Interface”. In: *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’80. New York, NY, USA: ACM, 1980, pp. 262–270. ISBN: 0-89791-021-4. DOI: 10.1145/800250.807503. URL: <http://doi.acm.org/10.1145/800250.807503> (visited on 09/17/2014).
- [30] Steven Bourke, Kevin McCarthy, and Barry Smyth. “The Social Camera: A Case-study in Contextual Image Recommendation”. In: *Proceedings of the 16th International Conference on Intelligent User Interfaces*. IUI ’11. Palo Alto, CA, USA: ACM, 2011, pp. 13–22. ISBN: 978-1-4503-0419-1. DOI: 10.1145/1943403.1943408. URL: <http://doi.acm.org/10.1145/1943403.1943408>.
- [31] Simon Bouvier-Zappa, Victor Ostromoukhov, and Pierre Poulin. “Motion cues for illustration of skeletal motion capture data”. In: *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*. ACM. 2007, pp. 133–140.
- [32] Jasper Brekelmans. *Microsoft HoloLens with Autodesk MotionBuilder*. 2016. URL: <https://www.youtube.com/watch?v=yfl7pwXftUs> (visited on 07/10/2016).
- [33] Aaron Bryden, George Phillips Jr., and Michael Gleicher. “Automated Illustration of Molecular Flexibility”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.1 (Jan. 2012), pp. 132–145. ISSN: 1077-2626. DOI: 10.1109/TVCG.2010.250. URL: <http://dx.doi.org/10.1109/TVCG.2010.250>.
- [34] Mountain Buggy. *Nano Travel Stroller Specs and Instructions — Mountain Buggy*. 2016. URL: <https://mountainbuggy.com/us/Products/buggies/nano/Specifications> (visited on 06/30/2016).
- [35] Andrea Bunt, Patrick Dubois, Ben Lafreniere, Michael A. Terry, and David T. Cormack. “TaggedComments: Promoting and Integrating User Comments in Online Application Tutorials”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 4037–4046. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557118. URL: <http://doi.acm.org/10.1145/2556288.2557118>.
- [36] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN: 0123740371, 9780123740373.

- [37] Scott Carter, John Adcock, John Doherty, and Stacy Branham. “NudgeCam: toward targeted, higher quality media capture”. In: *Proceedings of MULTIMEDIA*. ACM Press, 2010, pp. 615–618.
- [38] Scott Carter, Pernilla Qvarfordt, Matthew Cooper, and Ville Mäkelä. “Creating Tutorials with Web-Based Authoring and Heads-Up Capture”. In: *IEEE Pervasive Computing* 14.3 (July 2015), pp. 44–52. ISSN: 1536-1268. DOI: 10.1109/MPRV.2015.59.
- [39] 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*. DIS ’02. London, England: ACM, 2002, pp. 157–166. ISBN: 1-58113-515-7. DOI: 10.1145/778712.778737. URL: <http://doi.acm.org/10.1145/778712.778737>.
- [40] Jessica R. Cauchard, Jane L. E, Kevin Y. Zhai, and James A. Landay. “Drone & Me: An Exploration into Natural Human-drone Interaction”. In: *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. UbiComp ’15. Osaka, Japan: ACM, 2015, pp. 361–365. ISBN: 978-1-4503-3574-4. DOI: 10.1145/2750858.2805823. URL: <http://doi.acm.org/10.1145/2750858.2805823>.
- [41] Elizabeth Cha, Jodi Forlizzi, and Siddhartha S. Srinivasa. “Robots in the Home: Qualitative and Quantitative Insights into Kitchen Organization”. In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’15. Portland, Oregon, USA: ACM, 2015, pp. 319–326. ISBN: 978-1-4503-2883-8. DOI: 10.1145/2696454.2696465. URL: <http://doi.acm.org/10.1145/2696454.2696465>.
- [42] Tsung-Hsiang Chang and Yang Li. “Deep Shot: A Framework for Migrating Tasks Across Devices Using Mobile Phone Cameras”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: ACM, 2011, pp. 2163–2172. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979257. URL: <http://doi.acm.org/10.1145/1978942.1979257>.
- [43] Crystal Chao, Jinhan Lee, Momotaz Begum, and Andrea L Thomaz. “Simon plays Simon says: The timing of turn-taking in an imitation game”. In: *2011 RO-MAN*. July 2011, pp. 235–240. DOI: 10.1109/ROMAN.2011.6005239.
- [44] Derrick Cheng, Pei-Yu Chi, Taeil Kwak, Björn Hartmann, and Paul Wright. “Body-tracking Camera Control for Demonstration Videos”. In: *CHI ’13 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’13. Paris, France: ACM, 2013, pp. 1185–1190. ISBN: 978-1-4503-1952-2. DOI: 10.1145/2468356.2468568. URL: <http://doi.acm.org/10.1145/2468356.2468568>.
- [45] Kai-Yin Cheng, Sheng-Jie Luo, Bing-Yu Chen, and Hao-Hua Chu. “SmartPlayer: User-centric Video Fast-forwarding”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’09. Boston, MA, USA: ACM, 2009, pp. 789–798.

- ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518823. URL: <http://doi.acm.org/10.1145/1518701.1518823>.
- [46] 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.
- [47] Pei-Yu Chi, Mira Dontcheva, Wilmot Li, Danile Vogel, and Björn Hartmann. “Authoring Illustrations of Human Movements by Iterative Physical Demonstration”. In: *Proceedings of the 29th Annual ACM Symposium on User Interface Software and Technology*. UIST ’16. New York, NY, USA: ACM, 2016.
- [48] Pei-Yu Chi, Bongshin Lee, and Steven M. Drucker. “DemoWiz: Re-performing Software Demonstrations for a Live Presentation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 1581–1590. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557254. URL: <http://doi.acm.org/10.1145/2556288.2557254>.
- [49] Pei-Yu Chi and Henry Lieberman. “Intelligent Assistance for Conversational Storytelling Using Story Patterns”. In: *Proceedings of the 16th International Conference on Intelligent User Interfaces*. IUI ’11. Palo Alto, CA, USA: ACM, 2011, pp. 217–226. ISBN: 978-1-4503-0419-1. DOI: 10.1145/1943403.1943438. URL: <http://doi.acm.org/10.1145/1943403.1943438>.
- [50] 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.
- [51] Myung Geol Choi, Kyungyong Yang, Takeo Igarashi, Jun Mitani, and Jehee Lee. “Retrieval and visualization of human motion data via stick figures”. In: *Computer Graphics Forum*. Vol. 31. 7. Wiley Online Library. 2012, pp. 2057–2065.
- [52] CNET. *CNET News - Using HoloLens, Microsoft overlays physical robot with holographic robot*. 2015. URL: <https://youtu.be/xnrHFV34PfM> (visited on 06/30/2016).
- [53] Peter Cohan. *Great demo! : how to create and execute stunning software demonstrations*. New York: iUniverse, Inc, 2005. ISBN: 978-0595345595.
- [54] Gabe Cohn, Daniel Morris, Shwetak Patel, and Desney Tan. “Humantenna: using the body as an antenna for real-time whole-body interaction”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2012, pp. 1901–1910.
- [55] WikiHow Community. *How to Write a New Article on wikiHow*. 2016. URL: <http://www.wikihow.com/Write-a-New-Article-on-wikiHow> (visited on 06/15/2016).
- [56] Jonathan Corum. *Drawing Science in Sign*. 2012. URL: <http://style.org/sign/>.

- [57] James E. Cutting. “Representing motion in a static image: constraints and parallels in art, science, and popular culture.” In: *Perception* 31.10 (2002), pp. 1165–93. ISSN: 0301-0066. DOI: 10.1068/p3318.
- [58] Marc Davis, Jeffrey Heer, and Ana Ramirez. “Active capture: automatic direction for automatic movies”. In: *Proceedings of MULTIMEDIA*. ACM Press, 2003, p. 88.
- [59] Jonathan D. Denning, William B. Kerr, and Fabio Pellacini. “MeshFlow: Interactive Visualization of Mesh Construction Sequences”. In: *ACM Trans. Graph.* 30.4 (July 2011), 66:1–66:8. ISSN: 0730-0301. DOI: 10.1145/2010324.1964961. URL: <http://doi.acm.org/10.1145/2010324.1964961>.
- [60] Laurent Denoue, Scott Carter, Matthew Cooper, and John Adcock. “Real-time Direct Manipulation of Screen-based Videos”. In: *Proceedings of the Companion Publication of the 2013 International Conference on Intelligent User Interfaces Companion*. IUI ’13 Companion. Santa Monica, California, USA: ACM, 2013, pp. 43–44. ISBN: 978-1-4503-1966-9. DOI: 10.1145/2451176.2451190. URL: <http://doi.acm.org/10.1145/2451176.2451190>.
- [61] Nicholas Diakopoulos and Irfan Essa. “Videotater: An Approach for Pen-based Digital Video Segmentation and Tagging”. In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*. UIST ’06. Montreux, Switzerland: ACM, 2006, pp. 221–224. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166287. URL: <http://doi.acm.org/10.1145/1166253.1166287>.
- [62] Morgan Dixon and James Fogarty. “Prefab: Implementing Advanced Behaviors Using Pixel-based Reverse Engineering of Interface Structure”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’10. Atlanta, Georgia, USA: ACM, 2010, pp. 1525–1534. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753554. URL: <http://doi.acm.org/10.1145/1753326.1753554>.
- [63] Morgan Dixon, Daniel Leventhal, and James Fogarty. “Content and Hierarchy in Pixel-based Methods for Reverse Engineering Interface Structure”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: ACM, 2011, pp. 969–978. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979086. URL: <http://doi.acm.org/10.1145/1978942.1979086>.
- [64] Mira Dontcheva, Robert R. Morris, Joel R. Brandt, and Elizabeth M. Gerber. “Combining Crowdsourcing and Learning to Improve Engagement and Performance”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 3379–3388. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557217. URL: <http://doi.acm.org/10.1145/2556288.2557217>.
- [65] Mira Dontcheva, Gary Yngve, and Zoran Popović. “Layered Acting for Character Animation”. In: *ACM SIGGRAPH 2003 Papers*. SIGGRAPH ’03. San Diego, California: ACM, 2003, pp. 409–416. ISBN: 1-58113-709-5. DOI: 10.1145/1201775.882285. URL: <http://doi.acm.org/10.1145/1201775.882285>.

- [66] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. “Fusion4D: Real-time Performance Capture of Challenging Scenes”. In: *ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 2016.
- [67] Anca D. Dragan, Shira Bauman, Jodi Forlizzi, and Siddhartha S. Srinivasa. “Effects of Robot Motion on Human-Robot Collaboration”. In: *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’15. Portland, Oregon, USA: ACM, 2015, pp. 51–58. ISBN: 978-1-4503-2883-8. DOI: 10.1145/2696454.2696473. URL: <http://doi.acm.org/10.1145/2696454.2696473>.
- [68] Pierre Dragicevic, Gonzalo Ramos, Jacobo Bibliowicz, Derek Nowrouzezahrai, Ravin Balakrishnan, and Karan Singh. “Video Browsing by Direct Manipulation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’08. Florence, Italy: ACM, 2008, pp. 237–246. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357096. URL: <http://doi.acm.org/10.1145/1357054.1357096>.
- [69] 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*. UIST ’06. Montreux, Switzerland: ACM, 2006, pp. 47–56. ISBN: 1-59593-313-1. DOI: 10.1145/1166253.1166263. URL: <http://doi.acm.org/10.1145/1166253.1166263>.
- [70] Darren Edge, Joan Savage, and Koji Yatani. “HyperSlides: Dynamic Presentation Prototyping”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: ACM, 2013, pp. 671–680. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2470749. URL: <http://doi.acm.org/10.1145/2470654.2470749>.
- [71] Southern Illinois University Edwardsville. *Instructional Video Script*. 2007. URL: http://www.siu.edu/MASSCOMM/PDFs/2col_Instruction_vid.pdf (visited on 06/20/2016).
- [72] Michael Ekstrand, Wei Li, Tovi Grossman, Justin Matejka, and George Fitzmaurice. “Searching for Software Learning Resources Using Application Context”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, California, USA: ACM, 2011, pp. 195–204. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047220. URL: <http://doi.acm.org/10.1145/2047196.2047220>.
- [73] Ethan Fast, William McGrath, Pranav Rajpurkar, and Michael S. Bernstein. “Augur: Mining Human Behaviors from Fiction to Power Interactive Systems”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. Santa Clara, California, USA: ACM, 2016, pp. 237–247. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858528. URL: <http://doi.acm.org/10.1145/2858036.2858528>.
- [74] Steven Feiner. “Apex: An Experiment in the Automated Creation of Pictorial Explanations”. In: *IEEE Comput. Graph. Appl.* 5.11 (Nov. 1985), pp. 29–37. ISSN: 0272-1716. DOI: 10.1109/MCG.1985.276329. URL: <http://dx.doi.org/10.1109/MCG.1985.276329>.

- [75] Steven Feiner, Blair MacIntyre, Tobias Höllerer, and Anthony Webster. “A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment”. In: *Personal Technologies* 1.4 (1997), pp. 208–217. ISSN: 1617-4917. DOI: 10.1007/BF01682023. URL: <http://dx.doi.org/10.1007/BF01682023>.
- [76] Jennifer Fernquist, Tovi Grossman, and George Fitzmaurice. “Sketch-sketch Revolution: An Engaging Tutorial System for Guided Sketching and Application Learning”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, California, USA: ACM, 2011, pp. 373–382. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047245. URL: <http://doi.acm.org/10.1145/2047196.2047245>.
- [77] Ezra Fishman. *How Long Should Your Next Video Be?* 2016. URL: <https://wistia.com/blog/optimal-video-length> (visited on 07/30/2016).
- [78] Food.com. *About Food*. 2016. URL: <http://www.food.com/how-to/about-us-31> (visited on 07/15/2016).
- [79] Dustin E.R. Freeman, Stephanie Santosa, Fanny Chevalier, Ravin Balakrishnan, and Karan Singh. “LACES: Live Authoring Through Compositing and Editing of Streaming Video”. In: *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 1207–1216. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557304. URL: <http://doi.acm.org/10.1145/2556288.2557304>.
- [80] G. W. Furnas. “Generalized Fisheye Views”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’86. Boston, Massachusetts, USA: ACM, 1986, pp. 16–23. ISBN: 0-89791-180-6. DOI: 10.1145/22627.22342. URL: <http://doi.acm.org/10.1145/22627.22342>.
- [81] Susan R. Fussell, Leslie D. Setlock, and Robert E. Kraut. “Effects of Head-mounted and Scene-oriented Video Systems on Remote Collaboration on Physical Tasks”. In: CHI ’03 (2003), pp. 513–520. DOI: 10.1145/642611.642701. URL: <http://doi.acm.org/10.1145/642611.642701>.
- [82] Michelle Gantt and Bonnie A. Nardi. “Gardeners and Gurus: Patterns of Cooperation Among CAD Users”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’92. Monterey, California, USA: ACM, 1992, pp. 107–117. ISBN: 0-89791-513-5. DOI: 10.1145/142750.142767. URL: <http://doi.acm.org/10.1145/142750.142767>.
- [83] Michael Gleicher. “Retargetting motion to new characters”. In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM. 1998, pp. 33–42.
- [84] Dan B Goldman, Brian Curless, David Salesin, and Steven M Seitz. “Schematic storyboarding for video visualization and editing”. In: *ACM Transactions on Graphics (TOG)*. Vol. 25. 3. ACM. 2006, pp. 862–871.

- [85] Dan B. Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M. Seitz. “Video Object Annotation, Navigation, and Composition”. In: *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. UIST ’08. Monterey, CA, USA: ACM, 2008, pp. 3–12. ISBN: 978-1-59593-975-3. DOI: 10.1145/1449715.1449719. URL: <http://doi.acm.org/10.1145/1449715.1449719>.
- [86] Daniel R. Goldman. “A Framework for Video Annotation, Visualization, and Interaction”. AAI3275872. PhD thesis. Seattle, WA, USA, 2007. ISBN: 978-0-549-16099-1.
- [87] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. “A non-photorealistic lighting model for automatic technical illustration”. In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM. 1998, pp. 447–452.
- [88] Lance Good and Benjamin B. Bederson. “Zoomable User Interfaces As a Medium for Slide Show Presentations”. In: *Information Visualization* 1.1 (Mar. 2002), pp. 35–49. ISSN: 1473-8716. DOI: 10.1057/palgrave/ivs/9500004. URL: <http://dx.doi.org/10.1057/palgrave/ivs/9500004>.
- [89] Google. *Jump - Google VR*. 2016. URL: <https://vr.google.com/jump/> (visited on 06/30/2016).
- [90] Google. *Project Tango*. 2016. URL: <https://developers.google.com/tango/> (visited on 06/30/2016).
- [91] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. “Generating photo manipulation tutorials by demonstration”. In: *SIGGRAPH ’09* (July 2009).
- [92] Saul Greenberg, Sheelagh Carpendale, Nicolai Marquardt, and Bill Buxton. *Sketching User Experiences: The Workbook*. Morgan Kaufmann. Elsevier/Morgan Kaufmann, 2012. ISBN: 9780123819598. URL: <https://books.google.com/books?id=c-RAUXk3gbkC>.
- [93] Tovi Grossman and Ravin Balakrishnan. “The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor’s Activation Area”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’05. Portland, Oregon, USA: ACM, 2005, pp. 281–290. ISBN: 1-58113-998-5. DOI: 10.1145/1054972.1055012. URL: <http://doi.acm.org/10.1145/1054972.1055012>.
- [94] Tovi Grossman and George Fitzmaurice. “ToolClips: An Investigation of Contextual Video Assistance for Functionality Understanding”. In: *CHI ’10* (2010), pp. 1515–1524. DOI: 10.1145/1753326.1753552. URL: <http://doi.acm.org/10.1145/1753326.1753552>.
- [95] Tovi Grossman, George Fitzmaurice, and Ramtin Attar. “A Survey of Software Learnability: Metrics, Methodologies and Guidelines”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’09. Boston, MA, USA: ACM, 2009, pp. 649–658. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518803. URL: <http://doi.acm.org/10.1145/1518701.1518803>.

- [96] Tovi Grossman, Justin Matejka, and George Fitzmaurice. "Chronicle: Capture, Exploration, and Playback of Document Workflow Histories". In: *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*. UIST '10. New York, New York, USA: ACM, 2010, pp. 143–152. ISBN: 978-1-4503-0271-5. DOI: 10.1145/1866029.1866054. URL: <http://doi.acm.org/10.1145/1866029.1866054>.
- [97] Ankit Gupta, Maneesh Agrawala, Brian Curless, and Michael Cohen. "MotionMontage: A System to Annotate and Combine Motion Takes for 3D Animations". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, 2014, pp. 2017–2026. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557218. URL: <http://doi.acm.org/10.1145/2556288.2557218>.
- [98] Ankit Gupta, Dieter Fox, Brian Curless, and Michael Cohen. "DuploTrack: A Reatime System for Authoring and Guiding Duplo Model Assembly". In: *Proceedings of the 25th annual ACM symposium adjunct on User interface software and technology*. Cambridge, Massachusetts, USA: ACM, 2012.
- [99] Pavel Gurevich, Joel Lanir, Benjamin Cohen, and Ran Stone. "TeleAdvisor: a versatile augmented reality tool for remote assistance". In: *CHI '12*. ACM Press, 2012.
- [100] Abir Al-Hajri, Gregor Miller, Matthew Fong, and Sidney S. Fels. "Visualization of Personal History for Video Navigation". In: *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, 2014, pp. 1187–1196. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557106. URL: <http://doi.acm.org/10.1145/2556288.2557106>.
- [101] SM Harrison. "A Comparison of Still, Animated, or Nonillustrated On-line Help with Written or Spoken Instructions in a Graphical User Interface." In: *CHI '95*. 1995.
- [102] Rachel Heck, Michael Wallick, and Michael Gleicher. "Virtual videography". In: *ACM Trans. Multimedia Comput. Commun. Appl.* 3.1 (2007). ISSN: 1551-6857. DOI: 10.1145/1198302.1198306. URL: <http://doi.acm.org/10.1145/1198302.1198306>.
- [103] Jeffrey Heer, Nathaniel S Good, Ana Ramirez, Marc Davis, and Jennifer Mankoff. "Presiding over accidents: system direction of human action". In: *Proceedings of CHI*. ACM Press, 2004, pp. 463–470.
- [104] Julie Heiser, Doantam Phan, Maneesh Agrawala, Barbara Tversky, and Pat Hanrahan. "Identification and Validation of Cognitive Design Principles for Automated Generation of Assembly Instructions". In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '04. Gallipoli, Italy: ACM, 2004, pp. 311–319. ISBN: 1-58113-867-9. DOI: 10.1145/989863.989917. URL: <http://doi.acm.org/10.1145/989863.989917>.
- [105] 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.
- [106] Steven Henderson and Steven Feiner. "Exploring the benefits of augmented reality documentation for maintenance and repair". In: *IEEE Trans on Visualization and Computer Graphics* 17.10 (2011), pp. 1355–1368.

- [107] Hexo+. *Your Self-Flying Camera - Auto Follow Gopro Drone*. 2016. URL: <https://hexoplus.com/> (visited on 07/30/2016).
- [108] David Hodson. *Motorola Droid RAZR Battery Replacement*. 2012. URL: <https://www.ifixit.com/Guide/Motorola+Droid+RAZR+Battery+Replacement/8057> (visited on 07/30/2016).
- [109] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. “Direct Manipulation Interfaces”. In: *Hum.-Comput. Interact.* 1.4 (Dec. 1985), pp. 311–338. ISSN: 0737-0024. DOI: 10.1207/s15327051hci0104_2. URL: http://dx.doi.org/10.1207/s15327051hci0104_2.
- [110] IBM. *IBM Chef Watson*. 2016. URL: <https://www.ibmchefwatson.com/> (visited on 07/30/2016).
- [111] Autodesk Inc. *Instructables*. 2016. URL: <http://www.instructables.com/tag/type-id/> (visited on 06/15/2016).
- [112] Jeff A. Johnson and Bonnie A. Nardi. “Creating Presentation Slides: A Study of User Preferences for Task-specific Versus Generic Application Software”. In: *ACM Trans. Comput.-Hum. Interact.* 3.1 (Mar. 1996), pp. 38–65. ISSN: 1073-0516. DOI: 10.1145/226159.226161. URL: <http://doi.acm.org/10.1145/226159.226161>.
- [113] Neel Joshi, Sisil Mehta, Steven Drucker, Eric Stollnitz, Hugues Hoppe, Matt Uyttendaele, and Michael Cohen. “Cliplets: juxtaposing still and dynamic imagery”. In: *Proceedings of UIST*. ACM Press, 2012, pp. 251–260.
- [114] Wendy Ju, Rebecca Hurwitz, Tilke Judd, and Bonny Lee. “CounterActive: An Interactive Cookbook for the Kitchen Counter”. In: *CHI '01 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '01. Seattle, Washington: ACM, 2001, pp. 269–270. ISBN: 1-58113-340-5. DOI: 10.1145/634067.634227. URL: <http://doi.acm.org/10.1145/634067.634227>.
- [115] Thorsten Karrer, Malte Weiss, Eric Lee, and Jan Borchers. “DRAGON: A Direct Manipulation Interface for Frame-accurate In-scene Video Navigation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '08. Florence, Italy: ACM, 2008, pp. 247–250. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357097. URL: <http://doi.acm.org/10.1145/1357054.1357097>.
- [116] Caitlin Kelleher and Randy Pausch. “Stencils-based Tutorials: Design and Evaluation”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '05. Portland, Oregon, USA: ACM, 2005, pp. 541–550. ISBN: 1-58113-998-5. DOI: 10.1145/1054972.1055047. URL: <http://doi.acm.org/10.1145/1054972.1055047>.
- [117] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. “3D object manipulation in a single photograph using stock 3D models”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 127.

- [118] Joy Kim, Mira Dontcheva, Wilmot Li, Michael S. Bernstein, and Daniela Steinsapir. “Motif: Supporting Novice Creativity Through Expert Patterns”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. Seoul, Republic of Korea: ACM, 2015, pp. 1211–1220. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702507. URL: <http://doi.acm.org/10.1145/2702123.2702507>.
- [119] 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*. UIST ’14. Honolulu, Hawaii, USA: ACM, 2014, pp. 563–572. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647389. URL: <http://doi.acm.org/10.1145/2642918.2647389>.
- [120] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. “Crowdsourcing Step-by-step Information Extraction to Enhance Existing How-to Videos”. In: *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 4017–4026. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2556986. URL: <http://doi.acm.org/10.1145/2556288.2556986>.
- [121] David Kirk and Danae Stanton Fraser. “Comparing Remote Gesture Technologies for Supporting Collaborative Physical Tasks”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’06. Montré#233;al, Qu#233;bec, Canada: ACM, 2006, pp. 1191–1200. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124951. URL: <http://doi.acm.org/10.1145/1124772.1124951>.
- [122] Kiteman. *How to Write an Instructable*. 2011. URL: <http://www.instructables.com/id/How-to-Write-an-Instructable-1/> (visited on 06/15/2016).
- [123] René F. Kizilcec, Kathryn Papadopoulos, and Lalida Sritanyaratana. “Showing Face in Video Instruction: Effects on Information Retention, Visual Attention, and Affect”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 2095–2102. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557207. URL: <http://doi.acm.org/10.1145/2556288.2557207>.
- [124] Scott R. Klemmer, Björn Hartmann, and Leila Takayama. “How Bodies Matter: Five Themes for Interaction Design”. In: *Proceedings of the 6th Conference on Designing Interactive Systems*. DIS ’06. University Park, PA, USA: ACM, 2006, pp. 140–149. ISBN: 1-59593-367-0. DOI: 10.1145/1142405.1142429. URL: <http://doi.acm.org/10.1145/1142405.1142429>.
- [125] Jarrod Knibbe, Tovi Grossman, and George Fitzmaurice. “Smart Makerspace: An Immersive Instructional Space for Physical Tasks”. In: *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. ITS ’15. Madeira, Portugal: ACM, 2015, pp. 83–92. ISBN: 978-1-4503-3899-8. DOI: 10.1145/2817721.2817741. URL: <http://doi.acm.org/10.1145/2817721.2817741>.

- [126] Nicholas Kong, Tovi Grossman, Björn Hartmann, Maneesh Agrawala, and George Fitzmaurice. “Delta: A Tool for Representing and Comparing Workflows”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’12. Austin, Texas, USA: ACM, 2012, pp. 1027–1036. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208549. URL: <http://doi.acm.org/10.1145/2207676.2208549>.
- [127] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. “Webzeitgeist: Design Mining the Web”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: ACM, 2013, pp. 3083–3092. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2466420. URL: <http://doi.acm.org/10.1145/2470654.2466420>.
- [128] Stacey Kuznetsov and Eric Paulos. “Rise of the Expert Amateur: DIY Projects, Communities, and Cultures”. In: *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*. NordiCHI ’10. Reykjavik, Iceland: ACM, 2010, pp. 295–304. ISBN: 978-1-60558-934-3. DOI: 10.1145/1868914.1868950. URL: <http://doi.acm.org/10.1145/1868914.1868950>.
- [129] Bum Chul Kwon and Bongshin Lee. “A Comparative Evaluation on Online Learning Approaches Using Parallel Coordinate Visualization”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. Santa Clara, California, USA: ACM, 2016, pp. 993–997. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858101. URL: <http://doi.acm.org/10.1145/2858036.2858101>.
- [130] B Lafreniere, A Bunt, M Lount, M Terry, and D Cowan. “Looks cool, I’ll try this later!”: Understanding the faces and uses of online tutorials”. In: *University of Waterloo Tech Report* (2012).
- [131] Ben Lafreniere, Andrea Bunt, Matthew Lount, and Michael Terry. “Understanding the Roles and Uses of Web Tutorials”. In: *ICWSM* (2013).
- [132] Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. “Community enhanced tutorials: improving tutorials with multiple demonstrations”. In: *CHI 2013*. ACM Press, 2013, p. 1779.
- [133] Bongshin Lee, Rubaiat Habib Kazi, and Greg Smith. “SketchStory: Telling More Engaging Stories with Data Through Freeform Sketching”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (Dec. 2013), pp. 2416–2425. ISSN: 1077-2626. DOI: 10.1109/TVCG.2013.191. URL: <http://dx.doi.org/10.1109/TVCG.2013.191>.
- [134] Eric Lee, Marius Wolf, and Jan Borchers. “Improving Orchestral Conducting Systems in Public Spaces: Examining the Temporal Characteristics and Conceptual Models of Conducting Gestures”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’05. Portland, Oregon, USA: ACM, 2005, pp. 731–740. ISBN: 1-58113-998-5. DOI: 10.1145/1054972.1055073. URL: <http://doi.acm.org/10.1145/1054972.1055073>.

- [135] Wei Li, Tovi Grossman, and George Fitzmaurice. “CADament: A Gamified Multiplayer Software Tutorial System”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, 2014, pp. 3369–3378. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2556954. URL: <http://doi.acm.org/10.1145/2556288.2556954>.
- [136] Wilmot Li, Maneesh Agrawala, Brian Curless, and David Salesin. “Automated generation of interactive 3D exploded view diagrams”. In: *ACM Transactions on Graphics (TOG)*. Vol. 27. 3. ACM. 2008, p. 101.
- [137] 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*. ICMI '03. Vancouver, British Columbia, Canada: ACM, 2003, pp. 234–241. ISBN: 1-58113-621-8. DOI: 10.1145/958432.958476. URL: <http://doi.acm.org/10.1145/958432.958476>.
- [138] 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*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 547–556. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518786. URL: <http://doi.acm.org/10.1145/1518701.1518786>.
- [139] Henry Lieberman, Elizabeth Rosenzweig, and Christopher Fry. “Steptorials: Mixed-initiative Learning of High-functionality Applications”. In: *Proceedings of the 19th International Conference on Intelligent User Interfaces*. IUI '14. Haifa, Israel: ACM, 2014, pp. 359–364. ISBN: 978-1-4503-2184-6. DOI: 10.1145/2557500.2557543. URL: <http://doi.acm.org/10.1145/2557500.2557543>.
- [140] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. “Subspace video stabilization”. In: *ACM Trans. Graph.* 30.1 (2011), 4:1–4:10. ISSN: 0730-0301. DOI: 10.1145/1899404.1899408. URL: <http://doi.acm.org/10.1145/1899404.1899408>.
- [141] David Lumb. “Manga Generator” Uses The Kinect To Put Your Smooth Moves In A Custom Comic. Sept. 2013. URL: <http://www.fastcolabs.com/3016870> (visited on 04/08/2016).
- [142] W. E. Mackay. “EVA: an experimental video annotator for symbolic analysis of video data”. In: *SIGCHI Bull.* 21.2 (1989), pp. 68–71. ISSN: 0736-6906. DOI: 10.1145/70609.70617. URL: <http://doi.acm.org/10.1145/70609.70617>.
- [143] MagicLens. *Your interactive Sales-App*. 2016. URL: <http://www.magiclensapp.com/> (visited on 07/30/2016).
- [144] Lena Mamykina, Elizabeth Mynatt, and Michael A. Terry. “Time Aura: Interfaces for Pacing”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '01. Seattle, Washington, USA: ACM, 2001, pp. 144–151. ISBN: 1-58113-327-8. DOI: 10.1145/365024.365077. URL: <http://doi.acm.org/10.1145/365024.365077>.

- [145] Justin Matejka, Tovi Grossman, and George Fitzmaurice. “Ambient Help”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’11. Vancouver, BC, Canada: ACM, 2011, pp. 2751–2760. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979349. URL: <http://doi.acm.org/10.1145/1978942.1979349>.
- [146] Justin Matejka, Tovi Grossman, and George Fitzmaurice. “IP-QAT: In-product Questions, Answers, & Tips”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, California, USA: ACM, 2011, pp. 175–184. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047218. URL: <http://doi.acm.org/10.1145/2047196.2047218>.
- [147] Justin Matejka, Tovi Grossman, and George Fitzmaurice. “Swifter: Improved Online Video Scrubbing”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: ACM, 2013, pp. 1159–1168. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2466149. URL: <http://doi.acm.org/10.1145/2470654.2466149>.
- [148] Justin Matejka, Wei Li, Tovi Grossman, and George Fitzmaurice. “CommunityCommands: Command Recommendations for Software Applications”. In: *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*. UIST ’09. Victoria, BC, Canada: ACM, 2009, pp. 193–202. ISBN: 978-1-60558-745-5. DOI: 10.1145/1622176.1622214. URL: <http://doi.acm.org/10.1145/1622176.1622214>.
- [149] Tara Matthews. “Designing and Evaluating Glanceable Peripheral Displays”. In: *Proceedings of the 6th Conference on Designing Interactive Systems*. DIS ’06. University Park, PA, USA: ACM, 2006, pp. 343–345. ISBN: 1-59593-367-0. DOI: 10.1145/1142405.1142457. URL: <http://doi.acm.org/10.1145/1142405.1142457>.
- [150] Richard E Mayer, William Bove, Alexandra Bryman, Rebecca Mars, and Lene Tapangco. “When less is more: Meaningful learning from visual and verbal summaries of science textbook lessons.” In: *Journal of educational psychology* 88.1 (1996), p. 64.
- [151] Scott McCloud. *Understanding Comics*. English. Reprint edition. New York: Avon, 1994. ISBN: 9780060976255.
- [152] Microsoft. *Microsoft HoloLens — Skype*. 2016. URL: <https://www.microsoft.com/microsoft-hololens/en-us/apps/skype> (visited on 07/10/2016).
- [153] P. Mijksenaar and P. Westendorp. *Open here: the art of instructional design*. Joost Elffers Books, 1999. URL: <https://books.google.com/books?id=fsJVAAAAMAAJ>.
- [154] Niloy J Mitra, Yong-Liang Yang, Dong-Ming Yan, Wilmot Li, and Maneesh Agrawala. “Illustrating how mechanical assemblies work”. In: *ACM Transactions on Graphics-TOG* 29.4 (2010), p. 58.
- [155] Peter Mohr, Bernhard Kerbl, Michael Donoser, Dieter Schmalstieg, and Denis Kalkofen. “Retargeting Technical Documentation to Augmented Reality”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. Seoul, Republic of Korea: ACM, 2015, pp. 3337–3346. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702490. URL: <http://doi.acm.org/10.1145/2702123.2702490>.

- [156] Roxana Moreno and Richard Mayer. “Interactive Multimodal Learning Environments”. In: *Educational Psychology Review* 19.3 (2007), pp. 309–326. DOI: 10.1007/s10648-007-9047-2. URL: <http://dx.doi.org/10.1007/s10648-007-9047-2>.
- [157] Eggo Müller. “Where quality matters: discourses on the art of making a YouTube video”. In: *The YouTube Reader*. Stockholm: National Library of Sweden, 2009.
- [158] Brad A. Myers, David A. Weitzman, Andrew J. Ko, and Duen H. Chau. “Answering Why and Why Not Questions in User Interfaces”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’06. Montré#233;al, Qu#233;bec, Canada: ACM, 2006, pp. 397–406. ISBN: 1-59593-372-7. DOI: 10.1145/1124772.1124832. URL: <http://doi.acm.org/10.1145/1124772.1124832>.
- [159] Toshio Nakamura and Takeo Igarashi. “An Application-independent System for Visualizing User Operation History”. In: *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. UIST ’08. Monterey, CA, USA: ACM, 2008, pp. 23–32. ISBN: 978-1-59593-975-3. DOI: 10.1145/1449715.1449721. URL: <http://doi.acm.org/10.1145/1449715.1449721>.
- [160] Cuong Nguyen and Feng Liu. “Making Software Tutorial Video Responsive”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI ’15. Seoul, Republic of Korea: ACM, 2015, pp. 1565–1568. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702209. URL: <http://doi.acm.org/10.1145/2702123.2702209>.
- [161] Cuong Nguyen, Yuzhen Niu, and Feng Liu. “Direct Manipulation Video Navigation in 3D”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: ACM, 2013, pp. 1169–1172. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2466150. URL: <http://doi.acm.org/10.1145/2470654.2466150>.
- [162] Phuc Nguyen, Mahesh Ravindranatha, Anh Nguyen, Richard Han, and Tam Vu. “Investigating Cost-effective RF-based Detection of Drones”. In: *Proceedings of the 2Nd Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*. DroNet ’16. Singapore, Singapore: ACM, 2016, pp. 17–22. ISBN: 978-1-4503-4405-0. DOI: 10.1145/2935620.2935632. URL: <http://doi.acm.org/10.1145/2935620.2935632>.
- [163] K Okumura, H Oku, and M Ishikawa. “High-speed gaze controller for millisecond-order pan/tilt camera”. In: *ICRA 2011: IEEE International Conference on Robotics and Automation* (2011).
- [164] Fred Paas, Alexander Renkl, and John Sweller. “Cognitive load theory and instructional design: Recent developments”. In: *Educational psychologist* 38.1 (2003), pp. 1–4.
- [165] Susan Palmiter and Jay Elkerton. “Animated Demonstrations for Learning Procedural Computer-based Tasks”. In: *Hum.-Comput. Interact.* 8.3 (Sept. 1993), pp. 193–216. ISSN: 0737-0024. DOI: 10.1207/s15327051hci0803_1. URL: http://dx.doi.org/10.1207/s15327051hci0803_1.

- [166] Susan Palmiter, Jay Elkerton, and P. Baggett. “Animated Demonstrations vs. Written Instructions for Learning Procedural Tasks: A Preliminary Investigation”. In: *Int. J. Man-Mach. Stud.* 34.5 (May 1991), pp. 687–701. ISSN: 0020-7373. DOI: 10.1016/0020-7373(91)90019-4. URL: [http://dx.doi.org/10.1016/0020-7373\(91\)90019-4](http://dx.doi.org/10.1016/0020-7373(91)90019-4).
- [167] Costas Panagiotakis and Georgios G Tziritas. “A speech/music discriminator based on RMS and zero-crossings”. In: *IEEE Transactions on Multimedia* 7.1 (2005), pp. 155–166.
- [168] Amy Pavel, Floraine Berthouzoz, Björn Hartmann, and Maneesh Agrawala. *Browsing and Analyzing the Command-Level Structure of Large Collections of Image Manipulation Tutorials*. Tech. rep. UCB/EECS-2013-167. EECS Department, University of California, Berkeley, Oct. 2013. URL: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-167.html>.
- [169] Amy Pavel, Dan Goldman, Björn Hartmann, and Maneesh Agrawala. “Video-based Asynchronous Video Review”. In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST ’16. ACM, 2016.
- [170] Amy Pavel, Colorado Reed, 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*. UIST ’14. Honolulu, Hawaii, USA: ACM, 2014, pp. 573–582. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647400. URL: <http://doi.acm.org/10.1145/2642918.2647400>.
- [171] Kevin Pfeil, Seng Lee Koh, and Joseph LaViola. “Exploring 3D Gesture Metaphors for Interaction with Unmanned Aerial Vehicles”. In: *Proceedings of the 2013 International Conference on Intelligent User Interfaces*. IUI ’13. Santa Monica, California, USA: ACM, 2013, pp. 257–266. ISBN: 978-1-4503-1965-2. DOI: 10.1145/2449396.2449429. URL: <http://doi.acm.org/10.1145/2449396.2449429>.
- [172] Edward Pincus. *The filmmaker’s handbook : a comprehensive guide for the digital age*. New York, New York: Plume, 2012. ISBN: 978-0452297289.
- [173] M. Polanyi. *Personal Knowledge: Towards a Post-critical Philosophy*. Gifford lectures p. 762; p. 1958. University of Chicago Press, 1958. URL: <https://books.google.com/books?id=JUIinQEACAAJ>.
- [174] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F Cohen. “Pause-and-play: automatically linking screencast video tutorials with applications”. In: *Proceedings of UIST*. ACM Press, 2011, pp. 135–144.
- [175] Y Pritch, S Ratovitch, and A Hendel. “Clustered synopsis of surveillance video”. In: *Proceedings of AVSS*. IEEE Computer Society, 2009.
- [176] Vidya Ramesh, Charlie Hsu, Maneesh Agrawala, and Björn Hartmann. “ShowMeHow: Translating User Interface Instructions Between Applications”. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST ’11. Santa Barbara, California, USA: ACM, 2011, pp. 127–134. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047212. URL: <http://doi.acm.org/10.1145/2047196.2047212>.

- [177] Abhishek Ranjan, Jeremy P. Birnholtz, and Ravin Balakrishnan. “Dynamic shared visual spaces: experimenting with automatic camera control in a remote repair task”. In: *Proceedings of CHI*. San Jose, California, USA: ACM Press, 2007, pp. 1177–1186. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240802. URL: <http://doi.acm.org/10.1145/1240624.1240802>.
- [178] Abhishek Ranjan, Jeremy Birnholtz, and Ravin Balakrishnan. “Improving meeting capture by applying television production principles with audio and motion detection”. In: CHI ’08. Florence, Italy: ACM, 2008, pp. 227–236. ISBN: 978-1-60558-011-1. DOI: 10.1145/1357054.1357095. URL: <http://doi.acm.org/10.1145/1357054.1357095>.
- [179] Abhishek Ranjan, Rorik Henrikson, Jeremy Birnholtz, Ravin Balakrishnan, and Dana Lee. “Automatic camera control using unobtrusive vision and audio tracking”. In: *Proceedings of Graphics Interface 2010*. GI ’10. Ottawa, Ontario, Canada: Canadian Information Processing Society, 2010, pp. 47–54. ISBN: 978-1-56881-712-5. URL: <http://dl.acm.org/citation.cfm?id=1839214.1839224>.
- [180] Mark O. Riedl and Vadim Bulitko. “Interactive narrative: An intelligent systems approach”. In: *AI Magazine* (2013).
- [181] Mike Roberts and Pat Hanrahan. “Generating Dynamically Feasible Trajectories for Quadrotor Cameras”. In: *ACM Trans. Graph.* 35.4 (July 2016), 61:1–61:11. ISSN: 0730-0301. DOI: 10.1145/2897824.2925980. URL: <http://doi.acm.org/10.1145/2897824.2925980>.
- [182] Daniela K. Rosner and Kimiko Ryokai. “Spyn: Augmenting Knitting to Support Storytelling and Reflection”. In: *Proceedings of the 10th International Conference on Ubiquitous Computing*. UbiComp ’08. Seoul, Korea: ACM, 2008, pp. 340–349. ISBN: 978-1-60558-136-1. DOI: 10.1145/1409635.1409682. URL: <http://doi.acm.org/10.1145/1409635.1409682>.
- [183] Ben Ruedlinger. *Does Video Length Matter?* 2012. URL: <https://wistia.com/blog/does-length-matter-it-does-for-video-2k12-edition> (visited on 07/30/2016).
- [184] Gilbert Ryle. “Knowing How and Knowing That: The Presidential Address”. In: *Proceedings of the Aristotelian Society* 46 (1945), pp. 1–16. ISSN: 00667374, 14679264. URL: <http://www.jstor.org/stable/4544405>.
- [185] Advait Sarkar, Cecily Morrison, Jonas F. Dorn, Rishi Bedi, Saskia Steinheimer, Jacques Boisvert, Jessica Burggraaff, Marcus D’Souza, Peter Kotschieder, Samuel Rota Bulò, Lorcan Walsh, Christian P. Kamm, Yordan Zaykov, Abigail Sellen, and Siân Lindley. “Setwise Comparison: Consistent, Scalable, Continuum Labels for Computer Vision”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. Santa Clara, California, USA: ACM, 2016, pp. 261–271. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858199. URL: <http://doi.acm.org/10.1145/2858036.2858199>.

- [186] Eldon Schoop, Michelle Nguyen, Daniel Lim, Valkyrie Savage, Sean Follmer, and Björn Hartmann. “Drill Sergeant: Supporting Physical Construction Projects Through an Ecosystem of Augmented Tools”. In: *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA '16. Santa Clara, California, USA: ACM, 2016, pp. 1607–1614. ISBN: 978-1-4503-4082-3. DOI: 10.1145/2851581.2892429. URL: <http://doi.acm.org/10.1145/2851581.2892429>.
- [187] Dorée Duncan Seligmann and Steven Feiner. “Automated Generation of Intent-based 3D Illustrations”. In: *SIGGRAPH Comput. Graph.* 25.4 (July 1991), pp. 123–132. ISSN: 0097-8930. DOI: 10.1145/127719.122732. URL: <http://doi.acm.org/10.1145/127719.122732>.
- [188] Moushumi Sharmin, Lawrence Bergman, Jie Lu, and Ravi Konuru. “On Slide-based Contextual Cues for Presentation Reuse”. In: *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*. IUI '12. Lisbon, Portugal: ACM, 2012, pp. 129–138. ISBN: 978-1-4503-1048-2. DOI: 10.1145/2166966.2166992. URL: <http://doi.acm.org/10.1145/2166966.2166992>.
- [189] Edward Yu-Te Shen, Henry Lieberman, and Glorianna Davenport. “What’s Next?: Emergent Storytelling from Video Collection”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 809–818. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518825. URL: <http://doi.acm.org/10.1145/1518701.1518825>.
- [190] Fuhao Shi, Hsiang-Tao Wu, Xin Tong, and Jinxiang Chai. “Automatic Acquisition of High-fidelity Facial Performances Using Monocular Videos”. In: *ACM Trans. Graph.* 33.6 (Nov. 2014), 222:1–222:13. ISSN: 0730-0301. DOI: 10.1145/2661229.2661290. URL: <http://doi.acm.org/10.1145/2661229.2661290>.
- [191] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. “Real-time human pose recognition in parts from single depth images”. In: *CVPR* (2011).
- [192] Timothy P. Smith, J. P. Singer, G. M. Balliro, and N. D. Lerner. *Manufacturer’s Guide to Developing Consumer Product Instructions*. 2003.
- [193] Cynthia Solomon, Amartya Banerjee, and Michael S. Horn. “Ultimate Trainer: Instructional Feedback for Ultimate Frisbee Players”. In: *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction*. TEI '14. Munich, Germany: ACM, 2013, pp. 137–140. ISBN: 978-1-4503-2635-3. DOI: 10.1145/2540930.2540965. URL: <http://doi.acm.org/10.1145/2540930.2540965>.
- [194] Ryan Spicer, Yu-Ru Lin, Aisling Kelliher, and Hari Sundaram. “NextSlidePlease: Authoring and Delivering Agile Multimedia Presentations”. In: *ACM Trans. Multimedia Comput. Commun. Appl.* 8.4 (Nov. 2012), 53:1–53:20. ISSN: 1551-6857. DOI: 10.1145/2379790.2379795. URL: <http://doi.acm.org/10.1145/2379790.2379795>.

- [195] Yuta Sugiura, Daisuke Sakamoto, Anusha Withana, Masahiko Inami, and Takeo Igarashi. “Cooking with Robots: Designing a Household System Working in Open Environments”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’10. Atlanta, Georgia, USA: ACM, 2010, pp. 2427–2430. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753693. URL: <http://doi.acm.org/10.1145/1753326.1753693>.
- [196] Jeff Suovanen. *iPhone 6s Plus Display Assembly Replacement*. 2016. URL: <https://www.ifixit.com/Guide/iPhone+6s+Plus+Display+Assembly+Replacement/55423> (visited on 07/30/2016).
- [197] John Sweller. “Cognitive load during problem solving: Effects on learning”. In: *Cognitive science* 12.2 (1988), pp. 257–285.
- [198] John Sweller, Jeroen JG Van Merriënboer, and Fred GWC Paas. “Cognitive architecture and instructional design”. In: *Educational psychology review* 10.3 (1998), pp. 251–296.
- [199] Daniel Szafrir, Bilge Mutlu, and Terrence Fong. “Communication of Intent in Assistive Free Flyers”. In: *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*. HRI ’14. Bielefeld, Germany: ACM, 2014, pp. 358–365. ISBN: 978-1-4503-2658-2. DOI: 10.1145/2559636.2559672. URL: <http://doi.acm.org/10.1145/2559636.2559672>.
- [200] Anthony Tang and Sebastian Boring. “#EpicPlay: Crowd-sourcing Sports Video Highlights”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’12. Austin, Texas, USA: ACM, 2012, pp. 1569–1572. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208622. URL: <http://doi.acm.org/10.1145/2207676.2208622>.
- [201] Richard Tang, Hesam Alizadeh, Anthony Tang, Scott Bateman, and Joaquim A.P. Jorge. “Physio@Home: Design Explorations to Support Movement Guidance”. In: *CHI ’14 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 1651–1656. ISBN: 978-1-4503-2474-8. DOI: 10.1145/2559206.2581197. URL: <http://doi.acm.org/10.1145/2559206.2581197>.
- [202] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, Arran Topalian, Erroll Wood, Sameh Khamis, Pushmeet Kohli, Toby Sharp, Shahram Izadi, Richard Banks, Andrew Fitzgibbon, and Jamie Shotton. “Efficient and Precise Interactive Hand Tracking through Joint, Continuous Optimization of Pose and Correspondences”. In: *ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques*. 2016.
- [203] Laura Teodosio and Walter Bender. “Salient Stills”. In: *ACM Trans. Multimedia Comput. Commun. Appl.* 1.1 (Feb. 2005), pp. 16–36. ISSN: 1551-6857. DOI: 10.1145/1047936.1047940. URL: <http://doi.acm.org/10.1145/1047936.1047940>.
- [204] J. Thies, M. Zollhöfer, M. Stamminger, C. Theobalt, and M. Nießner. “Face2Face: Real-time Face Capture and Reenactment of RGB Videos”. In: *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*. 2016.

- [205] Cristen Torrey, Elizabeth F. Churchill, and David W. McDonald. “Learning How: The Search for Craft Knowledge on the Internet”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 1371–1380. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518908. URL: <http://doi.acm.org/10.1145/1518701.1518908>.
- [206] Cristen Torrey, David W. McDonald, Bill N. Schilit, and Sara Bly. “How-To pages: Informal systems of expertise sharing”. In: *ECSCW 2007: Proceedings of the 10th European Conference on Computer-Supported Cooperative Work*. London: Springer London, 2007, pp. 391–410. DOI: 10.1007/978-1-84800-031-5_21. URL: http://dx.doi.org/10.1007/978-1-84800-031-5_21.
- [207] Tiffany Tseng. “Spin: A Photography Turntable System for Creating Animated Documentation”. In: *Proceedings of the 14th International Conference on Interaction Design and Children*. IDC '15. Boston, Massachusetts: ACM, 2015, pp. 422–425. ISBN: 978-1-4503-3590-4. DOI: 10.1145/2771839.2771869. URL: <http://doi.acm.org/10.1145/2771839.2771869>.
- [208] Tiffany Tseng and Mitchel Resnick. “Product Versus Process: Representing and Appropriating DIY Projects Online”. In: *Proceedings of the 2014 Conference on Designing Interactive Systems*. DIS '14. Vancouver, BC, Canada: ACM, 2014, pp. 425–428. ISBN: 978-1-4503-2902-6. DOI: 10.1145/2598510.2598540. URL: <http://doi.acm.org/10.1145/2598510.2598540>.
- [209] Edward Tufte. *Layering and Separation*. Cheshire, Connecticut: Graphics Press, 1990. Chap. 3. ISBN: 978-0961392116.
- [210] Daisuke Uriu, Mizuki Namai, Satoru Tokuhisa, Ryo Kashiwagi, Masahiko Inami, and Naohito Okude. “Panavi: Recipe Medium with a Sensors-embedded Pan for Domestic Users to Master Professional Culinary Arts”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: ACM, 2012, pp. 129–138. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2207695. URL: <http://doi.acm.org/10.1145/2207676.2207695>.
- [211] Lauren Vasey, Tovi Grossman, Heather Kerrick, and Danil Nagy. “The Hive: A Human and Robot Collaborative Building Process”. In: *ACM SIGGRAPH 2016 Talks*. SIGGRAPH '16. Anaheim, California: ACM, 2016, 83:1–83:2. ISBN: 978-1-4503-4282-7. DOI: 10.1145/2897839.2927404. URL: <http://doi.acm.org/10.1145/2897839.2927404>.
- [212] Ron Wakkary, Markus Lorenz Schilling, Matthew A. Dalton, Sabrina Hauser, Audrey Desjardins, Xiao Zhang, and Henry W.J. Lin. “Tutorial Authorship and Hybrid Designers: The Joy (and Frustration) of DIY Tutorials”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, pp. 609–618. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702550. URL: <http://doi.acm.org/10.1145/2702123.2702550>.

- [213] Cheng-Yao Wang, Wei-Chen Chu, Hou-Ren Chen, Chun-Yen Hsu, and Mike Y. Chen. “EverTutor: Automatically Creating Interactive Guided Tutorials on Smartphones by User Demonstration”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’14. Toronto, Ontario, Canada: ACM, 2014, pp. 4027–4036. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557407. URL: <http://doi.acm.org/10.1145/2556288.2557407>.
- [214] Robert West, Ryen W. White, and Eric Horvitz. “From Cookies to Cooks: Insights on Dietary Patterns via Analysis of Web Usage Logs”. In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW ’13. Rio de Janeiro, Brazil: ACM, 2013, pp. 1399–1410. ISBN: 978-1-4503-2035-1. DOI: 10.1145/2488388.2488510. URL: <http://doi.acm.org/10.1145/2488388.2488510>.
- [215] wikiHow. *wikiHow: Statistics*. 2016. URL: <http://www.wikihow.com/Special:Statistics> (visited on 06/15/2016).
- [216] Andrew Wilson, Hrvoje Benko, Shahram Izadi, and Otmar Hilliges. “Steerable augmented reality with the Beamatron”. In: *UIST 2012*. 2012, pp. 413–422.
- [217] University of Wisconsin–Superior. *Example Video Script - Topic "Food Safety"*. 2016. URL: <https://www.uwsuper.edu/cetl/digitalstorytelling/> (visited on 06/20/2016).
- [218] Li-Chen Wu, I-Chen Lin, and Ming-Han Tsai. “Augmented Reality Instruction for Object Assembly Based on Markerless Tracking”. In: *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D ’16. Redmond, Washington: ACM, 2016, pp. 95–102. ISBN: 978-1-4503-4043-4. DOI: 10.1145/2856400.2856416. URL: <http://doi.acm.org/10.1145/2856400.2856416>.
- [219] Xiao Xiao and Hiroshi Ishii. “Inspect, Embody, Invent: A Design Framework for Music Learning and Beyond”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. Santa Clara, California, USA: ACM, 2016, pp. 5397–5408. ISBN: 978-1-4503-3362-7. DOI: 10.1145/2858036.2858577. URL: <http://doi.acm.org/10.1145/2858036.2858577>.
- [220] Ross Yeager. “An Automated Physiotherapy Exercise Generator”. MA thesis. EECS Department, University of California, Berkeley, May 2013. URL: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-91.html>.
- [221] Tom Yeh, Tsung-Hsiang Chang, and Robert C Miller. “Sikuli: using GUI screenshots for search and automation”. In: *UIST ’09*. ACM Press, 2009, pp. 183–192.
- [222] Ben Zhang, Yu-Hsiang Chen, Claire Tuna, Achal Dave, Yang Li, Edward Lee, and Björn Hartmann. “HOBS: Head Orientation-based Selection in Physical Spaces”. In: *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction*. SUI ’14. Honolulu, Hawaii, USA: ACM, 2014, pp. 17–25. ISBN: 978-1-4503-2820-3. DOI: 10.1145/2659766.2659773. URL: <http://doi.acm.org/10.1145/2659766.2659773>.

- [223] Yupeng Zhang, Teng Han, Zhimin Ren, Nobuyuki Umetani, Xin Tong, Yang Liu, Takaaki Shiratori, and Xiang Cao. “BodyAvatar: Creating Freeform 3D Avatars Using First-person Body Gestures”. In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST '13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 387–396. ISBN: 978-1-4503-2268-3. DOI: 10.1145/2501988.2502015. URL: <http://doi.acm.org/10.1145/2501988.2502015>.
- [224] Douglas E. Zongker and David H. Salesin. “On Creating Animated Presentations”. In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '03. San Diego, California: Eurographics Association, 2003, pp. 298–308. ISBN: 1-58113-659-5. URL: <http://dl.acm.org/citation.cfm?id=846276.846319>.
- [225] Amit Zoran and Joseph A. Paradiso. “FreeD: A Freehand Digital Sculpting Tool”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, 2013, pp. 2613–2616. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481361. URL: <http://doi.acm.org/10.1145/2470654.2481361>.

Vita

Pei-Yu (Peggy) Chi received her M.S. from the MIT Media Lab and her M.S. in Computer Science and B.B.A. in Information Management from National Taiwan University. She develops interactive systems that support users' creativity and learning activities. Her research has received a Best Paper Award at ACM CHI, a Google PhD Fellowship in Human-Computer Interaction, a Berkeley Fellowship for Graduate Study, and a MIT Media Lab Fellowship with ITRI. After five years of study in Computer Science at UC Berkeley, Peggy received the Doctor of Philosophy degree in August 2016. She will join Google Inc. as a research scientist in Mountain View, California.