UNIVERSITY OF CALIFORNIA SAN DIEGO

Contextually Recommending Expert Help and Demonstrations to Improve Creativity

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Cristin Ailidh Fraser

Committee in charge:

Scott R. Klemmer, Chair
Mira Dontcheva
William G. Griswold
Philip J. Guo
Björn Hartmann
James D. Hollan

2020

The dissertation of Cristin Ailidh Fraser is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

_____

Chair

University of California San Diego

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am grateful to Scott and Mira for their endless patience, encouragement, support, and wisdom. Thank you for believing in my research even when I didn't. When I first visited UCSD, Scott took me for a walking meeting through the gorgeous eucalyptus forest. Not only was this a genius recruiting tactic, it gave me a first glimpse into his unique approach to research, communication, and life. I've learned so much more from Scott than I realize even now, and am ever grateful for the tight-knit community he fostered in our lab and amongst his students. Mira and I first met in a North Toronto coffee shop, and she has continued to awe and inspire me ever since with her incredible mind, generous heart, and endless passion for her work. It has been such a joy to work with Mira and I could not be more excited to continue doing so at Adobe. I'm rarely sure of anything in my life, but I am sure that this next step is exactly where I want to go.

I am lucky to have such a talented and supportive committee. Thanks to Jim for always giving me such positive and encouraging feedback and guidance, and always being open to a chat. Thanks to Philip for reminding me why my research is exciting when I lose steam, and for so much valuable advice over the years. Thank you to Bill for many stimulating conversations about research, photography, and life. And finally, thanks to Björn for guiding my work with insightful questions and thoughtful feedback.

The Design Lab and CSE Department have both fostered wonderful communities that I am grateful to be a part of. I am thankful to Don Norman for his leadership of the Design Lab and for welcoming me on that first day back in 2014 when I shyly walked into the lab. Thank you to the many members of the Design Lab who have given me so much valuable feedback. Thanks to Julia and James for being such wonderful mentees and collaborators, and for giving me the joy of seeing their passion for research grow. Thanks to Ian, Vanessa, Sara, Olga, and the entire Design Lab operations team past and present, for their indispensable support. And Teenah, who has supported me with her endless generosity since the very start: it's been a delight to learn and grow alongside you. Thank you to Julie for always keeping me on track, and for the many quick

meetings that turned into long conversations. I am grateful for funding from UCSD's Powell Fellowship, NSERC, Adobe Research, and the CSE Department.

I could not have gotten through this journey without an incredible support system of labmates, collaborators, and friends. Thank you to Cat for patiently teaching me the ropes and welcoming me with open arms into your San Diego world. Thank you to Vineet, Tricia, Ariel, and the rest of the crewtons for sushiboocha, coffee breaks, tea time, naps in the lab, late night deadline companionship, and everything else. Thank you Vineet for paving the way ahead, keeping my spirits up in our windowless office, and testing all your good (and bad) jokes on me. I'll always remember those three magical months when we got to work in the office with windows. Tricia, my two-time-award-winning co-author, thank you for keeping me sane through our research projects (especially the studies) and for always being awkward with me. Thank you Amy for all our adventures, and for the world's best napping chair. Thank you Tiffany for your life-changing support throughout this journey, and Vivian for going above and beyond in your care.

Thank you Ariana for being an exceptional roommate, friend, and colleague, and for your endless support and energy. In the time you have been here, you have transformed not only my life, but also GradWIC and the entire CSE Department. Thanks to all those who have contributed to GradWIC for helping us build an important and ever-growing community. Thank you to the many aerial friends and mentors I've made at UCSD and AR, especially Chava and her Angells, for getting me in better shape (physically and mentally) than I've ever been. Thank you Diana for following me to San Diego (sorry I'm leaving now) and including me in all your family adventures. Thank you Chris for growing with me this entire way, supporting me through the worst times, and celebrating with me in the best times. You make me a better person, and I can't wait to continue this adventure together.

I am grateful to Adobe and Autodesk for supporting me through four summer internships. To have spent one summer reading all of Tovi Grossman and George Fitzmaurice's papers, and

then the following summer writing a paper with both of them was an absolute dream. I am grateful to have worked with and learned from so many exceptional researchers: Joel Brandt, Joy Kim, Valentina Shin, Holger Winnemöller, Sheryl Ehrlich, Andy Wilson, Alison Thornsberry, Fraser Anderson, and others. Thanks to the many fellow interns I've shared these experiences with, especially the Miracles (Yea-Seul, Jasper, Amanda, Daniel) and my Canadian buddies, David and Rahul. Thanks to Minsuk, who I've somehow never actually done an internship with but who was never too far away. Possibly one of the most successful meetups enabled by Confer! Thanks to Jordana, Rachel, and Willy for the luckiest summer roommate situation.

Diane Horton's passion for teaching and constant encouragement is what first got me interested in Computer Science, and for that I will be forever grateful. Thanks also to Karen Reid and Paul Gries for their incredible teaching and mentorship. Thank you to Kyros Kutulakos and Matt O'Toole for showing me how to do good research. Thank you to my former office-mate and soon-to-be colleague Peter O'Donovan for introducing me to HCI and to Scott's research.

And finally, thanks to my family. To my mother Nancy for all the crucial stats help, my father Don for telling people I was doing a PhD before I'd even decided to, my sister Donelle for her never-ending kindness and generosity, and of course Sammy and Tibby.

Chapter 3, in part, includes portions of material as it appears in *RePlay: Contextually Presenting Learning Videos Across Software Applications* by C. Ailie Fraser, Tricia J. Ngoon, Mira Dontcheva, and Scott Klemmer in the Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is currently being prepared for submission for publication of the material. C. Ailie Fraser, Julia M. Markel, N. James Basa, Mira Dontcheva, and Scott Klemmer. The dissertation author was the primary investigator and author of this material.

Chapter 5, in part, includes portions of material as it appears in *Sharing the Studio: How*

Creative Livestreaming can Inspire, Educate, and Engage by C. Ailie Fraser, Joy O. Kim, Alison Thornsberry, Scott Klemmer, and Mira Dontcheva in the Proceedings of the 2019 on Creativity and Cognition (C&C '19). The dissertation author was the primary investigator and author of this paper.

Chapter 5, in part, includes portions of material coauthored with Andy Edmonds and Mira Dontcheva. The dissertation author was the primary investigator and author of this material.

Chapter 6, in part, includes portions of material as it appears in *DiscoverySpace: Suggesting Actions in Complex Software* by C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer in the Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16). The dissertation author was the primary investigator and author of this paper.

Chapter 7, in part, includes portions of material as it appears in *Interactive Guidance Techniques for Improving Creative Feedback* by Tricia J. Ngoon, C. Ailie Fraser, Ariel S. Weingarten, Mira Dontcheva, and Scott Klemmer in the Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). The dissertation author was one of the primary investigators and authors of this paper.

Appendix A, in part, includes portions of material as it appears in *Sharing the Studio: How Creative Livestreaming can Inspire, Educate, and Engage* by C. Ailie Fraser, Joy O. Kim, Alison Thornsberry, Scott Klemmer, and Mira Dontcheva in the Proceedings of the 2019 on Creativity and Cognition (C&C '19). The dissertation author was the primary investigator and author of this paper.

VITA

2013	Honours B.Sc. with High Distinction, Specialist in Math & Computer Science, Major in Music, University of Toronto, Canada
2016	M.S. in Computer Science, University of California San Diego
2020	Ph. D. in Computer Science, University of California San Diego

PUBLICATIONS

C. Ailie Fraser, Joy Kim, Valentina Shin, Joel Brandt, and Mira Dontcheva. 2020. Temporal Segmentation of Creative Live Streams. To appear in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*.

C. Ailie Fraser, Mira Dontcheva, Joy O. Kim, and Scott Klemmer. 2019. How live streaming does (and doesn't) change creative practices. *interactions* 27, 1 (December 2019), 46–51.

C. Ailie Fraser, Joy O. Kim, Alison Thornsberry, Scott Klemmer, and Mira Dontcheva. 2019. Sharing the Studio: How Creative Livestreaming can Inspire, Educate, and Engage. In *Proceedings of the 2019 on Creativity and Cognition (C&C '19)*. Association for Computing Machinery, New York, NY, USA, 144–155.

C. Ailie Fraser, Tricia J. Ngoon, Mira Dontcheva, and Scott Klemmer. 2019. RePlay: Contextually Presenting Learning Videos Across Software Applications. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, Paper 297, 1–13.

Tricia J. Ngoon, C. Ailie Fraser, Ariel S. Weingarten, Mira Dontcheva, and Scott Klemmer. 2018. Interactive Guidance Techniques for Improving Creative Feedback. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, Paper 55, 1–11.

C. Ailie Fraser, Tovi Grossman, and George Fitzmaurice. 2017. WeBuild: Automatically Distributing Assembly Tasks Among Collocated Workers to Improve Coordination. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery, New York, NY, USA, 1817–1830.

C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer. 2016. DiscoverySpace: Suggesting Actions in Complex Software. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16)*. Association for Computing Machinery, New York, NY, USA, 1221–1232.

Catherine M. Hicks, Vineet Pandey, C. Ailie Fraser, and Scott Klemmer. 2016. Framing Feedback: Choosing Review Environment Features that Support High Quality Peer Assessment. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. Association for Computing Machinery, New York, NY, USA, 458–469.

ABSTRACT OF THE DISSERTATION

Contextually Recommending Expert Help and Demonstrations to Improve Creativity

by

Cristin Ailidh Fraser

Doctor of Philosophy in Computer Science

University of California San Diego, 2020

Scott R. Klemmer, Chair

Translating goals into concrete actions is a major challenge for people doing creative activities. Expert examples, tutorials, and videos abound online, helping people find inspiration and learn from the best. However, online resources are detached from the user's work. The user must determine which resource is most relevant to their particular situation, and adapt it to their context. The sensemaking challenge of using help resources combined with the open-endedness of creative work can be tedious at best, and paralyzing at worst, preventing people from reaching their full creative potential.

My dissertation introduces methods for **curating existing expert help and demonstrations** and **presenting them to users in context**, with the goal of better supporting the iterative

creative process and thus enabling people to produce better work. Leveraging existing expert-made resources enables people to learn from others more easily, and placing them in context promotes serendipity and unexpected connections.

When performing creative activities, people can be driven by processes and/or outcomes: sometimes they are interested in understanding the process behind creative tasks, while other times they wish to reach a particular outcome the fastest and easiest way possible. This dissertation presents four interactive systems that explore how three different types of expert resource – **visual media**, **executable code**, and **written text** – can help people accomplish one or both of the above goals at a level beyond what they could have achieved otherwise.

Specifically, *RePlay* and *ReMap* support both process and outcome by enabling in-task search and navigation of help videos. *LiveClips* supports discovery and exploration of expert processes by embedding inspirational clips from live streamed videos into the user's workflow. *DiscoverySpace* helps novices discover and achieve expert outcomes by recommending action macros in-situ. Finally, *CritiqueKit* combines adaptive suggestions with interactive guidance to help novices improve their outcomes in the moment.

Together, these systems and their evaluations demonstrate how curating and presenting expert help and demonstrations in-situ can help novices build confidence, accomplish tasks, and produce higher-quality creative work. My dissertation enables more people to reach their creative potential by lowering the barriers to getting started and completing projects.

Chapter 1

Supporting Creative Software Activities

People love to be creative [208]. Software gives people the ability to produce incredible work, from illustrations to movies to visualizations. However, to allow the flexibility and control that leads to great work, creative software applications often grow bloated with features [156]. This creates a steep learning curve: since creative activities often comprise multiple steps that each use different tools, the gulf of execution [98] between these tools and the user's goals can be daunting. People spend years practicing and taking formal training to master creative software like Adobe Photoshop or Autodesk Maya. The prevalence of free and subscription-based software means that casual users now have access to software that was once only used by professionals. Anyone can make their own animations or illustrations... if they can figure out how.

While the ideal way to get help with creative software might be to ask an expert in person (Figure 1.1 left), one-on-one tutoring like this does not scale as the number of available experts exceeds the number of people seeking help [18]. Instead, people seek help online from resources shared by expert users, such as tutorials, examples, video demonstrations, and discussions. Web search provides a powerful way to quickly browse and access these resources, but it is detached from the user's work. The user must determine which resource is most relevant to their situation, and adapt it to their context [41,50,58,123]. In addition, creative work is often open-ended and

features multiple paths to success, or even multiple definitions of success [210]. The sensemaking challenge of using help resources combined with the open-endedness of creative work can be tedious at best, and paralyzing at worst, preventing people from reaching their full creative potential [210].

This dissertation introduces methods for **curating existing expert help and demonstrations** and **presenting them to users in context**, with the goal of better supporting the creative process and thus enabling people to produce better work. Leveraging existing expert-made resources enables people to learn from others more easily, and placing them in context promotes serendipity and unexpected connections.

The type of expert resource that is most helpful will differ based on whether the user wishes to understand a *process*, reach an *outcome*, or both (Figure 1.2). Sometimes people are interested in understanding the process behind creative tasks, such as when learning a domain they wish to become an expert at. Other times, people are interested only in a particular outcome, and want it done the fastest and easiest way possible. Understanding a process requires theoretical or conceptual knowledge (such as an explanation about how the process works), while achieving an outcome requires practical steps (such as a concise step-by-step guide) [186]. For example,

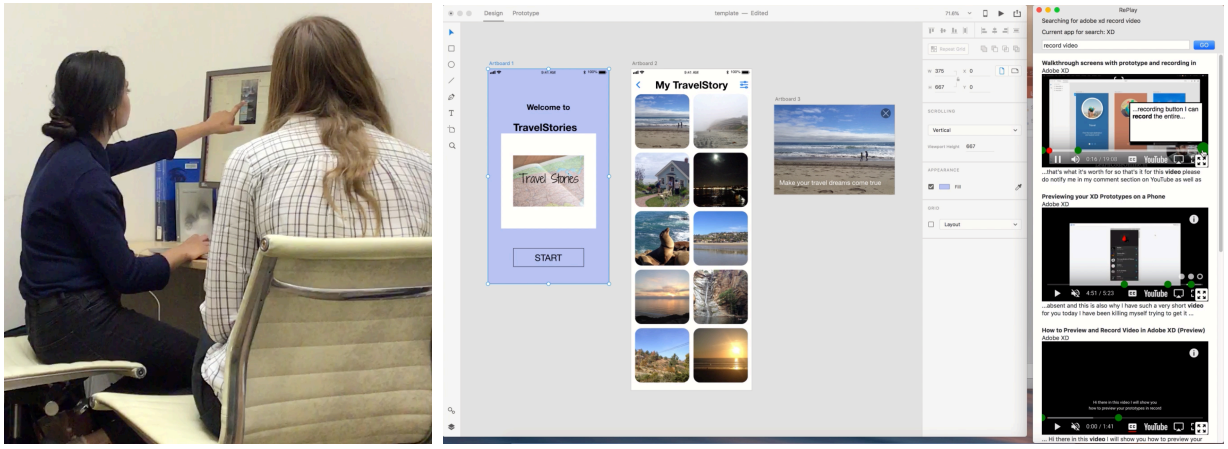

Figure 1.1: This dissertation aims to bring automated help one step closer to the personalized help we get from human tutors (left). For example, a screenshot from our evaluation of RePlay (right) shows a participant leveraging in-context video help to complete a creative task.

someone seeking to become an expert at poster-making in InDesign would benefit from a detailed video explaining the concepts of good poster design and how to make a poster from scratch, while someone who needs to make a poster for an event by tomorrow might prefer using a template they can paste their event details into. A user who seeks to understand a process may or may not have a desired outcome; sometimes people follow tutorials or explore application features simply for the sake of learning the application or getting inspired, while other times they seek to learn while doing a particular task [122, 186].

This dissertation presents four interactive systems that explore how three different types of expert resource – **visual media**, **executable code**, and **written text** – can help people understand

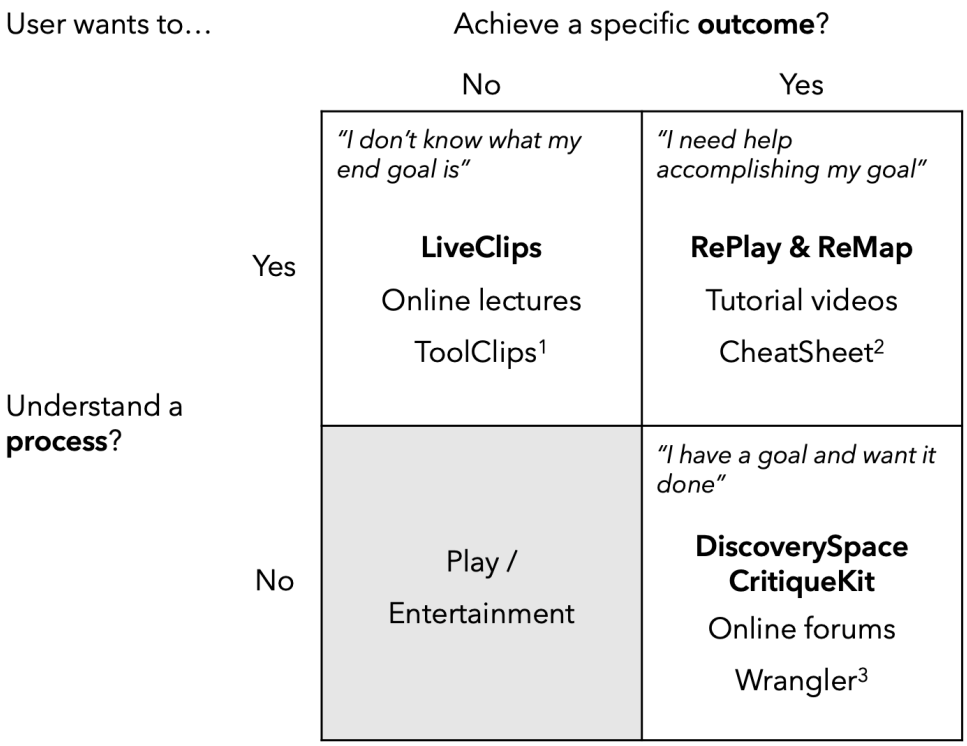

Figure 1.2: The interactive systems presented in this dissertation cover three of the above four quadrants depending on the goals of the user: whether they seek to understand the *process* behind how something is done and/or whether they seek to accomplish a specific *outcome*. Bolded entries are presented in this dissertation. Non-bolded entries are examples of widely-used solutions on the web (first) and research systems that integrate with specific applications (second). [1] [77]; [2] [231]; [3] [104].

processes and/or produce outcomes beyond what they could have achieved otherwise (Figure 1.2). Specifically, visual media in the form of screencast videos are used to understand processes, both for achieving specific outcomes (*RePlay* & *ReMap*) and for general inspiration (*LiveClips*). Executable code (*DiscoverySpace*) and written text (*CritiqueKit*) are both used for achieving specific outcomes. Together, these systems demonstrate how expert resources can be reused, repurposed, and curated to provide people with the help they need, when they need it.

1.1 Visual Media For Understanding Processes

Videos are a popular learning resource for all kinds of tasks, especially creative ones [40, 165, 184]. Tutorial videos that provide step-by-step instructions for a specific task are helpful for people whose goal is to learn how to do that task. Placing videos in context allows people to apply the information directly to the task at hand, promoting active learning [20]. Other types of videos such as live streams support casual learning and inspiration by giving viewers a window into the entire, unstructured process behind a real-world project. Placing live streams in context can allow people to discover new techniques and get ideas they might not have otherwise thought of. Chapters 3-5 of this dissertation demonstrate how each of these two types of video can be curated and presented in context to support in-the-moment process understanding, with and without a specific desired outcome.

1.1.1 Search and Navigation of Help Videos Supports Process and Outcome

Tutorial videos are a popular way for creators to impart knowledge and learners to absorb it. However, videos are difficult to search, browse, and navigate, because they show a single unstructured stream of information and can typically only be explored by scrubbing on the timeline [112, 177, 178]. Chapters 3-4 of this dissertation hypothesize that integrating search and navigation of help videos into the user's task and enhancing them with relevant context helps

users achieve procedural support toward accomplishing desired outcomes. I embody this insight in the RePlay and ReMap systems for contextually presenting help videos. RePlay and ReMap gather context about the user's activity across different software applications, using it to find relevant videos. Multimodal search and contextual presentation of videos allow users to search and interact with results without taking their attention away from the task at hand. A field study ($n = 7$) and lab study ($n = 24$) showed that contextual video assistance helps people spend less time away from their task than web video search and replaces current video navigation strategies with more efficient workflows. An initial study with multimodal search ($n = 13$) showed promise while also revealing important challenges with using speech and pointing for help-seeking.

1.1.2 Recommendation of Live Stream Clips Supports Process Exploration

Tutorials are helpful for accomplishing specific goals, but people do not always have a particular outcome in mind when using creative software; they may not even know what outcomes are possible [37, 153, 170]. Some people watch live stream videos of artists sharing their process to get inspired and learn about new techniques. However, finding moments that are relevant and personally inspiring from these long, unstructured videos is tedious and difficult, as such moments may be few and far between. Chapter 5 of this dissertation hypothesizes that extracting contextually-relevant clips from creative live stream videos and recommending them inside creative software promotes serendipitous discovery and inspiration throughout the creative process. I embody this insight in the LiveClips system, which automatically generates and presents clips showing demonstrations of relevant tools. Formative studies motivate this approach by exploring how and why people currently watch creative live streams. Initial user feedback suggested that this is a promising approach, and an online study showed that our algorithm reasonably predicts a clip's inspirational value.

1.2 Code and Text For Achieving Outcomes

Not everyone wants to learn the process behind a task; some people just want to get a task done, for which video demonstrations are too detailed. Chapters 6-7 of this dissertation demonstrate how expert-created examples of executable code and written text can help people explore and achieve outcomes quickly and easily.

1.2.1 Executable Code Suggestions Enable Quick Exploration of Outcomes

A key difficulty users of complex software often face is the *gulf of execution*: the gap between user intentions and the available instructions or facilities the software provides for achieving them [98]. Achieving a goal requires combining multiple low-level operations. This can be prohibitively difficult for novices, who are often already overwhelmed and/or uninterested in low-level details. One way to package a sequence of low-level operations into a single high-level action is to record the operations as a macro. Experts already share macros online to the user community; what if novices could more easily benefit from them?

Chapter 6 of this dissertation hypothesizes that integrating expert-made macros directly into the software streamlines their usage by reducing the steps and complexity needed to achieve advanced outcomes. I embody this insight in DiscoverySpace, a plugin for Photoshop that curates relevant photo-editing action macros shared online by the user community, and recommends relevant ones to the user for immediate use on their own photo. DiscoverySpace helps novices explore possibilities and try out advanced techniques in complex creative software. A between-subjects study ($n = 28$) showed that action suggestions prevent novices from losing confidence in their abilities and help users to accomplish tasks and discover new features.

1.2.2 Adaptive Suggestions of Text Examples Improve Outcomes

Some creative tasks are complex not because of specific software, but because of the tasks themselves [23]. Providing feedback on creative work is one such example. Receiving useful feedback is critical for improving one's creative work, but people are rarely taught how to *give* good feedback, and developing this skill takes time and practice. How might we help novice feedback givers improve the usefulness of their feedback in the moment, while they are giving it?

Chapter 7 of this dissertation hypothesizes that combining examples of expert feedback with guidance that highlights their structure helps reviewers improve their feedback. I embody this insight in CritiqueKit, a system for providing feedback on creative work that incorporates suggested examples and adaptive guidance. Two real-world deployment studies ($n = 37$) and two controlled experiments ($n = 87$) showed that adaptive suggestions combined with interactive guidance improved the quality of feedback from novice reviewers.

1.3 Thesis Statement and Contributions

The four systems described above demonstrate how carefully curating and presenting expert help and demonstrations can support both understanding processes and achieving outcomes in creative software. Together, these systems and their evaluations support my thesis statement:

> *Curating expert help & demonstrations and placing them in context enables people to produce better work.*

Each system introduces an approach for curating and ranking a different type of expert resource mined from the web (tutorial and live stream videos, executable macros, and written feedback) and a novel way to interact with these resources in-situ. Online expert help and demonstrations present a unique opportunity for creative support as they are readily available and ripe with knowledge, yet hard to find and make sense of by traditional methods. Leveraging context from the user's environment and activities provides a common ground [43] for connecting

users with relevant information. And finally, giving people the right resource at the right time by bringing it into their environment supports just-in-time learning and doing, enabling people to achieve outcomes that would otherwise be out of their reach.

1.3.1 Contributions

My dissertation contributes the following:

- Algorithms that curate and rank four different types of expert resource (tutorial videos, live stream videos, action macros, and written feedback) based on their relevance to the user's context
- Interfaces that present these curated resources in-situ for users to search, browse, navigate, and use in the context of their own work
- Evaluations of these algorithms and interfaces that demonstrate the efficacy of contextually presenting expert resources to creative software users
- Formative knowledge about how people currently approach creative activities and use the web for creative support

Chapter 8 presents a discussion of the challenges that remain and open questions prompted by this research. It also discusses how the approaches presented may generalize beyond creative software tasks to other types of tasks and domains. This dissertation demonstrates the potential of bringing knowledge from user communities to everyone's fingertips, with the hope that expert help can move away from the siloed world of the web and into the creative process where everyone can benefit from it.

Chapter 2

Related Work

Prior research has leveraged expert help and demonstrations to support people learning and using creative and complex software. In this chapter, I first describe the challenges of using web search to find expert resources. Then, I discuss how different types of expert resources support the different user goals of understanding processes, achieving outcomes, and both together. Through examples of both out-of-context and in-context assistance, I highlight the benefits of presenting help and demonstrations in context, motivating the systems in this dissertation.

2.1 Expert Resources Abound Online but are Hard to Use

People often need help when using complex software due to its steep learning curve [5]. Most software provides some in-situ assistance and official documentation, but these are typically limited to introductory tasks and individual tool descriptions [123]. As a result, people often turn to expert resources shared in online user communities, such as tutorials, discussion fora, examples, and video demonstrations. Expert resources are more abundant than ever, as they are relatively easy to make and share online [122, 184]. Web search provides a powerful way to quickly browse and access these resources, but it poses three key problems. First, users must articulate an appropriate query, an almost paradoxical requirement for users who are there because they don't know the domain [195]. Novices often don't have enough prior knowledge to even

know *what* to ask, let alone how to ask it [159]. Second, search is blind to relevant contextual information that could connect users to better resources [50, 55, 116]. People must know exactly what they're looking for, or spend time looking at potentially irrelevant information. Finally, search is divorced from the application UX, requiring users to bounce back and forth to connect online resources with their own task [58]. The systems in this dissertation bring online expert help and demonstrations into the user's context, and use contextual information to improve the relevance of the presented resources.

Searching is helpful when users know what they need, but novices often don't [159], and starting from a blank slate can be intimidating [107]. Proactively recommending help and examples in-situ based on user context can lead users to information they might not have even thought to search for [41, 99, 150, 153], and can spark creative inspiration [13, 92, 118]. All the systems in this paper therefore present *something* to the user by default: LiveClips, DiscoverySpace, and CritiqueKit all recommend resources automatically based on the user's behaviour or document. RePlay and ReMap present general videos related to the current application when opened, and encourage the user to search for the tools they are using by automatically adding their names to the search field when clicked.

The remainder of this chapter discusses relevant approaches for leveraging expert resources to help novices understand processes and achieve desired outcomes in creative or complex software. This prior work highlights the importance of presenting resources in context and using contextual information to improve their relevance and usefulness.

2.2 "I Need Help Accomplishing My Goal": Tutorials and Demonstrations Enable Learning While Doing

Online tutorials, demonstrations, and walkthroughs are popular resources for users of complex software looking for procedural help. Although they tend to show an entire task from

start to finish, one of the biggest reasons people seek both text [122] and video [6] tutorials is for in-task help with a specific goal. People learning complex software often "learn on the job," seeking process-related help for achieving a specific outcome (Figure 1.2 top right).

Prior work has made tutorials easier to navigate and understand by allowing people to add comments and questions at specific locations in text tutorials [26] and specific timestamps and visual locations in video tutorials [6], or recording multiple versions of tutorials so viewers can explore different use cases [124]. Video tutorials are especially helpful for learning when people are in the execution stage of creative work [242], because they show clear demonstrations that might be harder to explain or understand in text [40, 77, 184]. However, videos are harder to navigate and skim than text because because they show a single unstructured stream of information and can typically only be explored by scrubbing on the timeline with a small visual preview [112,177,178]. Segmenting videos into conceptual chunks [11, 40, 77, 150, 165, 184] and adding navigational cues [11, 40, 77, 78, 110, 112, 150, 165, 178, 184] can help viewers more easily find the information they need. Methods for navigating between video segments include interactive timeline markers [11,78,110,112,150], thumbnail images [11,40,77,110,178,184], transcript text [110], and clickable elements overlaid on the video [165]. For example, ToolScape [108, 112] shows a clickable thumbnail image and description of the tools used for each step of the video's process, highlights areas of the video with no visible progress, and allows users to browse storyboard views of videos based on their steps (Figure 2.1). These cues make it easier for users to quickly browse and navigate videos, and skip to or repeat relevant steps when following along, leading users to produce better work with higher self-efficacy [108,112]. However, they require manually annotating videos or crowdsourcing and verifying labels.

Systems for viewing tutorials on the web (like ToolScape [108, 112]) require the user to find a relevant tutorial for their current task from a set of search results. One way to improve the relevance of search results is to augment web search queries with context from the user's software [22, 50, 151]. Systems can also leverage user context to highlight relevant information

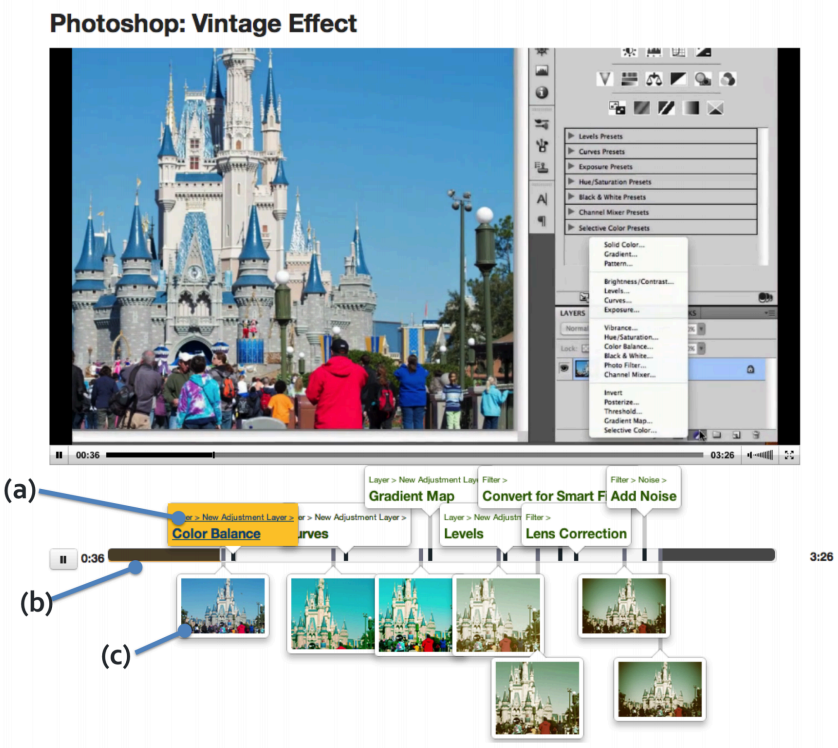

Figure 2.1: ToolScape helps users follow along with video tutorials by adding clickable descriptions (a) and thumbnail images (c) for each step of the video's process, and highlighting areas of the video with no visible progress (b). Figure originally published by Kim *et al.* [112].

within a result, making it easier for users to relate to their task [50, 58]. But even once the user finds a relevant result, viewing it in a separate window presents difficulties such as switching back and forth between the browser and the application and mapping screenshots or videos of the application to the user's own version [106, 184].

Embedding learning resources in software reduces the need for context switching, and supports active learning, or learning while doing [20, 75, 77, 99, 150, 153]. For example, interactive tutorials guide users step-by-step through hands-on example tasks inside the target application [106, 123, 184]. Such tutorials can even be generated automatically as a user demonstrates them [73], though this requires domain-specific heuristics. In-application search for tutorials [123] or question & answer forum posts [41] also helps users stay focused on their task while seeking help. CheatSheet [231] and Social CheatSheet [232] support manual and collabo-

rative curation (respectively) of help resources, which the systems then leverage to display relevant help in context while using web applications. Like the systems in this dissertation, CheatSheet and Social CheatSheet display help resources in a sidebar to make them easily accessible and searchable in context while completing a task (Figure 2.2).

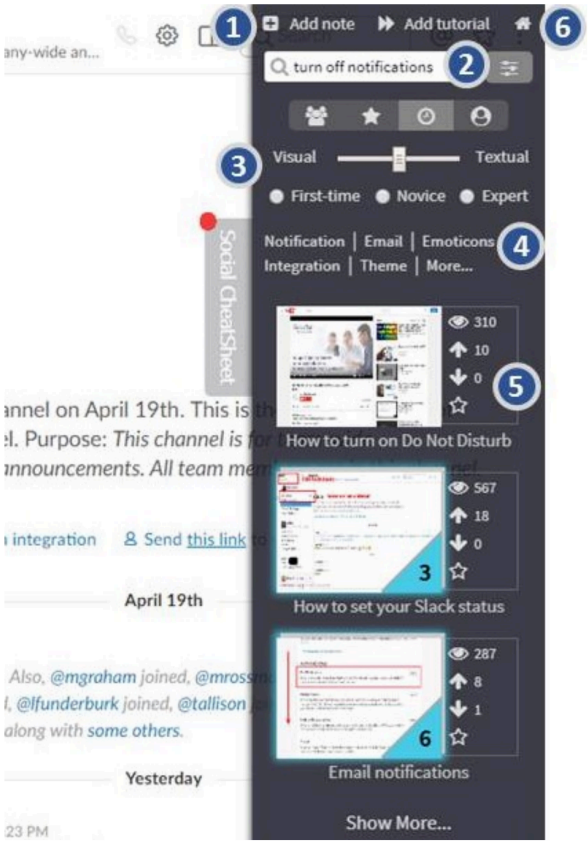

Figure 2.2: Social CheatSheet displays help resources for the user's current web application in a sidebar to make them easily accessible and searchable in context while completing a task. Figure originally published by Vermette *et al.* [232].

2.3 "I Don't Know What My End Goal Is": Proactive Suggestions Enable Discovery and Inspiration

Besides help with specific tasks, common general-purpose uses for tutorials include learning a new topic or software, studying for a class, or shadowing an expert [122] (Figure 1.2 top left). Other popular resources for these goals besides tutorials include online lectures via video (*e.g.*, through MOOC platforms like Coursera[1]) or podcast (*e.g.*, through platforms like iTunes U[2]), online textbooks, blogs, and live stream videos [142]. People use these resources not just for learning, but also for inspiration, motivation, and entertainment [39, 94, 142, 143].

Resources that encourage trying things out and/or discovering new features can be helpful when people are interested in process outside of a task. People sometimes tinker in software without having an explicit goal in mind, learning features by trying them out [27, 31]. To support this behaviour, interactive tutorials can help users learn and practice skills through predefined tasks [47, 106, 123]. Proactively suggesting related commands [136, 153] or videos [77, 150] can help users discover new features that are useful for their workflows or learn about new functionalities for tools they already use. For example, ToolClips [77] augments traditional tooltips that appear when hovering over a tool with video demonstrations of that tool being used (Figure 2.3). They are a good example of contextual assistance that is readily available in-situ but not distracting or obtrusive; ToolClips only appear if a user pauses while hovering over a tool, and are easily dismissed by mousing away. Users can view multiple videos of each tool to see different possible uses for them. Most importantly, the videos shown are short clips that show only the tool use and its surrounding context independent from any particular task, thus steering users directly to the relevant information.

This prior work focuses on helping users *learn* processes; what about when users seek *inspiration*? Inspiration is difficult to study because its unpredictable nature and influence makes

[1] `coursera.org`
[2] `apps.apple.com/us/app/itunes-u/id490217893`

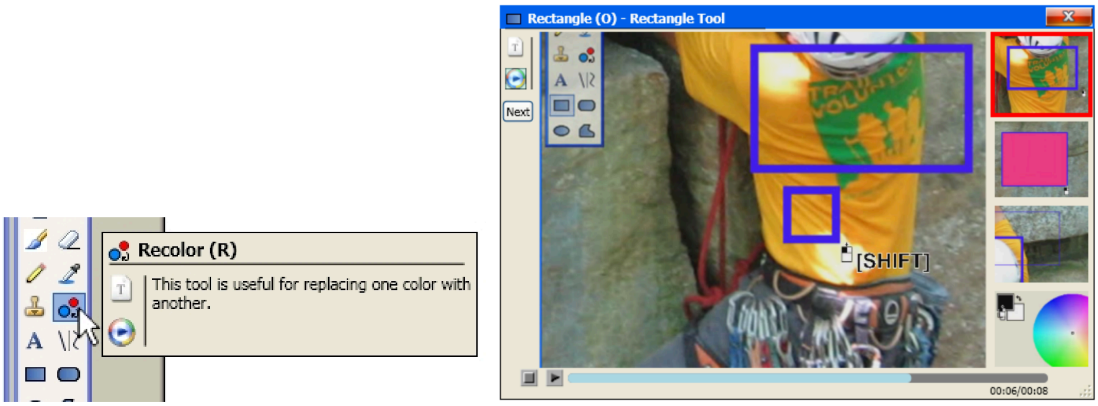

Figure 2.3: ToolClips helps users learn how to use software tools in-situ by augmenting traditional tooltips with video demonstrations. Clicking the media button on the tooltip (left) opens a video player (right) showing concise demonstrations of the tool being used. Figure originally published by Grossman *et al.* [77].

it hard to quantify [210]. While inspiration can stem from just about anywhere, one proven way to stimulate creative inspiration is browsing examples of other artists' work [13, 16, 56, 75, 161, 208, 210]. Creative work often combines preexisting works in novel ways [16], or draws new connections between seemingly disparate things [56]. Examples can be beneficial not just at the beginning of the creative process, but throughout [118, 190, 213]. LiveClips (Chapter 5) therefore recommends video examples to users in context. Unlike ToolClips [77], LiveClips takes its video clips from creative live streams and aims to *inspire* users rather than educate them.

2.4 "I Have a Goal and Want It Done": Operations and Answers Help People Accomplish Tasks

Users often have high-level goals when using creative software, such as making a photo look "better" [127]. To accomplish these goals, users must bridge the *gulf of execution* [98], figuring out which tools to combine and how to use them in order to achieve their goal. Sometimes, people don't have the time or patience to learn the details of each tool required – they simply

want to get a task *done* [127, 148] (Figure 1.2 bottom right).

The classic solution when one wants to get a task done but doesn't know how would be to ask a friend to do the task for you or provide help. Or, if you don't know who to ask, systems like Answer Garden [4] could automatically route a question to an appropriate expert for answers. On the web, people seek answers or results from experts by posting questions in discussion fora such as StackOverflow or requests in fora such as `r/PhotoshopRequest` on Reddit [148]. However, questions asked on the web lack the context that is normally available when seeking help in-person, such as system settings or details about what the user has already tried [10, 38]. Looking through previously-answered questions on discussion fora can help people find solutions without needing to post a new question, but reading through past discussions to find a single relevant answer is tedious and requires users to know what keywords they are looking for [41].

One way to bring question-asking into the user's context and ensure that relevant information is included is to have users record their question directly in the application [38]. Another is to embed discussion fora directly into the application, allowing users to narrow down questions based on relevant interface elements [41] or recently-used commands [151]. To achieve outcomes even more directly than by question and answer, other systems embed contextual search or suggestion of resources such as reusable examples [22], executable operations [104], solutions to errors [85], and software features [5]. Presenting such resources in context allows users to

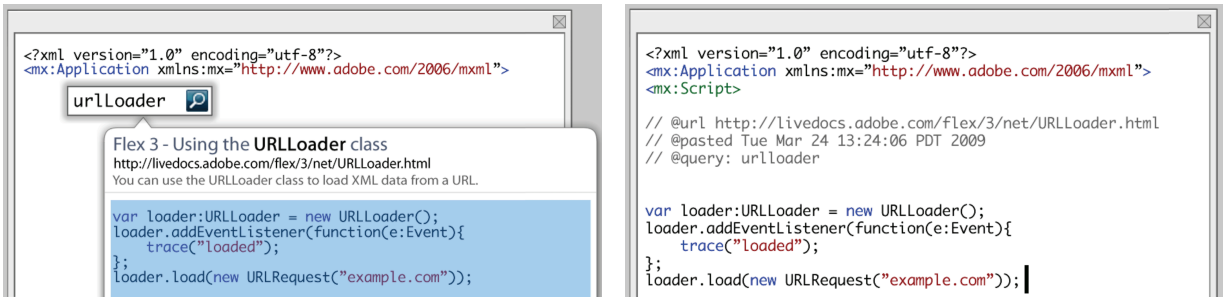

Figure 2.4: Blueprint enables in-context search and use of code examples from the web. It extracts code snippets from web pages (left) and allows users to integrate them into their code in one click (right). Figure originally published by Brandt *et al.* [22].

apply them directly to their work without having to stop their task, and allows systems to improve the relevance of results by leveraging contextual information. For example, Blueprint [22] enables in-context search of example code to help programmers more easily integrate code from the web into their own tasks (Figure 2.4). Blueprint extracts code examples from existing web resources, presents them in context, and allows users to instantly apply them to their own code. Blueprint also augments search queries with relevant context from the user's environment to find more relevant results. Like Blueprint, DiscoverySpace (Chapter 6) and CritiqueKit (Chapter 7) suggest reusable examples in context to help users achieve better outcomes more easily.

Chapter 3

RePlay: Contextually Presenting Learning Videos Across Software Applications

Complex activities often require people to work across multiple software applications. However, people frequently lack valuable knowledge about at least one application, especially as software changes and new software emerges. These gaps in knowledge often arise while people are in the middle of the task, leading them to search for help. Unfortunately, existing help systems either lack contextual knowledge or are tightly-knit into a single application. We introduce an application-independent approach for contextually presenting video learning resources and demonstrate it through the *RePlay* system. RePlay uses accessibility APIs to gather context about the user's activity. It leverages an existing search engine to present relevant videos and highlights key segments within them using video captions. We report on a week-long field study ($n = 7$) and a lab study ($n = 24$) showing that contextual assistance helps people spend less time away from their task than web video search and replaces current video navigation strategies. Our findings highlight challenges with representing and using context across applications.

3.1 Introduction

Most software activities span multiple applications. The slogan "there's an app for that" illustrates that we live in a world filled with specialized apps that each focus on a few specific tasks. To accomplish larger activities requires composing multiple applications together into a

"toolbelt" [221]. For example, designing an interface might comprise drawing a logo in Illustrator, mocking up a prototype in Sketch, adding animations using Flinto, and presenting it to a client using Keynote. Analyzing data might involve formatting it using Python, viewing and graphing it in Excel, modifying graph aesthetics in Photoshop, and reporting results in Word. Toolbelts help users tailor custom ecosystems and support distributed innovation. However, this bricolage creates a user experience problem: even with design guidelines, every application is different [14]. As new applications appear and existing ones change, few people are fluent experts in all the steps

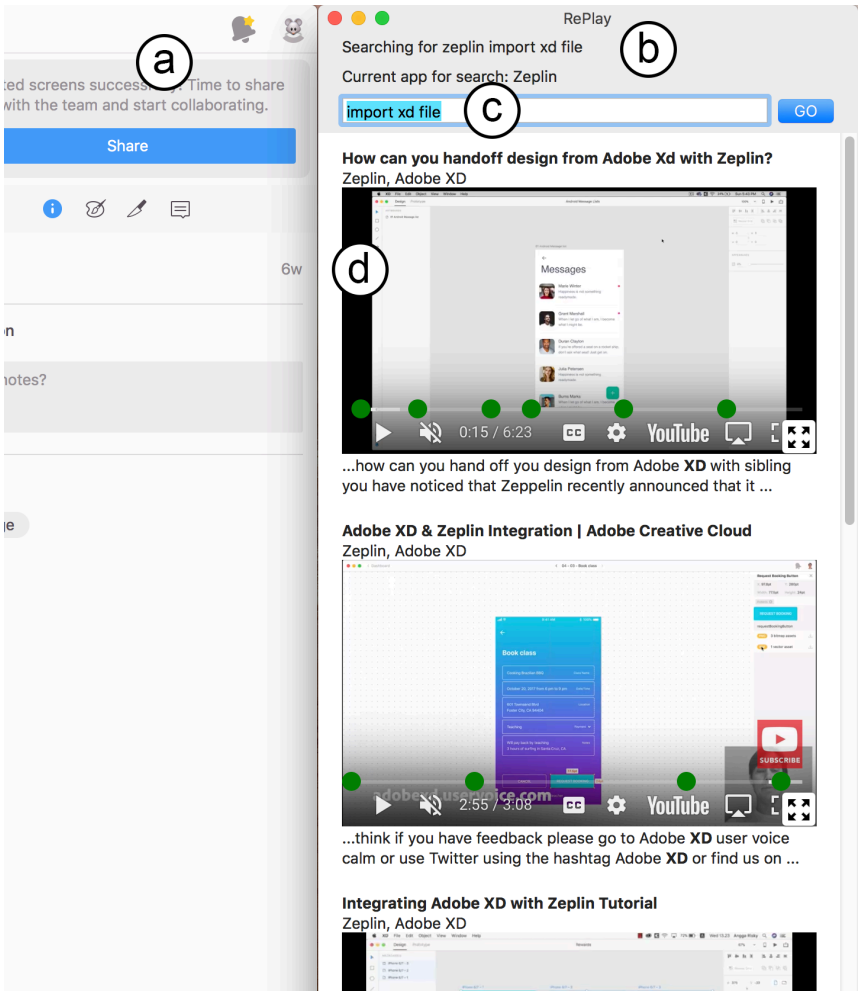

Figure 3.1: RePlay - shown to the right of Zeplin (a) - includes a status area (b) and search field (c) at the top. Video results (d) include timeline markers and caption excerpts to highlight moments relevant to the user's query and context.

towards their goals.

Presenting learning resources in-application [22, 41, 77, 99, 150, 151] and augmenting search queries with contextual information [22, 50] can offer a more fluid experience with lower cognitive load. However, existing solutions require deep integration with applications. And since today's applications are "walled gardens" with limited integration across software vendors [14], help resources typically focus on one application at a time. This leaves gaps when users want to move from one application to another (*e.g.,* export an Adobe XD prototype to Zeplin) or interleave applications (*e.g.,* coding a website in Sublime while debugging in Chrome and resizing graphics in GIMP).

We introduce an application-independent approach for contextually presenting video learning resources. We embody this approach in the RePlay system (Figure 3.1), which enables users to search for learning videos based on their application usage. RePlay gathers application context using system accessibility APIs. It extends online video search and cues videos to relevant segments based on their captions. We focus on video assistance because despite video's growing popularity (Cisco predicts that by 2021, 82% of all internet traffic will be video [2]), searching and browsing videos remain cumbersome [112, 177, 178]. Video is popular for content creators as it is often easier to author than tutorials or diagrams (which require careful curation). Learners value video for its efficacy in communicating complex or continuous visual actions such as brushing or setting parameters [40]. However, interacting with videos remains difficult because they are harder to navigate and scan for steps than text [40].

We report on two studies observing how people use RePlay and web video help: a week-long field study ($n=7$) and a lab study ($n=24$) where half the participants used RePlay and half used web video search. Both studies used visual design as the domain, as video is especially helpful for visual tasks [184]. The field study examined how designers with varying experience used RePlay in-situ. Participants used an average of 17 different applications in a week, emphasizing the importance of system-wide integration. Our findings also suggested that contextual video

assistance benefits targeted tasks more than open-ended ones. The lab study found that contextual video assistance helps people spend less time away from their task than web video search, and replaces strategies typically used in navigating between and within videos. This work makes the following contributions:

1. An application-independent method for finding relevant clips in learning videos that leverages user context,
2. the RePlay system, which demonstrates this method using accessibility APIs and online video corpora, and
3. insights from two studies that highlight the importance of multi-application support and the promise of cross-application search.

3.2 Related Work

This work builds on prior contextual search and video segmentation work with a novel focus on multi-application activities.

3.2.1 Multi-application Activities are Hard to Support Consistently

People often work across multiple applications that each support an individual task to perform higher-level activities [221]. Help systems for such applications tend to focus on their individual tasks rather than the transitions and interactions between applications [168]. Implementing system-wide assistance that captures activity context is difficult, as every application has its own conventions and interface, and software interoperability tends to be limited [14]. Accessibility APIs are one useful entry point for diverse system-wide extensibility, including visualizing user behavior [152], voice control [135, 243], and modifying or enhancing existing user interfaces [36, 46, 219]. RePlay uses accessibility APIs for detecting system-wide application context.

3.2.2 Application Context Improves Relevance and Presentation of Help Resources

Effectively searching the Web for help is an acquired skill; coming up with the right keywords and search settings can be difficult, especially for novices [195]. In addition, web search environments lack context that human tutors use to proactively offer help and tailor feedback [198]. Adding contextual terms automatically to search queries (*e.g.,* os version, application, recently-used tools) can help improve the relevance and utility of search results without requiring the user to know app-specific terminology [22, 50, 151]. RePlay augments queries with the current application name and uses context from both the current and recently-used applications to support cross-application activities when ranking search results.

Adding contextual cues to search results provides information scent that can help users more quickly and easily navigate the results (Figure 3.3). Such cues show how and why a result is relevant to the user's own situation and direct them to the relevant information within a result [50, 58]. RePlay expands these ideas to video, displaying contextual cues for recently-used applications and tools.

3.2.3 Videos are Popular for Learning but Hard to Search and Browse

Videos are widely available and relatively easy to make. Software is always being updated, and new video demonstrations are constantly added to popular platforms by the user community to keep up with updates and current trends. However, not only is navigating within a *single* video difficult and tedious [112, 177, 178], finding the right video in a list of search results can be even harder. The predominant video search approach displays results as thumbnail images with a title and summary (*e.g.,* YouTube, Vimeo). This presentation only provides a broad overview without cues about matching content; prior work has shown that people look for indications of how search results are relevant to their query [89].

Like some prior work [67, 177, 178], RePlay uses captions to select relevant clips from online videos. Automatic clip selection plays a limited but growing role in web search. Google displays a "suggested video clip" [220] as the automated summary for some searches, and Bing's "smart motion preview" feature [1] shows a 30-second preview for video search results. In contrast, RePlay focuses on searching with and presenting results within application context. Chi *et al.* [40] showed people benefit most from a mix of text and video; RePlay combines video and text instruction by presenting captions with videos for both navigational and learning assistance.

Like Ambient Help [150], ToolScape [112], and Chronicle [78], RePlay overlays markers on the timeline indicating tool use. We use timeline markers over other types of annotations as they take up little space and allow for pop-up text previews, which aid browsing. While prior work marks *all* instances of tool use and command execution, RePlay marks only *contextually-relevant* moments (recently-used tools and words from the user's query) to reduce clutter and unnecessary detail in a small interface.

3.3 RePlay System

This section describes RePlay's user interface, context detection, and system architecture.

3.3.1 User Interface

The RePlay panel (Figure 3.1) can be positioned and sized as desired; its default size is 465px ×1055px. It is designed to fit next to the user's primary applications to minimize switching windows. The narrow pane makes videos small but easier to browse and watch in context. The interface comprises a status area, search field, and video results.

The status area updates as the user works, displaying the name of the last tool clicked and the current application (Figure 3.2). "Tools" refer to clickable interface elements and commands within an application. The status area provides awareness of what context RePlay will use for

search (*i.e.,* so the user does not need to include the application name in their search query). When the user initiates a search, the status area updates to show the query (Figure 3.1b).

As the user works, the search field updates with the name of the last tool clicked (Figure 3.2). Users can edit or delete it to form their own query. Pressing the GO button or return key triggers a search. RePlay displays five video results, each cued to start at a relevant moment. A two-line excerpt from that moment's caption appears below the video with query words highlighted in bold [89].

Often, videos have multiple moments that may be relevant. RePlay renders green markers on the video timeline to indicate these moments. Mousing over a marker invokes a pop-up text area displaying a caption excerpt with words from the query in bold (Figure 3.3). This pop-up obscures YouTube's default thumbnail pop-up but provides more useful information, as software videos tend to show an entire screen and shrinking this to a thumbnail makes it hard to see. Clicking a marker starts the video from that moment.

RePlay also displays contextual cues [50] on search results (Figure 3.4). For each video, RePlay lists the three most-recently used applications that are mentioned in the video (Figure 3.4a). This list is especially useful when users move between applications and want videos that mention both the current and recent applications. RePlay renders grey timeline markers to indicate moments where recently-used tools are mentioned (Figure 3.4b). Caption pop-ups italicize tool names.

RePlay's panel shows all five results at the same time, allowing users to quickly skim

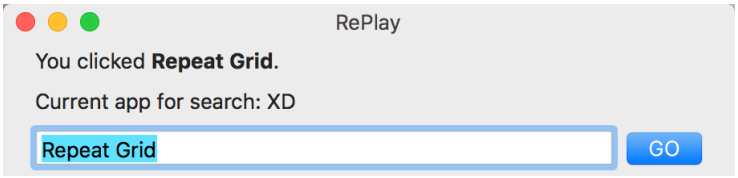

Figure 3.2: RePlay's status area displays tool names after they are clicked and adds them to the search field.

multiple videos and browse other results while one video plays. This ability to "see inside" multiple resources from a single page increases foraging efficiency [69, 176, 232].

Clicking the full-screen button in a video's bottom-right corner opens the video in a separate window that stays above all other windows while it is open, so that users can watch a video at a larger size when desired.

When the user switches to a new application and clicks a tool, RePlay automatically searches for the application name. This seeds the panel with application-relevant videos to give users a starting point.

3.3.2 Detecting Application Context

RePlay uses contextual information to augment the user's query and search within video results. The motivation for context-augmented search is to increase relevance, especially when

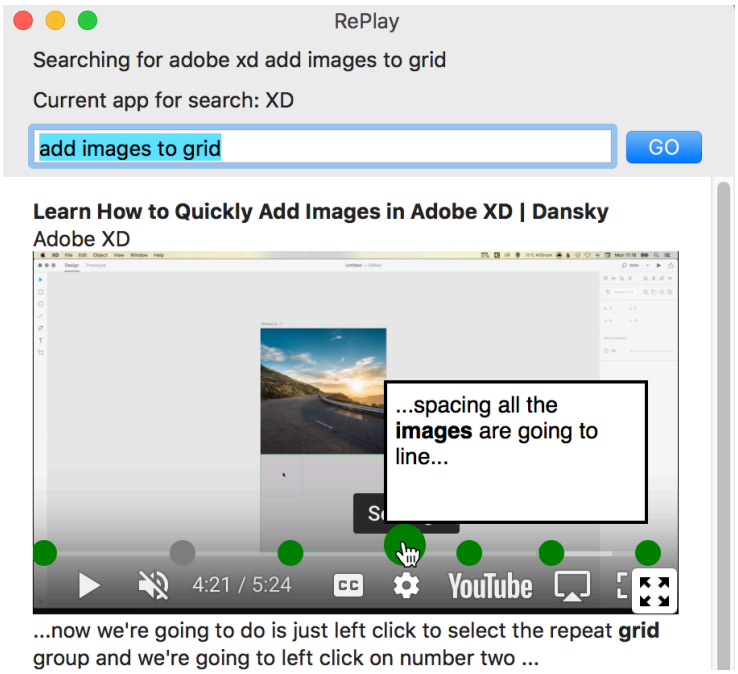

Figure 3.3: RePlay overlays green markers on the video timeline to indicate moments where the captions match the user's search term. Mousing over a marker shows a pop-up with an excerpt from the captions at that moment; clicking it plays the video from that moment.

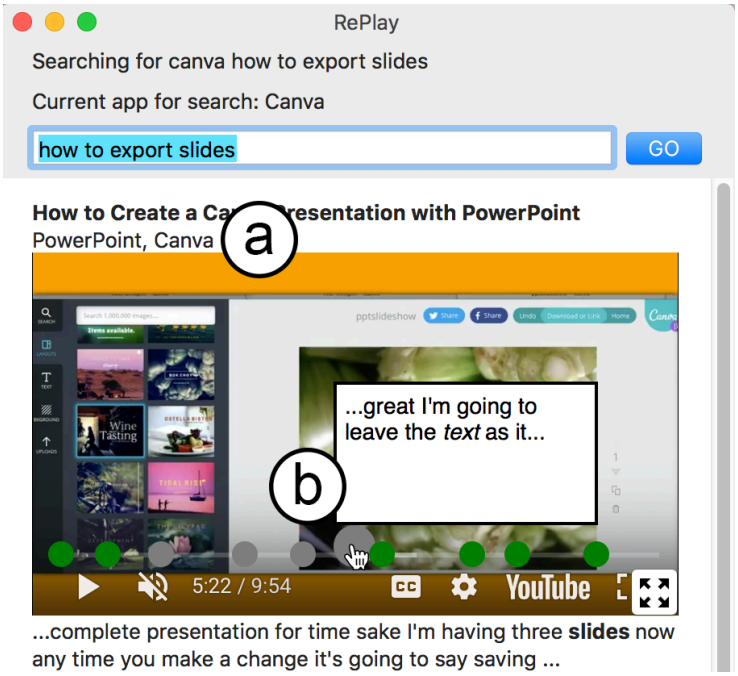

Figure 3.4: RePlay displays contextual cues based on recent application and tool use. a) RePlay lists the three most recent applications that the video mentions. b) Grey markers indicate mentions of recently-used tools. In this example, the user recently used the "text" tool in Canva. Mousing over a marker shows a caption excerpt; clicking a marker plays the video.

users don't know what to ask for. However, if not done well, adding terms has the opposite effect: excluding relevant results and/or presenting irrelevant ones [55]. We tried several heuristics with RePlay; the current implementation includes the three most recent applications and three most recently-clicked tools.

How to detect context?

RePlay leverages OS accessibility APIs to detect every click. On each click event, RePlay retrieves the name of the click's source application, the type of element clicked, and the element's accessibility description (when present). If the element is a button, checkbox, text field, slider, or menu item, RePlay stores it as a recent tool and updates the status area and search field with the tool's name. If the user switched applications since their last click, RePlay updates its status area to reflect the new application, and resets its list of recent tools.

Challenges with detecting context via accessibility

Extracting accessibility text obviates the need for hard-coded knowledge about specific applications. The challenge of this system-wide approach is that despite platform accessibility guidelines, applications vary widely in what accessibility they offer and how [97].

In MacOS, RePlay can always extract the application name and menu items. Buttons and other interface elements have accessibility labels in many applications (*e.g.,* Adobe XD, Microsoft Office, iMovie, Tableau, Sketch), but not in others (often long-existing software, *e.g.,* Photoshop). Applications differ in which accessibility fields they support and what information is in what field. For example, tool names may be in the *Title*, *Help*, *Description*, or *Value* attribute. RePlay checks all four, preferring them in that order. RePlay also gathers accessibility information for websites, as long as browser accessibility access is enabled. It is by default in Safari, and as an option in Chrome. Many sites implement accessibility labels (*e.g.,* Canva, Wordpress, Sketchpad, Snappa, G Suite). Those that do not still include some information by default (*e.g.*, text area contents via the *Value* attribute). RePlay infers website names from the URL and website title.

3.3.3 Video Search & Ranking

RePlay leverages existing online video search engines to retrieve video results. It then finds and ranks relevant clips within these videos (Figure 3.5). RePlay's architecture requires no prior understanding of the applications or videos; it relies on captions for clip extraction and ranking. Video authors often talk about things they are doing and provide tips about tools; captions thus provide narrative information beyond the visual content in the video that can be useful for learners.

Available data

RePlay's current video corpus comprises all English videos on YouTube that have a caption track. Most do: YouTube auto-generates captions by default. We used YouTube for its popularity and captions; any video search engine with an API could be used. For example, Vimeo (`vimeo.com`) and Dailymotion (`dailymotion.com`) also provide API access to videos and captions.

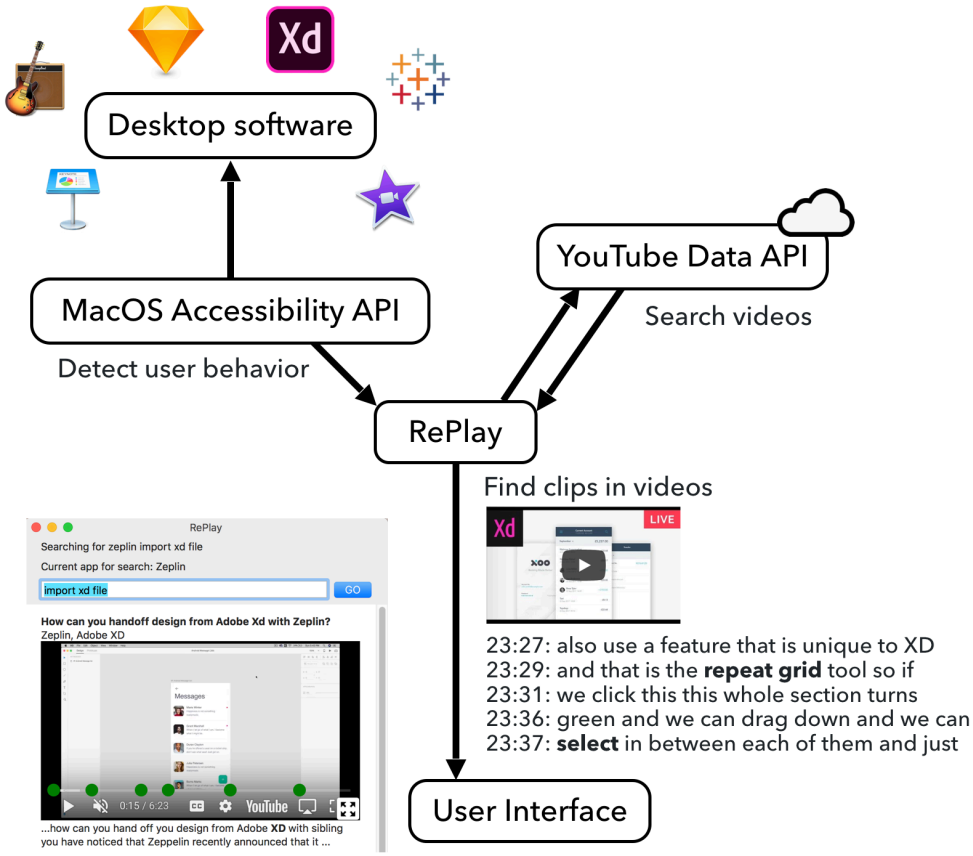

Figure 3.5: RePlay uses accessibility APIs to detect user context, which it uses to augment the user's query and to search and rank videos. RePlay finds matching clips by searching video captions for the user's query and recent tool names.

Searching

Video search requires more steps than document search, because captions are obtained separately. This two-step search means that issuing multiple queries with context terms added (such as tool usage) like prior work [50] would be too slow. To speed responses, RePlay constructs and issues a single query concatenating the current application's name with the user's query. Leaving context terms out of the query also ensures that the user-provided search terms are not "washed out". RePlay queries YouTube and selects its top ten video results that have English captions and mention the current application in any of the title, description, or captions (to avoid results that may contain other keywords but do not pertain to the current application).

Finding clips and re-ranking videos

Several techniques automatically extract instructional video clips from screencasts of software use. The dominant approach leverages application usage [40, 78, 125, 233], requiring that the video be recorded in an instrumented version of the software. Alternatively, computer vision can detect tool-selection events [150, 184], even without prior knowledge about the specific software [11]. To be application-independent and embed online videos directly without waiting to download and process them, RePlay instead uses metadata and caption text to rank and segment videos.

For each video result, RePlay divides its captions into 30-second segments, searching each for the queried keywords (with stop words removed) and names of the three most-recently used tools in the current application. It ranks all segments by the total number of keyword matches. To break ties it uses number of tool name matches. The highest-ranked segment determines the video's start time. Timeline markers denote the top ten segments: green for those with a query term; grey if only a tool is mentioned. RePlay re-orders the video results based on the total number of matching clips, and displays the top five videos in the re-ordered list. To break ties it uses the total number of matching keywords within the clips.

Although automatic captions are far from perfect, we found them to be sufficient for searching in RePlay. Captions are already an approximation of what the demonstrator is doing, so despite some errors, they work well enough for identifying potentially relevant moments. Having any aids for navigating within videos is still an improvement over standard viewers. Still, future systems could allow viewers to easily correct errors as they watch to improve caption accuracy for future viewers.

3.3.4 Implementation

RePlay is implemented as a MacOS application in Swift. In addition, a custom server is used for usage logging and displaying videos, described in further detail below. RePlay uses the MacOS Accessibility API to extract information from input events in other desktop and web applications. For both studies, RePlay used a whitelist to only detect clicks in certain applications and websites. A blacklist could be used instead; we explore this in the discussion.

RePlay generates a unique anonymous user ID for each computer it runs on. This is solely for the purposes of usage logging to analyze participants' usage behavior. The custom server includes a usage logging script that saves each usage event it receives in a `csv` file on the server. While RePlay is being used, it sends details about each interaction event the user makes to this logging script along with the user's ID.

When a search is triggered, RePlay queries the YouTube Data API's `search` method, which returns an ordered list of video IDs. For each ID, RePlay checks if English captions are available using YouTube's `get_video_info` method, which returns a string in URI parameter-value format. If captions are available, this string includes a parameter called `captionTracks`. The value of this parameter is a list of dictionaries containing information about each available caption track, including its language and a URL to the caption track in XML. RePlay selects the English caption track and follows its URL to obtain the captions. The resulting XML includes time

stamps for every 5 to 10 words[1]. RePlay stops once it has found 10 videos with English captions (or it has gone through all 50 search results).

Since MacOS applications cannot execute Javascript directly, RePlay displays each YouTube video in a Swift `WKWebView` object. To display a video, RePlay passes the `WKWebView` a URL to a webpage hosted on the custom server. The webpage takes the following query parameters as input from the URL: the video's YouTube ID, the start time at which to cue the video, the user's user ID, the index of the video in the list of displayed videos, and a stringified array containing the start times and caption excerpts for each clip selected by RePlay's ranking algorithm. The webpage uses Javasript to embed the YouTube video in an `iframe`, cue it to the given start time, and overlay timeline markers and pop-up captions for each clip. It also logs all user interactions with the video (*e.g.,* play, pause, seek) to the usage logging script.

Limitations of current implementation

A search that returns all new videos can take up to a few minutes to finish depending on network speed, due to the multiple requests needed to retrieve captions for each video. On a typical fast internet connection (~35 Mbps), the initial search request to YouTube takes about 0.5-0.75 seconds on average. For each video in the list of search results, RePlay must issue two requests in succession to obtain captions; the first takes ~0.2 seconds, and the second takes ~0.15 seconds on a fast internet connection. Therefore, in a fully optimized program the entire process could take about 1 second total. The current implementation of RePlay does issue the caption requests for different videos in parallel but only processes the responses in series. This could be better optimized by fully parallelizing the web requests and buffering results as they come in. Currently, RePlay speeds up future searches by caching all retrieved captions locally (since they are pure text, this takes up little space).

RePlay's implementation is also complicated somewhat by the need for a custom webpage

[1]For more details on this process for obtaining YouTube captions, see `https://medium.com/@cafraser/how-to-download-public-youtube-captions-in-xml-b4041a0f9352`

to display videos. If RePlay were built as a web application or as a desktop application with web technologies (*e.g.,* Electron), there would be no need for a separate webpage, as the videos could instead be integrated directly into the application. However, in order to have access to the MacOS Accessibility API, RePlay needed to be built as a MacOS application. In the future, an architecture that allows both web technology and accessibility API access could eliminate this complexity.

3.4 Study 1: How Does RePlay Support Activities in the Wild?

A week-long field study with seven participants investigated whether and how people might use contextually-augmented video search in their own work. We focused on visual design, as videos are especially helpful for visual work [40]. Participants were recruited from the local design community and a prominent creative software company (Table 3.1). Participants had mixed amounts of experience with the main design software they used during the study (either Sketch or Adobe XD); all participants were proficient in at least one other creative application. Several had recently switched to Adobe XD or Sketch from other software, or had recently started their current job. After an initial interview, we installed RePlay on each participant's computer. Participants kept RePlay open throughout the next week while they worked (one used it for 10 days). Participants returned for a followup interview and received a $45 gift card as compensation for their time.

Participants used an initial version of RePlay (Figure 3.6); based on their feedback, we revised it to the version presented in the previous section. Study 1's RePlay did not consider cross-application context; it only used context from the current application. This RePlay only monitored Adobe XD and Sketch to avoid capturing unrelated or private data from other applications. RePlay logged the following events to the custom server when it was open: all clicks on interface elements in Adobe XD and Sketch, all interactions with RePlay, and all switches to and

between other applications.

We took notes on all interviews, noting similar answers to questions and identifying common themes. This data and feedback helped motivate the final version's focus on cross-application support, helped us understand what kinds of tasks RePlay may be most useful for, and highlighted some advantages and challenges of contextual support.

3.4.1 Results

Over the week, participants reported spending between 1.5 and 30 hours on their design work (Table 3.1). Some said they kept RePlay open the entire time; others closed it at times to focus on their work.

Generally, participants appreciated having help readily available. As *P2* explained, *"this gives me an interface where I can search and do everything and it's automatically there right next to the design. There are no extra steps."* *P4* enjoyed seeing RePlay react to her actions: *"it felt like I had a buddy."* Only one (*P6*, the expert) did not use it at all: he was fluent in the software and did not seek assistance.

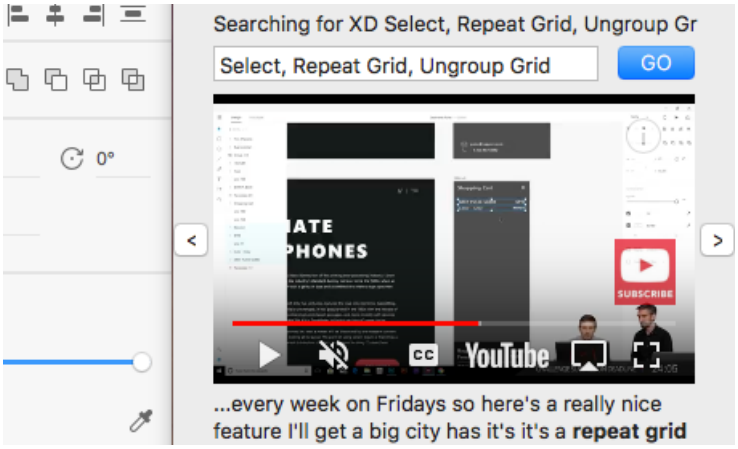

Figure 3.6: Study 1 used this initial version of RePlay, shown here next to Adobe XD. It did not show video titles or timeline markers; instead it had arrow buttons next to each video to skip between that video's ranked clips. It also added the 3 most recent tools to the search field instead of one.

Table 3.1: Study 1 participant background and usage. Participants self-reported their experience, design work done, hours spent designing, and % of that time with RePlay open. # queries and # videos watched were calculated from RePlay's usage data.

	Job title	Main app	Experience w/ app	Design work done	Hours designing	Time w/ RePlay open	# queries	# videos watched
P1	Sr. Product Manager	Adobe XD	Beginner	Style guide and wireframing	20	100%	3	4
P2	Freelance Designer	Adobe XD	Beginner	Wireframing / prototype design	4.5	100%	10	5
P3	Freelance Designer	Sketch	Beginner	Trying to learn Sketch	1.5	100%	4	2
P4	PhD Student	Sketch	Intermediate	Screen design and grid customizing	5	40%	10	0
P5	UX Designer	Sketch	Beginner	Creating templates and logos	30	70%	24	8
P6	Sr. UX Designer	Adobe XD	Expert	UX workflows	20	25%	0	0
P7	Sr. UX Designer	Adobe XD	Intermediate	Wireframing an app UI	12	33%	19	13

Contextual clip search was most useful for specific tasks

Four participants said they tended to use RePlay when stuck trying to figure out how to do something specific. Three also said they used it to find out what a particular tool could do or how to use it. All but one participant worked on targeted tasks (see Table 3.1). *P3* wanted general resources for getting started with Sketch, and did not find RePlay helpful.

Three users recounted similar stories of searching for a particular question and quickly finding a clip within a video that answered it. *P1* described searching for "Make Symbol" after clicking on the "Make Symbol" tool in Adobe XD. The answer he needed came from an auto-selected moment near the end of a 3-hour video. *P1* added, *"if I had searched for that myself I would've given up."* Similarly, *P5* found what he needed in a 1.5-hour video that was cued to a moment 20 minutes in. He was trying to create margins, searched for "layout grid", and *"the first video in the list showed me exactly what I needed to do"*, which was to check a box he hadn't noticed. *P4* found RePlay useful for indicating that her specific goal was *not* possible: she wanted to customize a grid system in Sketch and searched for "grid settings", but the caption excerpts indicated that none of the results mentioned customizing grids.

Contextual video clips sometimes invited opportunistic learning: *P1* and *P7* recounted instances where a video they were watching taught them something they didn't know and hadn't thought to look for. *P1* continued watching the "make symbols" video as it described grouping and layering symbols. This part *"wasn't originally what I was searching for but it was exactly what*

I needed ... [I] gained a lot more knowledge." P7 described how he *"searched for 'character styles' and actually found new information that I've never seen before"* about the Libraries panel.

Tool context helps, but not in the search query

All participants who searched with RePlay preferred deleting tool names from the search field and instead typing their own. Five participants mentioned that including three tools in the query was too many, as it made the query too general. *P3* said the automatic query seemed *"stuck on the last thing I did, which might not be relevant to what I'm thinking now."* Both *P4* and *P5* said tool *names* were not as useful because they wanted to search for an *action*, not its constituent tools (*e.g.*, "rotate object" or "export nested artboard"). Higher-level activity inference may provide more useful assistance.

Two participants said they liked that RePlay populated the query with tool names because *"coming up with the right search terms is really hard and I don't know the names of the tools [...] so it's nice not to have to think of them"* (*P4*). Even *P6*, the expert, said *"I know how to use everything but if you asked me the names of the tools, I have no idea."* Participants appreciated that RePlay added the application name to the query, as it allowed them to *"spend more brain power on the details of [the query]" (P2)*.

Participants frequently switched between applications

RePlay logged every time users switched between *any* applications when it was running. Excluding system applications (such as Dock, Finder, and System Preferences), participants used an average of 17 different applications while RePlay was running ($SD = 8$), and switched between applications a mean of every 6.6 minutes ($SD = 8.5$min). If we exclude all continuous periods of over 3 hours (assuming that these sessions were breaks of some sort), this average lowers to every 2.5 minutes ($SD = 1.8$min). This diverse application usage supports our motivation for improving cross-application workflows.

Improvements to RePlay

Because participants thought that automatically including three tools was too many, RePlay now includes only the most recent one. Four participants also mentioned that more general information like video titles would be helpful, as caption excerpts often seemed *"out of context"* or *"snatched out of middle of a sentence"* (*P2*). Consequently, RePlay now includes video titles. Four participants wished they could watch videos at a larger size; RePlay now includes a resizable video player window. Two participants also mentioned that some video results did not pertain to the current application; RePlay now excludes results that do not mention the current application in the metadata or captions.

3.5 Study 2: How Does RePlay Compare to Current Methods for Video Assistance?

A between-subjects study ($n=24$) investigated how contextual assistance for multi-application activities might affect behavior compared to standard web video search. It found that contextual assistance can reduce time spent searching and navigating videos.

3.5.1 Study Procedure

Participants were asked to imagine that they were designers working for a client developing a travel journal mobile app. To replicate a multi-application activity, the study task spanned three applications: Canva (`canva.com`), Adobe XD, and Microsoft PowerPoint. Participants were asked to improve and change an initial design for their client (Figure 3.7). The task had both creative and technical requirements. Creative requirements included making the logo more travel-themed, adding visual appeal to the prototype welcome screen, and making a PowerPoint presentation to show to the client. Technical requirements included adding additional photos

to a grid, rounding their corners, and recording a video walkthrough. If participants needed help, they were instructed to search for tutorial videos using either RePlay or YouTube in a web browser. RePlay participants were introduced to its features prior to the task. To ensure that all participants had access to the same resources, they were asked not to use other search engines or resources. Participants had 45 minutes to complete the task, and answered questions about their experience and help-seeking at the end. Participants were compensated with a $15 gift card.

Participants

Twenty four participants (14 female) were recruited through online and paper advertisements on a university campus. Prior to the study, participants filled out a survey about their design experience. Participants were randomly assigned to either the RePlay ($n = 12$) or Web ($n = 12$) condition and counterbalanced based on experience. "Novice" was defined as completing at most two courses in design and reporting experience with at most two design applications. "Experienced" was defined as completing more than two courses in design or reporting experience with more than two design applications. Six of the 24 participants were "experienced".

Participants rated their pre-study familiarity with each of the three study applications on a 5-point Likert Scale. Participants were generally familiar with Powerpoint ($mean = 4.2$), and unfamiliar with Canva ($mean = 1.5$) and Adobe XD ($mean = 1.2$).

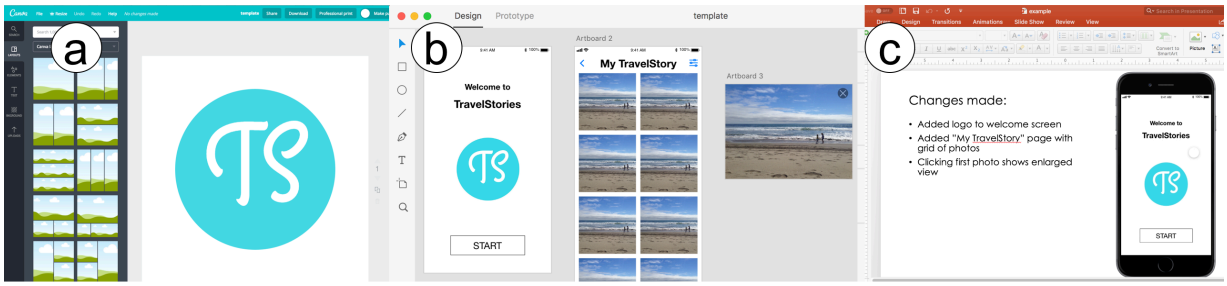

Figure 3.7: Study 2 asked participants to make changes to an initial logo in Canva (a), update a prototype in Adobe XD (b), and make a presentation showing their changes in PowerPoint (c).

Measures

We were primarily interested in participants' search behavior. We measured the number of queries, their length, and time spent in the search interface. Qualitative measures included how participants determined which videos to watch and their navigation strategies (observed through screen recordings). We also gathered participant feedback in the RePlay condition on its features.

3.5.2 Results

RePlay participants averaged 3.3 queries each (33 total); Web participants averaged 3 queries each (24 total) ($x^2 = 1.42$, $df = 1$, $p = .23$). Web participants typed longer queries (*mean* = 4.33 words, *SD* = 1.41) than RePlay participants (*mean* = 2.53 words, *SD* = 0.59)($t = 2.52$, $df = 15.98$, $p < .01$), because Web participants often manually added the application name, whereas RePlay auto-included it. 72% of search queries were for Adobe XD and 28% were for Canva; none were for PowerPoint since participants were more familiar with it. Four Web and two RePlay participants did not search for any assistance; we cover this in the Discussion.

Participants varied considerably in the amount of time and effort they spent. This is a challenge with a task that has both creative and technical components; people prioritize these components differently. Many participants spent a long time perfecting their design and had to be cut off after 45 minutes; others did the minimum required and finished in as few as 23 minutes. Because the task was open-ended, we did not compare completion times across participants. Our analysis focuses on search behavior.

Contextual assistance lessens time away from task

Web participants spent nearly twice as long in the search interface (*mean* = 214.5 seconds, *SD* = 127.1) as RePlay ones (*mean* = 116.2 seconds, *SD* = 58.9). Due to the high variance and small sample size, the difference was marginally significant ($t = 2.02$, $df = 9.4$, $p = .07$)

(Figure 3.8 left). When the time each participant spent in search is averaged by the number of queries they made, the difference is significant: RePlay participants spent about 40% less time in search per query ($mean = 42.83$ seconds, $SD = 29.5$) than Web participants ($mean = 72.62$ seconds, $SD = 22.4$)($t = 2.52$, $df = 15.28$, $p = .02$) (Figure 3.8 right). Though RePlay pre-cached many video captions beforehand, we could not predict all queries users would make. Thus, some query responses in RePlay took up to two minutes. Because this latency could be reduced, we subtracted loading times in both conditions from the total time in search.

Easier navigation both within and between video results may explain why RePlay participants spent less time in the search interface per query. Web participants used various strategies for navigating within videos, including keyboard shortcuts to fast-forward and rewind, increasing video speed, and hovering over the timeline. Navigating between different video results in YouTube required selecting a video, watching or skipping through it to determine whether it was relevant, and if not, going back to the results page and selecting another. RePlay's timeline

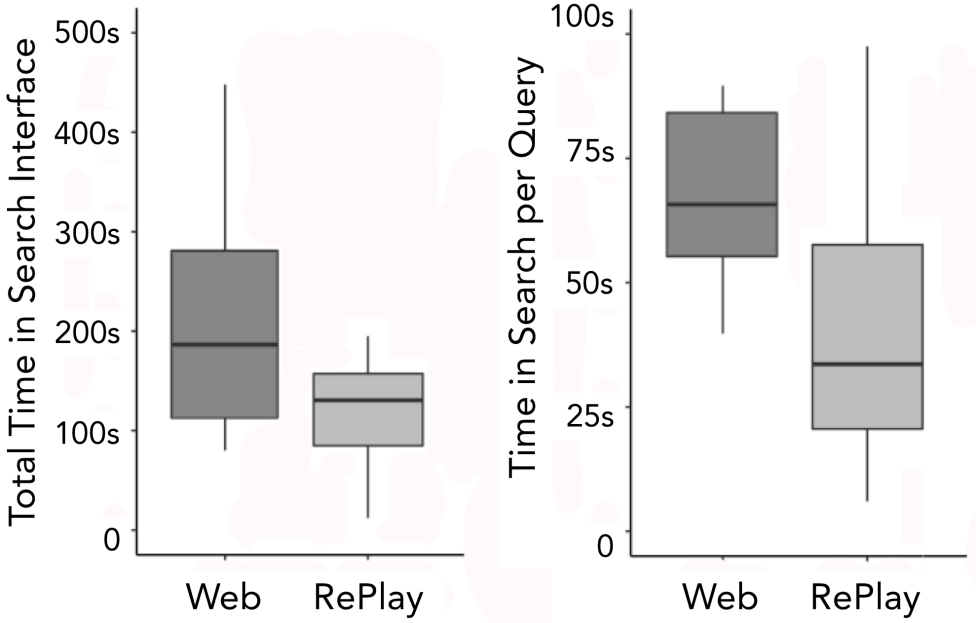

Figure 3.8: RePlay participants spent marginally less time overall in the search interface than Web participants ($p = .07$, left) and significantly less time *per query* ($p = .02$, right).

Table 3.2: Average helpfulness ratings for RePlay's contextual features ($n = 12$). Most participants found the timeline markers most helpful for determining which videos to watch.

RePlay Feature	Title	Thumbnail	Caption	Timeline
Helpfulness	2.8	3.1	2.5	4.2

markers replaced many of these strategies, increasing efficiency: participants could first examine the timeline markers before deciding if and at what point to watch a video result. Eight RePlay participants hovered over timeline markers to read the caption previews, often hovering over multiple points before deciding where to watch. RePlay's panel interface also enabled participants to simultaeously play one video and examine others. We observed three participants do this, likely to decide whether another video was better without giving up on their first choice. One such participant said they wished for better visual cues of video relevance to help decide which to watch. Other participants browsed videos one at a time, perhaps to focus on the playing video.

Search queries were action-oriented, not tool-oriented

As in Study 1, participants removed tool names from queries. Instead, they wrote action-oriented queries: the most common were "crop photos", "round corners", and "record video." Despite RePlay adding the current tool name to the search field, in all instances but one, participants deleted the tool name from their query.

Intra-video context is most helpful

In interviews, participants rated timeline markers as the most helpful (4.2/5) (Table 3.2). RePlay participants hovered over timeline markers a total of 69 times and clicked on markers 22 times. One participant mentioned that timeline markers provide a *"scaffold of what to look for and where to start watching."* Participants rated caption excerpt and video title as less helpful. A few participants mentioned ignoring titles and excerpts in favor of timeline markers and video thumbnails. Twice in each condition, participants selected the first video result even though the title mentioned the wrong application (Photoshop or Illustrator). This suggests that the video

region is a strong magnet for people's attention.

3.6 Discussion & Future Work

Observations and feedback from the two studies suggest several opportunities for future work.

3.6.1 How Can Contextual Assistance aid Exploration?

Study 2 participants often preferred manually exploring when it would have been more efficient to search for help. The study's subjective, open-ended task may have encouraged such exploration. Only one participant used the best method to update the photo grid - using Adobe XD's repeat grid feature to update all photos simultaneously - after learning about it in a video. Other participants used various less-efficient methods (*e.g.*, un-grouping the repeat grid and manually adding and re-sizing photos). Because these methods achieved their desired goal, participants may not have thought to investigate whether a faster option existed.

Interface exploration and browser search each have shortcomings. Exploration is a common problem-solving strategy because it can be enjoyable, it builds on domain knowledge, and each individual action is low cost [123, 191]. However, for novices especially, exploration is cumulatively slow, its learned knowledge is hard to integrate, and it induces a high cognitive load [123, 229]. A limitation of learning unfamiliar domains via exploration is that people may settle for sub-optimal methods or strategies because they are unaware of a better alternative. Moving from an application to a web search can yield better results, but has a higher initial cost as it lurches people out of their task. We believe that contextual search has the potential to offer the benefits of both approaches without either of the drawbacks.

Six participants in Study 2 did not search at all; most felt they could figure things out via exploration. One participant said they *"felt like I could find it if I searched for it [in the interface],*

which I did. I was able to figure it out." Two other participants mentioned that they felt searching for help would be more time-consuming than trial-and-error exploration. One stated that he wanted to search for help, but *"I knew I didn't have time. I wanted to complete the task so I just hacked it."* If Study 2's result that contextual search reduces search time holds, these perceptions may change over time. RePlay's occasionally-slow loading times may have also affected this perception; prior work shows that a difference in latency of search results as small as 300ms can discourage people from searching [25].

How might contextual assistance encourage productive exploration while providing intervention when needed? While proactive support can be beneficial, a challenge is providing assistance without being too disruptive [150]. To minimize distractions, the RePlay interface mostly changes only in direct response to user input. However, novices may not even realize they need help, making proactive suggestions more valuable. For example, RePlay could automatically refresh video results when the user seems stuck, or suggest relevant queries when the user begins to search. Future work should examine how these alternatives might change people's behavior and workflows over time.

3.6.2 What and How Much Context to Include?

While logging recent tools can help suggest next steps [153], we found that using recent tools explicitly in search queries is not useful. Participants in both studies did not use tool context in their queries, preferring action-oriented queries instead. Usage history is by definition retrospective (*i.e.*, it describes what the user has already done). In contrast, search is often prospective (*i.e.*, looking for something the user hasn't done yet). Tool context may only be helpful closer to where users interact with tools (*e.g.,* as part of tooltips [77]).

Interestingly, people used the same terms and concepts across different applications. Study 2 participants searched "crop photos" for both Canva and Adobe XD despite neither application having an explicit crop tool. This highlights both a challenge and an opportunity: people bring

mental models that may not carry over into different applications. Study 1 suggested that displaying tool names may help people learn app-specific terminology. However, for both knowledge and speed reasons, users sometimes omit valuable terms. RePlay's interface benefits are reduced when users don't search using the same terms as videos. A natural language mapping [5] between video captions (along with other natural language data like comments and tags) and the tools they mention may increase captions' value for search.

Finally, what other contextual information besides tool use might be helpful for identifying relevant videos? Liu *et al.* [139] showed that in addition to command logs, information about the use of layers and the time between interaction events can help segment usage logs from image editing tasks into subtasks. Similarly, information such as the name of the user's active layer might be helpful as additional context for a search query. Liu *et al.* also found that the visual change and location of the user's edits were less useful for segmentation, but perhaps the visual content of the user's canvas or document could be used to identify videos with similar content. Future work should explore how we might collect and use such types of contextual information to search for help in an application-general way.

3.6.3 What Design Challenges Remain?

Many design decisions were motivated by our focus on cross-application workflows; *e.g.,* showing results in a separate window in a consistent location. Showing all results together made it easy to browse multiple resources at once. However, some participants still preferred focusing on one video at a time. Future work should consider how different layouts influence browsing behaviors and which behaviors lead to more effective workflows.

Currently, users must explicitly whitelist an application for RePlay to capture its events. While this approach offers more privacy, it also adds burden for users. Blacklisting, on the other hand, would allow RePlay to respond to all applications except for explicitly-omitted potentially-sensitive ones (*e.g.,* Messages), offering broader benefits. For our initial studies, we chose the

greater privacy of a whitelist. For real use, users could choose whitelist or blacklist, and/or RePlay could request approval for each new application, similar to websites that ask for a user's location.

3.6.4 What Other Domains Might Benefit?

RePlay's main insight is that given a source of user context, we can search, curate, and index into resources from a large corpus. RePlay demonstrates this approach using video; different activities (*e.g.,* programming) may benefit from other types of content (*e.g.,* text resources). RePlay could naturally be extended to any textual resource (or resource with textual metadata). Text results could be displayed as short summaries with clickable keywords to expand more detail [50]. For detecting context, RePlay used MacOS's accessibility API; other OSs (*e.g.* Windows [152]) also have similar APIs. Beyond software, RePlay's approach could extend to any domain for which online videos are abundant (*e.g.,* physical building tasks). To detect activity context, one could augment physical tools with sensors [145,200] or track body poses with wearable sensors or computer vision. A challenge for future work is to convert sensor or vision data into text searches, or to index videos using the sensor or vision data directly.

3.7 Conclusion

This chapter introduced an application-independent approach for contextually presenting videos and a demonstration of this approach in the RePlay system. RePlay shows how system accessibility features and video captions can be used to detect context and search within videos in a flexible, domain-general way. Like curb cuts or closed captioning [194], RePlay demonstrates how accessibility features can provide universally-beneficial assistance. Expanding accessibility and increasing cross-application consistency through guidelines and enforcement would benefit everyone. It would also expand application tailoring, integration, and assistance with systems like RePlay. Two studies demonstrated that cross-application contextual video assistance helps

users spend more time on their task and less time searching for help. We also observed how contextual assistance can sometimes be at odds with peoples' desire to explore and figure things out via trial-and-error, due to the perceived time and attention switching costs of searching for help. The next chapter builds on RePlay to explore how we might further reduce these costs to searching for help. This work brings us one step closer to leveraging the wisdom of the web for personalized, just-in-time learning.

3.8 Acknowledgments

We are thankful to Michelle Lee for her assistance in conducting Study 2, and to all study participants for their time and feedback. This work was supported in part by NSERC and Adobe Research.

This chapter, in part, includes portions of material as it appears in *RePlay: Contextually Presenting Learning Videos Across Software Applications* by C. Ailie Fraser, Tricia J. Ngoon, Mira Dontcheva, and Scott Klemmer in the Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19). The dissertation author was the primary investigator and author of this paper.

Chapter 4

ReMap: Multimodal Help-Seeking

People are reluctant to search for help when they feel it will take more time than trial-and-error exploration, even with a contextual search system like RePlay. However, trial-and-error often takes a long time or does not result in a correct solution. Given this cognitive bias, how might we lower the barrier to searching for help? This chapter aims to make searching easier and faster through multimodal interaction. We introduce *ReMap*, an extension to the RePlay system that allows users to speak search queries, adding application-specific terms deictically. Users can navigate ReMap's search results via speech or mouse. A lab study showed that ReMap helped people stay focused on their task while simultaneously searching for and using help resources. Users' experiences with ReMap also raised a number of important challenges with incorporating multimodal features into a help-seeking system.

4.1 Introduction: Making Search Feel More Natural

Despite the ubiquity of online search, help-seeking remains a tedious and difficult process for many, because people are often reluctant to take their hands off their current task to search for help when they need it. Although RePlay's contextual search helped people spend less time help-seeking once they decided to search, many were still discouraged from searching in the first place. Creating good search queries can be prohibitively difficult for novices, who may not know the domain vocabulary [195]. Searching also switches the user's attention from the task at hand to the task of articulating their goal in a way that will match the resources they seek [229]. How

might we lower the barrier to searching and make it easier for people to find the resources they need in the moment?

When people help each other, they use language, gestures, and shared context to communicate. In contrast, finding help through search engines is still primarily text-based. This chapter explores how multimodal interaction might make searching easier, by enabling people to articulate their thoughts like they would with each other. Different types of information lend themselves better to different modalities; leveraging the strengths of multiple modalities and integrating them smoothly can be extremely effective for improving communication [171].

We introduce ReMap (Figure 4.1), a multimodal interface that enables users to search for help videos using speech and pointing, without taking their hands (or mouse) off their current task. ReMap builds on RePlay (Chapter 3), which uses relevant context from the user's activities to improve the relevance of search results. ReMap's multimodal help search demonstrates three main design insights:

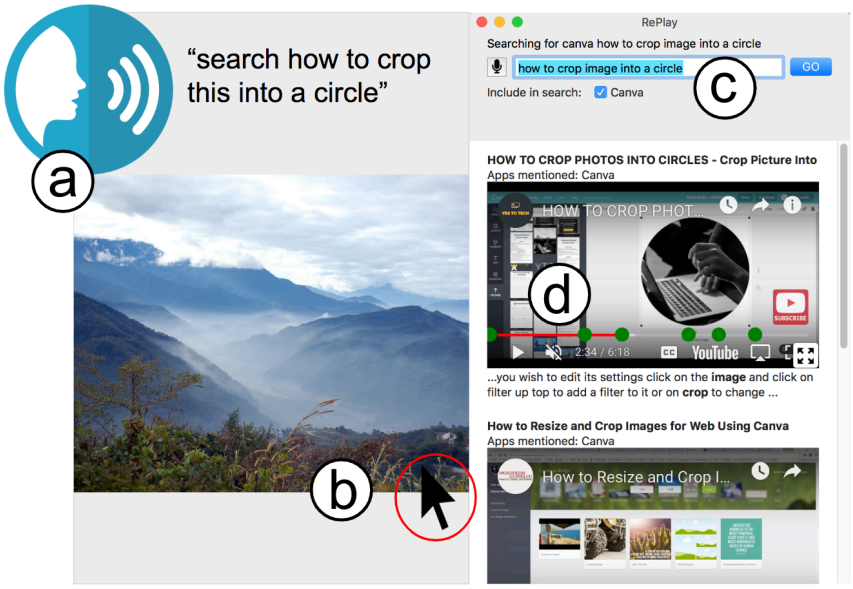

Figure 4.1: ReMap is a multimodal search interface for finding learning videos. a) The user speaks their query. b) The user clicks on an image on the canvas while saying the word "this." c) ReMap automatically changes the word "this" to "image." d) ReMap highlights relevant moments by placing markers on the timeline of each video result.

1. To avoid context-switching, users can initiate and dictate a search using speech.

2. To avoid needing to learn or recall application-specific terminology, users can point at relevant interface elements to include their names in the search query.

3. To simultaneously work on their task while following along with a tutorial, users can play, pause, and navigate video results using speech.

A study with 13 participants found that ReMap allows people to stay focused on their task while help-seeking. The study, along with iterative prototyping and pilot testing, also raised a number of important challenges with incorporating multimodal features into a help-seeking system. We conclude this chapter by discussing ways future work can integrate multimodal search even more smoothly into creative tasks, further lowering the barrier to help-seeking.

4.2 Related Work: Multimodal Interaction

4.2.1 The Power of Speech

In 2016, Google reported that 20% of all search queries on mobile Android devices were spoken [182]; that number has likely grown. Speech input is especially appealing on mobile devices, as people often use them while on the go and busy with other tasks [81]; but even on desktop computers, voice assistants are often used for web search [157]. Speaking a search query is often easier and faster than typing it out, and it allows the user to ask a question in the way they might ask a friend; mobile voice queries tend to be closer to natural language and more often phrased as questions than text queries [81]. Also, people tend to search by voice more when seeking audio or video results [81], supporting ReMap's approach of using speech to search for videos. Finally, speech may be more useful for users with specific goals: Laput *et al.* [127] found that when editing photos, speech was most useful when people knew exactly what they wanted to do. When people didn't know what they wanted, browsing a gallery of examples was more

helpful as it allowed them to compare visual previews of potential effects before choosing one. Therefore, ReMap's main intended use case is for situations where users have targeted questions in the middle of a task, as RePlay was found to be most helpful in such cases and speech is likely to be especially beneficial. In other words, ReMap supports users seeking procedural help for achieving a specific outcome.

4.2.2 Combining Modalities can Maximize Cognitive Abilities

Combining input from multiple modalities (*e.g.*, speech, gesture, touch) can reduce cognitive load for complex tasks [172,188], make tedious tasks more efficient [96,105,171,197], reduce errors [171], increase precision [44], and even make tasks more enjoyable [127, 171]. Such combinations work best when they maximize users' working memory by using different modalities for different types of information [103, 172, 188, 218]. For example, telling the user about keyboard shortcuts for the operations they execute via auditory feedback increases the likelihood that users will retain them, as users' auditory working memory is otherwise unused when working in software [76]. Similarly, using speech to navigate tutorial videos while one's hands are busy with a physical task allows users to process both the video and the task simultaneously [35]. ReMap similarly partitions different modalities between the user's creative task and ReMap's help-seeking interface (Figure 4.2), allowing users to speak search queries and video navigation commands.

Pointing at objects and spatial locations is often easier and more natural than describing them in words only [19, 128, 188]. Similarly, using a screenshot of an interface element as a search query is easier than describing it in text [239]. When combined with speech, pointing allows people to communicate more precisely by using deictic terms (*e.g.,* "this", "here") to refer to objects and spatial locations while pointing at them [19, 127]. ReMap similarly allows users to point at interface elements and canvas objects in their creative software and reference them deictically while speaking a search query.

Multimodal input can be especially beneficial for creative tasks, where maintaining a flow

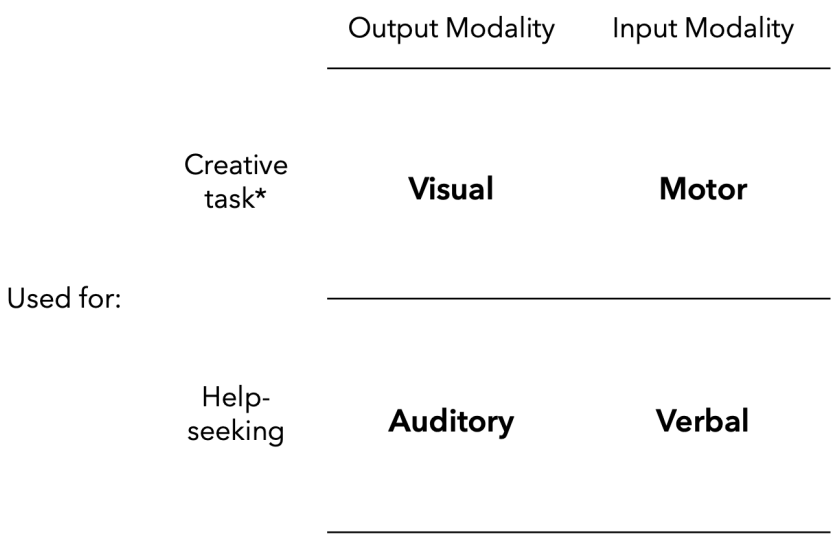

Figure 4.2: ReMap partitions the above modalities between the user's creative task and their help-seeking task to maximize cognitive abilities. Users primarily use speech and listening to search for and navigate help videos while keeping their hands and eyes on their creative task. The * indicates that the visual and motor modalities are not *exclusively* used for the creative task; users can also transfer their visual and motor attention to the help resources when necessary, *e.g.*, to watch a video or type a search query manually.

state is important. Using different modalities for software-related actions allows users to keep their focus and hands on their creative work. For example, using speech to access tools instead of menus or keyboard shortcuts can help artists maintain creative flow and stay focused on their task [113, 201]. While doing digital painting, an artist could switch from the brush tool to the eraser by saying "eraser" instead of switching their attention and moving their hands to the toolbar or keyboard [113]. Not surprisingly, much prior work exploring multimodal input has centered around creative tasks such as photo editing [127], graphic design [113], drawing [175, 201], data visualization [204], and 3D modeling [205]. However, most of this prior work has used speech to execute commands, rather than issue search queries. This requires either defining a list of acceptable commands (which users must then memorize) or parsing natural language commands (which is prone to errors). This chapter combines insights from multimodal creative systems and voice search systems to explore how using voice to search might be useful in creative software.

4.3 ReMap System Design and Implementation

ReMap (Figure 4.1) extends the RePlay contextual search system (Chapter 3) . RePlay enables users to search for learning videos in context while working in software, and it highlights relevant moments in video results based on the user's context and query. While RePlay helps people find results faster, the attentional cost of switching to RePlay discouraged participants from using it more frequently. ReMap lowers the switching cost and cognitive load of help-seeking by introducing three main improvements over RePlay: searching for help using speech, making deictic references in a search query, and navigating video results using speech commands.

4.3.1 Searching for Help using Speech

At any time, the user can search for help by saying the word *"search"* followed by their query (Figure 4.1a). Much like other speech interfaces (*e.g.,* Siri), ReMap's search field displays the user's query as it is being transcribed. Once the user is finished speaking, ReMap executes a search with the transcribed query. This allows the user to quickly and easily search for help without taking their hands (or mouse) off their current task.

4.3.2 Making Deictic References in a Search Query

Especially with new software, people are often unfamiliar with an application's vocabulary but can point at application elements that are relevant to their goal. To alleviate the challenge of remembering application-specific terms, ReMap allows users to deictically reference interface elements and objects while speaking their query. If the user says *"this"* or *"that"* while clicking on a detectable element, ReMap replaces the pronoun with the reference element's name (Figure 4.1b-c). Like RePlay, ReMap uses OS accessibility APIs to resolve element names.

While ReMap is detecting a speech query, it stores a list of every detectable element clicked. Once the user is finished speaking, ReMap replaces all occurrences of *"this"* and *"that"*

with the element names in the order they were clicked before issuing the search query. This means that users need not speak the deictic reference at the exact same time as they click (which people rarely do [171]), they only need to click before they are finished speaking.

4.3.3 Navigating Video Results Using Speech Commands

In addition to dictating search queries, ReMap allows users to navigate video results using speech commands. This was inspired by Chang *et al.*'s [35] findings that people often pause and skip forward or backward when following along with video tutorials so that they can keep pace with the video, skip irrelevant content, or replay sections they need to see again. Chang *et al.* [35] propose using speech to navigate tutorials for physical tasks so that users do not have to take their hands off the task to navigate videos. Although ReMap is currently intended for digital tasks, not physical, our evaluations of RePlay found that there is still a significant cost to moving one's mouse and attention to a video from a software task, so we expect speech navigation to be similarly helpful for digital tasks.

ReMap currently supports the following speech commands: *"play"* (plays the first or most-recently played video), *"play {first, second, third, fourth, fifth} video"*, *"play {next, previous, last}"* (plays the next/previous/last video in the list), *"{next, previous, repeat} marker"* (skips to the next or previous timeline marker, or re-starts from the current marker), and *"pause"* / *"stop"* (pauses the currently playing video).

4.3.4 Implementation

ReMap is implemented as a MacOS Swift application (Figure 4.3a). Like RePlay, it also has a custom web server that hosts a script for logging usage data and a webpage for displaying YouTube videos. In addition, ReMap's custom server hosts a webpage for detecting speech. ReMap uses socket.io to communicate between the web server and the three client interfaces (the ReMap desktop application, speech webpage, and video player webpages). The web server

(Figure 4.3b) is implemented in Node.js. The speech webpage (Figure 4.3c) uses the Web Speech API to detect and transcribe speech, and the video player webpage (Figure 4.3d) uses the YouTube Player API to load and control videos (like RePlay).

When launched, ReMap opens the speech webpage in the user's web browser. To display videos, ReMap loads the video player webpage in `WKWebView` objects. Like RePlay, ReMap includes the user's anonymous ID as a query parameter for both types of webpages. For video player pages, it also includes that video's index in the result list. The server keeps track of these values for each client page that opens a socket connection, so that it can pass the commands it receives to the appropriate client.

The speech webpage invokes continuous listening using the Web Speech API, so that the user can interact using speech at any time. The Web Speech API automatically determines when the user starts and finishes speaking, returning each phrase separately. If a phrase begins

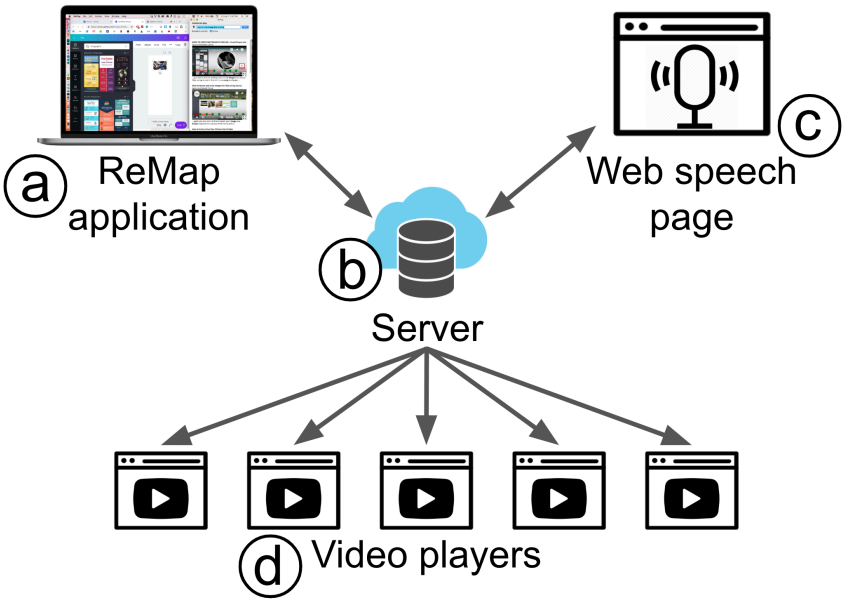

Figure 4.3: The ReMap system architecture. a) ReMap is a MacOS application that uses the Accessibility API to detect user context. b) ReMap connects to a custom web server, which hosts c) a webpage for speech recognition and d) a video player webpage embedded in ReMap for each video result.

with the word *"search"*, the webpage sends the rest of the phrase to the server which sends it to ReMap as a query. As the user continues to speak their query, the speech webpage sends the server the updated phrase every time a new word is detected, so ReMap can display it in real time (Figure 4.1c).

If the phrase does not start with *"search"* and it matches a video navigation command, the server sends this command to ReMap. ReMap then determines which video index the command applies to, and tells the server which video player client to send the command to. For example, if the third video in ReMap's result list is playing and the user says *"pause"*, the web speech page passes this command to the server which passes it to ReMap. ReMap keeps track of which video is playing, so it then tells the web server to pause the video at index 2. The web server sends a *"pause"* command to the video player client with index 2 that matches the given user ID. The video player receives this command and uses JavaScript to pause the YouTube video. If the command is to play a different video than the currently playing video, ReMap also tells the server to pause the currently playing video.

Why use a web server?

Like RePlay, ReMap uses a webpage to display videos because there is no way of incorporating web technologies like JavaScript directly into a MacOS application. It was necessary to build ReMap as a MacOS application so it could have access to the MacOS Accessibility API.

ReMap uses the Web Speech API to transcribe speech because there is no library for converting arbitrary speech to text in MacOS applications. The only available speech service for MacOS at the time of writing is command recognition (recognizing spoken commands from a pre-defined list). To enable searching by speech, we needed a speech-to-text service that could transcribe arbitrary words. To enable this, we used the Web Speech API, which requires the user to have a browser window open and relies on a steady internet connection for ReMap and the speech client to communicate. We initially used the MacOS command recognition service

(`NSSpeechRecognizer`) for detecting video navigation commands, but found it had greater latency than our Web Speech API approach.

The current implementation works well enough as a research prototype; it can detect and transcribe speech with no noticeable delay on an average internet connection. The speech webpage can be minimized or hidden by the user, and it only asks the user to grant access to the microphone the first time it is opened; all subsequent uses grant access automatically. However, to support a large number of users and avoid needing to open a webpage, a commercial system could instead implement custom in-house speech detection. Further, we hope that in the future, an architecture that allows both web technology and accessibility API access might eliminate the need for a web server entirely, thus simplifying ReMap's implementation substantially.

4.4 Study: Using ReMap for a Graphic Design Task

To gain an initial understanding of how people use multimodal search for help, we conducted a think-aloud lab study with thirteen participants. Participants re-created a given design in Canva (`canva.com`) and used ReMap to search for help when necessary. Overall we found that despite some usability and implementation challenges, multimodal video search was helpful, allowing participants to stay focused on their task while simultaneously searching for help and navigating video resources.

4.4.1 Participants

Thirteen participants were recruited from mailing lists and flyers at a university. 10/13 participants had at least some experience using voice assistants (*e.g.*, Siri, Amazon Echo) (*mean* = 2.3/5, 1 = never used, 5 = use every day). 6/15 participants had never used Canva before, and only one was very familiar with it (*mean* = 1.8/5, 1 = never used, 5 = very familiar).

4.4.2 Procedure

Participants were given two images of an infographic design (Figure 4.4) and were asked to choose one and re-create it as accurately as possible in Canva without using any of Canva's built-in templates. Participants were asked to use only ReMap (no web search) when they needed to search for help. Participants were given a brief tutorial on how to use ReMap, and were asked to try three example speech commands to ensure they understood how it worked. They were encouraged to use speech as much as possible, but also had the option to type their search queries into ReMap's search field and navigate videos using the mouse. Participants were asked to think out loud while they worked, and the experimenters captured audio and screen recordings. Participants were scheduled for 2-hour slots and were told to take as much time as they needed. Once they announced they were done (or if 1 hour and 45 minutes had passed), participants were asked a series of interview questions including prompts for Likert-scale responses to understand how they felt the task went, and whether they found ReMap helpful. Participants received a $30 USD gift card for their time.

Both infographics were designed to require several operations that are not straightforward in Canva, to increase the likelihood that participants would have to search for help. These included: cropping an image in a circle, masking an image with text, creating a bar chart given a spreadsheet of data, making a swish shape behind text (Figure 4.4 left only) and adding a shadow to text (Figure 4.4 right only).

ReMap's speech recognition requires a clear signal of the user's speech, mostly free of interference from sound output (*i.e.* from videos) or background conversations. We have found commodity headsets to be sufficient, high-quality headsets to be optimal, and built-in laptop microphones insufficient. For the study, participants wore a high-quality headset.

Figure 4.4: Study participants chose one of the above two infographic designs to re-create in Canva, using ReMap to search for help.

4.4.3 Results: Multimodal Search Enables Multitasking

Overview of Search Behaviour

Participants issued a total of 118 intentional search queries, 111 of which used speech (Table 4.1). An additional 3 queries were issued by mistake (not realizing they had spoken the *"search"* command) and an additional 7 were issued before the participant had finished speaking (because the Web Speech API detected a pause). One participant did not search at all (the same participant that rated their familiarity with Canva as 5/5); the rest issued between 3 and 17 queries each.

Table 4.1: Summary of each participant's usage of ReMap. "# speech commands" includes all play, pause, and marker navigation commands. "# manual commands" includes all plays, pauses, seeking along the timeline, and clicking on markers.

	Total # intentional searches	**# accidental or pre-emptive searches**	**# intentional speech searches**	**# unique videos played**	**# speech commands**	**# manual commands**	**# deictic references**	**# successful deictic resolutions**
P1	17	1	17	13	41	3	5	1
P2	5	0	5	6	2	93	1	0
P3	13	3	13	7	20	23	10	1
P4	4	0	4	5	14	19	0	0
P5	0	0	0	0	0	0	0	0
P6	8	1	8	5	17	2	0	0
P7	7	1	7	6	24	10	0	0
P8	3	1	3	2	4	1	0	0
P9	8	1	6	6	0	49	0	0
P10	16	0	13	5	10	5	3	0
P11	10	1	10	11	9	51	1	1
P12	11	0	11	4	8	16	1	1
P13	16	1	14	7	18	0	3	2
Total	**118**	**10**	**111**	**77**	**167**	**272**	**24**	**6**

61/118 queries (52%) were new queries and 34/118 (29%) were reformulations of previous queries (*i.e.*, rephrasing a query to find better results). The rest were either attempts to fix a failed deictic resolution or fixing a transcription error. 55/118 queries (47%) began with either the phrase *"how to"* (42/118), or the phrase *"how do I"* (13/118).

8/111 speech queries included a speech recognition error. One participant tried to rephrase their query while speaking it, leading to long queries with some repetition (*e.g.*, *"resize this without aspect ratio without keeping aspect ratio"*). 6 of the 7 non-speech queries were a manual revision of a previous speech query (either error fixes or reformulations).

Deictic Resolution: Mostly Used for Canvas Elements

7 of 13 participants used deictic references at least once, and 22% (24/111) of spoken queries included a deictic reference. 23/24 deictic references referred to objects on the canvas (*e.g.*, text boxes, images, and charts); the other referred to a button on the toolbar. This reinforces RePlay's study finding (section 3.5) that participants mostly made action-oriented queries, rather than queries about tools.

Unfortunately, only 25% (6/24) of deictic references were successfully resolved to a name, mainly due to missing accessibility labels. For example, 10/18 unsuccessful deictic references were for charts, but charts in Canva do not have accessibility labels. We chose Canva for this study because it has more accessibility labels for canvas objects than most other creative applications; many do not label anything on the canvas at all. However, even Canva does not label all elements.

Video Navigation Behaviour: Some Preferred Manual, Others Preferred Speech

Participants watched a total of 77 videos (~6 videos per participant). In total, participants played videos by clicking on them 60 times, and by saying one of the *"play"* commands 61 times. Participants paused manually 60 times, and via the *"pause"* speech command 46 times. Participants manually seeked to points in the video by dragging on the timeline 124 times, by clicking timeline markers 28 times, and by saying the *"next/previous/repeat marker"* commands 60 times. In general, most participants seemed to exhibit a preference for either speech commands or manual navigation of videos (Table 4.1). This highlights the importance of providing both options in a multimodal interface, especially as peoples' preferences and needs may change depending on their task or environment [129, 188].

Participant Feedback: Speech for Navigating Timeline Markers was Most Helpful

As Figure 4.5 shows, participants generally found all four of ReMap's multimodal features moderately helpful. Based on our observations, most participants used the multimodal features to work and search or watch videos simultaneously. They confirmed this during the post-task interviews, with several participants explicitly mentioning multitasking as a benefit of ReMap's multimodal features. For example, one participant described how ReMap made them more efficient than their current strategies for help-seeking:

> *"When I'm designing graphics myself sometimes I get stuck, so I will have to stop everything and go to YouTube or Google to find it, but if I'm able to work while listening or ... watching it on the side, through voice, I think it will help with efficiency ... 'cause at one point I was able to ... move the things around while listening for the things I need."* — P4

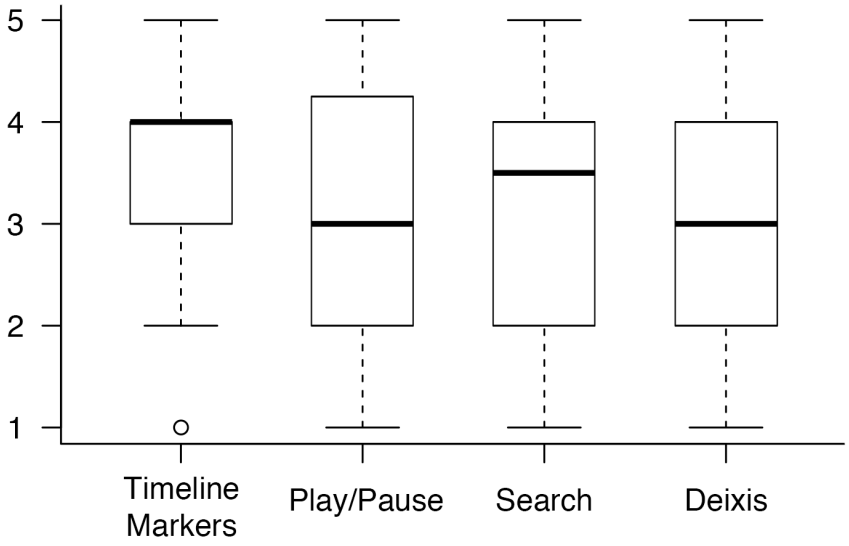

Figure 4.5: Distribution of participants' ratings of ReMap's multimodal features. 1 = not helpful at all, 5 = very helpful. Bold lines represent median values, and boxes illustrate interquartile ranges.

Timeline markers: Navigating the timeline markers was rated as the most helpful multimodal feature on average, and was the most common answer participants gave when asked what their favorite feature of ReMap was. This is mainly because it supported participants in multitasking:

> *"I ... found it helpful because while it was switching I could multitask, and I could just tell it 'hey, go to the next marker', and then I would try to do stuff on my own on the side until it plays."* — P6

> *"If I couldn't use speech then I would have to ... actually click on the videos, but since I'm already working on this part of the screen, then [speech is] just more convenient."* — P7

The markers themselves were useful (as the RePlay study (section 3.5) also showed) as they provided a shortcut to potentially relevant moments in the video. The speech commands for skipping between them made it easy for participants to back-up or fast-forward the video to a reasonable point. This is easier than having to specify a specific time interval to skip between, which Chang *et al.* [35] found can be difficult when following along with video tutorials. However, some participants did mention that they preferred using the mouse to hover over markers

before selecting them, so they could see the caption previews (Figure 3.3): *"I don't really know what they are until I scroll over them*" (P11).

Play/pause: Some participants found speech for playing and pausing videos to be helpful, as it allowed them to follow along with videos at their own pace, which prior work has also demonstrated the importance of [35, 184].

> *"It's really helpful because if I hear the info I want and I'm kind of following along I can pause it, take the instructions, apply it, and then ... continue watching ... without having to stop the fluidity of work."* — P8

> *"It was nice to be able to keep track of the video in the background while also starting to think through the next thing and having my focus on the other window."* — P13

Other participants preferred using the mouse to play and pause videos as they felt it was just as fast or easy, and didn't require them to remember the correct wording of the commands.

Speaking queries: Many participants found it helpful to speak their query out loud rather than type it, as it allowed them to stay focused on the task in Canva while searching. Some also said that it was easier or faster than typing. However, some other participants found it more difficult to speak their query, as they didn't always know exactly what to say when they started, and they were not used to speaking out loud to search. As P13 described, *"I had a lot of trouble getting my queries totally straight or thought out in my head before starting to speak them."* One difficulty sometimes encountered with ReMap's implementation was that it would cut off a participant's query and issue the search before they were done speaking, often because they paused slightly while thinking of what to say. This happened 7 times and was due to the Web Speech API prematurely detecting the end of a phrase. As one participant described, it made them feel rushed to finish speaking:

> *"It kept rushing me to think about what I wanted to search for ... Usually when I search by typing, I type out half of what I want and then think about ... what to type for the rest, whereas here as soon as i said 'how to add' I had to immediately know what I was searching."* — P6

Deixis: The 7 participants who attempted using deixis in their queries mostly found it helpful, although many noted that it needed some improvement to be useful, as it often failed to recognize the referenced element. When it did succeed, it helped participants include words they didn't know in their queries and was often easier and faster than saying or typing the words explicitly. However, it is likely most helpful in cases where users do not know the terminology, and Canva's interface has a relatively straightforward vocabulary. Deictic resolution might be used more frequently in software that has a more complex interface. For example, P1, who rated their prior experience with Canva as 1.75/5, said *"it might be more helpful for things that you don't know the exact terminology for, but I think I knew some of the terminology so just saying it felt faster."*

Two participants also pointed out that their eyes naturally went to the search field where their query was appearing while they spoke it, which made it difficult to look at Canva to reference elements: *"even though I was trying to click here I was focusing on [the search field]"* (P9).

Overall: Finally, when asked if participants could see themselves using ReMap in their daily lives, 9 participants said yes (3 saying for certain tasks only, and 2 saying if the technology improved first). The 4 participants who said no shared reservations about using speech in public or around people, and some said they simply prefer typing.

4.5 Discussion: The Potential of Multimodal Help Search

Overall, most participants were positive about multimodal help search, and many of the challenges they exhibited stemmed from either implementation issues or unfamiliarity with using speech and pointing for search and navigation. It is likely that an improved implementation combined with more time spent using ReMap would further increase users' success. This section outlines how ReMap's usability and implementation could be improved based on the study findings.

4.5.1 Usability Challenges With Speech for Search

The Midas Touch Problem: How best to initiate a search?

Since ReMap's speech detection is always on, participants occasionally encountered a "Midas touch" problem [100] of searching unintentionally. This is a challenge with a system that is always listening; personal assistants that can be activated via a voice command (*e.g.*, "Hey Siri") cause similar accidents [91]. On the other hand, needing to press a button to engage speech recognition can be burdensome, and in our study, accidental searches only happened 3 times out of all 121 speech queries issued. ReMap's current design (like many personal voice assistants today) was informed largely by the goal to make searching as quick and easy as possible, but for something as intentional as a search query, the added burden of pressing a button may still be more convenient than the risk of error. Future work should compare ReMap's current design of speaking a keyword to search with a button or keyboard shortcut to activate speech detection.

Pre-emptive searching: How to indicate a query is finished?

Several participants were frustrated by ReMap issuing a search before they had finished speaking their query, usually because they had paused briefly to think about what to say next and the Web Speech API interpreted this as the end of a phrase. Most voice assistants have this same functionality of automatically detecting when the user is finished speaking (*e.g.*, Siri, Alexa), but they do sometimes fail and cut the user off early. A study by Jiang et al. [102] found that these "system interruptions" accounted for about 10% of all transcription errors. In our study, they accounted for about 24% (7/29) of all transcription errors, including speech recognition errors and unintentional searches. Similar to the alternatives proposed above, future work should explore whether it is preferable to require the user to explicitly press a button, say a keyword (*e.g.*, "go"), or use a keyboard shortcut to indicate that they are finished speaking. Again, this adds an extra step to the process but would likely prevent errors and remove the pressure some

participants felt to rush to finish speaking.

How much to show users and when?

A few participants noted that while they were speaking their query, they would watch its transcription appear in ReMap's search field. This may have been distracting, as it took their attention away from the task at hand and made deixis feel less natural. Indeed, Kalyuga *et al.* [103] showed that splitting one's attention in the same modality increases cognitive load; in this case, user's visual attention was split between the software and ReMap's search field. We have since updated ReMap with a setting that can be toggled to specify whether to show the query as it is transcribed or only when it is finished; future work should compare these to see whether one consistently performs better, or if personal preference should dictate the setting. Seeing one's speech transcribed in real time (as many mobile voice assistants do) assures the user that the system is actively listening and gives them immediate feedback as to whether their speech has been accurately transcribed. But in the case where the search interface is not the primary application being used, it may add more distraction than the assurance is worth.

How to correct mistakes in a speech query?

In general, the problem of making a correction to a speech command or query is a difficult one [102,162,173]. Errors may be caused by the system (*e.g.*, speech transcription errors) or by the user (*e.g.*, saying a word they didn't intend). When such errors occurred with ReMap, participants mostly resorted to either editing the query manually by typing or repeating the entire query over again, echoing previous findings [102,162]. One participant corrected their query in real time by repeating it while speaking, much as one might correct oneself in normal conversation, which led to the entire utterance being transcribed as one long query. A smarter system might recognize this repetition and automatically use the corrected version only. Future work should explore how this and other methods for correction might be implemented. For example, Jiang *et al.* [102]

recommend letting users specify and repeat only a certain portion of a query that they want to correct. Shokouhi *et al.* [211] have shown that (at least on mobile) people prefer not to switch between speech and text when correcting a query, so one option could be to support speech commands such as "change X to Y" that allow users to replace incorrect words using voice alone.

4.5.2 Improving the Robustness of System-wide Contextual Support

ReMap contributes the first system-wide means for multimodal contextual help search. It achieves this by leveraging accessibility APIs to obtain system-wide information about the user's actions. However, this approach only works when accessibility APIs provide the necessary information, which they sometimes do not. This section discusses how future work might address these gaps.

Limitations of current accessibility APIs

ReMap is only able to detect interface elements that have been explicitly labeled with accessibility information by the application developer, which as both RePlay and prior work [36, 97] have shown, varies widely across applications. This chapter's study also found that even in one of the most thoroughly labeled creative software applications (Canva), some accessibility labels were still missing (*e.g.,* charts on the canvas). In general, accessibility labeling tends to be more common for interface "widgets" such as menu items and tools, and less common for the "insides" of applications such as editing areas or canvases [97]. However, there are existing efforts in the research community to make applications and systems more accessible using crowdsourcing and computer vision [79, 80], so accessibility labeling may improve in the future.

ReMap's reliance on accessibility APIs also complicates its setup and implementation substantially. As discussed in the Implementation (subsection 4.3.4), ReMap needed to be built as a MacOS application in order to access the MacOS Accessibility API, but this required building separate webpages to handle video playing and speech detection, as well as a custom server to

communicate between the webpages and ReMap. In our pilot tests and study, this setup generally worked well, although occasional delays in speech transcription did happen. However, opening a page in the web browser to detect speech would be impractical if a system like ReMap were to be deployed more widely. How else might we enable both multimodal support and detection of contextual information?

Leveraging Accessibility on Mobile Systems

ReMap may work more reliably as a mobile application, because mobile operating systems such as Android and iOS tend to have more accessibility features built in and more standardized interface components in their applications. Prior work has shown how mobile applications can leverage accessibility information to enable programming by demonstration [135]; a mobile version of ReMap could similarly use accessibility to provide contextual support across tablet and smartphone applications (which are increasingly used for creative work, as evidenced by the increasing prevalence of mobile apps from companies like Adobe[1]). In addition, speech and pointing are already common modes of interaction with mobile devices, which may make ReMap's multimodal interaction feel more natural.

Computer Vision for Detecting Interface Elements

Prior work has used computer vision as an alternative to accessibility APIs for recognizing interface elements [36, 45, 97]. Future work should explore how computer vision might enable system-wide contextual assistance instead of relying on accessibility labels. Computer vision may in many cases be *more* effective than accessibility as it could allow for higher-level interpretation of the elements being clicked. As our studies of RePlay and ReMap found, participants rarely found tool-level context useful for including in their search query. Even when accessing commands directly, Bota *et al.* [21] found that people often search for actions or outcomes rather than

[1] `https://www.adobe.com/ca/creativecloud/catalog/mobile.html`

command names. ReMap's addition of canvas elements helped make deictic resolution useful, as most deictic references participants made were for canvas elements. However, even when such elements have accessibility labels, they tend to be very general. Higher-level semantics about the element may be more useful; for example, a user might reference a photo of a sky when searching for help, in which case knowing that it shows a sky might be more useful than just knowing that it is a photo.

Of course, computer vision may also have limitations compared to accessibility APIs; the description or meaning of something is not always apparent from its visual attributes alone. For example, many tools in creative applications are represented by icons only; their names are not displayed. Or, the action a tool performs may not always be apparent from its visible name.

System-wide Support vs. Detailed Contextual Knowledge

ReMap illustrates both the benefits and drawbacks of system-wide, application-general assistance. While it provides users with a consistent interface for finding help across multiple applications, it is not able to tailor itself to a specific application the way an integrated plugin could. We believe the benefits outweigh the costs, especially given the potential improvements mentioned above that could be made to ReMap's approach for gathering contextual information. Nonetheless, one potential middle ground could be to build ReMap as a browser extension (similar to CheatSheet [231]). As web applications are becoming more prevalent than desktop software (with some laptops running *only* web applications, *e.g.,* Chromebooks), this might be a fair compromise. A browser extension could use voice recognition directly via the Web Speech API, enable deictic resolution and context recognition by accessing a webpage's document structure, and provide consistent support across multiple applications (as long as they are on the web).

4.6 Conclusion

ReMap introduces multimodal interaction for quick, in-context help-seeking by leveraging the strengths of multiple modalities. Users can search for videos using speech, use deixis to include application-specific terminology, and use speech to navigate videos. An initial study showed that ReMap helps people stay focused on their task while navigating help resources, and highlighted several important challenges with multimodal search.

RePlay and ReMap demonstrated how bringing learning videos into the user's context can help people find procedural help while working toward a specific desired outcome. The next chapters in this dissertation break apart the goals of *process* and *outcome*, exploring how contextual systems can support users who desire only one or the other.

4.7 Acknowledgements

This chapter, in part, is currently being prepared for submission for publication of the material. C. Ailie Fraser, Julia M. Markel, N. James Basa, Mira Dontcheva, and Scott Klemmer. The dissertation author was the primary investigator and author of this material.

Chapter 5

LiveClips: Contextual Recommendation of Inspirational Clips from Live Streamed Videos

Tutorials are helpful for accomplishing specific goals, but people do not always have a particular outcome in mind when using creative software – sometimes users seek resources for general learning or inspiration. One such resource is live streamed videos, where artists share a window into their creative process. This chapter explores the benefits and challenges of using such videos for learning and inspiration. We find that despite the wealth of expert knowledge in live stream archives, their length and volume makes them hard to search or browse. To address this challenge, we introduce *LiveClips*: a system for automatically selecting inspirational segments from live streamed videos and recommending them to users in the context of their creative workflows. We present three methods for recommending and displaying video clips with varying levels of contextual support, and implement them in a popular creative application, Adobe Photoshop. We compare the accuracy of LiveClips' clip ranking to human ranking and present initial user feedback on the three prototypes.

5.1 Introduction

Browsing and exploring inspiring examples is a key part of the creative process [13, 75, 92, 208, 210]. Prior work has shown that seeing examples throughout the entire process, including

the beginning, middle, and even towards the end, is valuable [118, 213]. One popular source for creative examples is online communities such as 500px, Behance and Dribbble[1]. However, these tend to showcase finished projects, which give the viewers little to no insight about *how* or *why* a project was created.

Recent work has shown that seeing the process behind an artist's work is beneficial for creativity, as it encourages self-reflection on one's own process and methods [109]. Some artists share works-in-progress, how-to tutorials, and videos describing the process that leads them to a final product, but these highly curated windows into process require time and effort for the creators to produce and share.

Many artists have begun to broadcast live video as they work on tasks such as graphic design, crafting, drawing, and music through platforms like Twitch and YouTube[2] (Figure 5.1). Live streaming allows creators to share their unedited process *while* they work. As a result, there is a rapidly growing collection of archived live stream videos that contain both inspirational and educational content regarding artists' creative processes. However, finding moments that are relevant and personally inspiring to a creator from this large collection of videos can be difficult, making live streams an under-utilized source for potential examples.

This chapter introduces LiveClips, a system for automatically selecting inspirational segments from long live streamed videos and recommending them as examples to users in the context

[1] `500px.com`, `behance.net`, `dribbble.com`
[2] `twitch.tv`, `youtube.com`

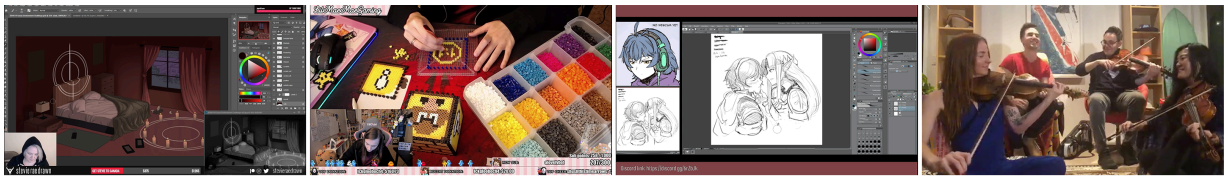

Figure 5.1: Examples of creative live streams on Twitch, YouTube, and Facebook. Artists stream videos of themselves working on creative projects. Sources for video screenshots, from left to right: `bit.ly/2SK5zWE`, `bit.ly/2Bv9Y69`, `bit.ly/2SJYFRa`, `bit.ly/2TK12Rq`

of their creative workflows. We choose to present examples inside the user's software in an effort to make examples more available throughout the entire creative process. Contextually available examples increase the likelihood of unexpected, or "serendipitous" discoveries, which research has shown can spark new ideas in a wide range of creative domains, such as scientific research, writing, and visual art [13,16,51,56]. LiveClips makes examples pervasive in the creative process to promote and encourage serendipitous moments of inspiration.

LiveClips combines telemetry and computer vision techniques to automatically segment long videos into short 25-second clips, crop clips intelligently to a thumbnail size for easy viewing, and recommend clips to creative software users based on their usage behaviour. We demonstrate LiveClips' method by using it to automatically extract and rank clips from 17 live streamed videos of artists working in two popular creative applications, Adobe Photoshop and Illustrator. We focus on digital art tasks such as design and illustration, as these are currently popular tasks for live streaming, and the software used for these tasks has wide audiences.

We compare LiveClips' ranking algorithm to human ranking and find that LiveClips is able to predict a clip's inspirational value with reasonable accuracy. To demonstrate how short inspirational clips can be contextually embedded in creative software, we present a design space and three prototypes that lie within this space, implemented in Adobe Photoshop (Figure 5.6). Initial user feedback suggests that this approach is promising and warrants further study. In summary, we make the following contributions:

- a formative understanding of creative live streaming from the perspective of live stream viewers,
- a method for extracting short (25-second) clips from long live streamed videos and cropping them to thumbnail size,
- an approach for selecting example clips to present inside creative software based on their visual properties, the user's tool usage, and the presentation location within the software,

- three prototype implementations in a creative application and validation of the ranking approach with human raters.

5.2 Related Work

5.2.1 Creative Live Streaming

Live streaming democratizes the studio-apprentice model, enabling anyone to see experts' in-context choices by working alongside them [199]. Perhaps the three most popular genres for live streams are video gaming [83, 131, 181, 215], programming [52, 82], and lifestyle [144, 222]. Popular live streaming platforms that host a variety of live stream genres include Twitch, YouTube, Facebook, Instagram, and Periscope. Creative live streams can be found on all of these platforms, but platforms dedicated specifically to *creative* live streaming have also emerged, such as Picarto, Pixiv Sketch, and Behance [3]. Twitch defines creative work as "visual art, woodworking, costume creation, prop building, music composition, or any other process in which you entertain and connect around a creative activity" [160]. This chapter similarly considers creative live streams as the streaming of any activity that focuses on creating a novel artifact. This differentiates creative live streams from other popular types of streams like video games or lifestyle activities. Section 5.3 of this chapter analyzes the domain of creative live streams, comparing and contrasting them with prior work on live streaming in other domains.

5.2.2 Examples are Important for Creative Inspiration

Searching and browsing examples is an important part of the creative process [13, 75, 92, 161, 208, 210]. While search engines are a common means for finding examples, directed search can prevent serendipitous discovery [16]. Happening upon an unexpected or even seemingly unrelated example can spark new ideas that the creator wouldn't have otherwise thought of [16,

[3] `picarto.tv`, `sketch.pixiv.net/lives`, `behance.net/live`

51]. Creative software can support serendipitous discovery by presenting the user with examples while they work [13, 92, 118].

The selection of examples is important; prior work shows that diverse and far-ranging examples lead to more novelty in creative output [33] and more diverse sets of ideas [212] than examples that are similar to each other and the task at hand. The timing of examples is also important; Lewis *et al.* [133] show that people can be primed to be more creative through exposure to examples before a creative task, while Kulkarni *et al.* [118] show that seeing examples early on in the process as well as interspersed throughout the process improves creativity. However, diverse examples can actually be harmful if they are presented while the user is being productive [34]. Siangliulue *et al.* [213] found that the most novel ideas occur when examples are only shown at the user's request rather than automatically shown by the system. However, users may not always remember to look for examples, and so systems that ambiently update or recommend examples when the user is idle have also shown creative benefits [190, 213].

LiveClips selects a diverse set of examples that are relevant to the user's context. Guided by the research above, our prototype implementations explore variations in the timing and availability of examples.

5.2.3 Contextual Recommendations Support Learning in Software

Given the increased focus in education (*e.g.*, [185]) and software learning (*e.g.*, [75,77]) on active learning, or "learning while doing", we believe similar benefits may arise by integrating the process of inspiration with the process of doing. Despite the known benefits of seeing examples throughout the creative process [118], most creative software today lacks support for in-context inspiration. Contextual presentation of learning content is an effective method for supporting learning while doing [77, 99, 150, 153]; in this work we explore whether contextual presentation of examples can similarly support "inspiration while doing".

Embedding any kind of content in-application runs the risk of interrupting or distract-

ing the user. Therefore, in-application content should be unobtrusive but easy to access [77]. Tooltips are one promising avenue for this, as they only require a hover to access and can be easily dismissed by mousing away. Inspired by ToolClips [77], this work also uses tooltips as one potential interface for contextual assistance. CommunityCommands, a command recommender for AutoCAD [153], demonstrated that personalizing recommendations based on the user's own tool use makes them more likely to be helpful. Over a 6-week user study of CommunityCommands, Li *et al.* [136] found that recommendations based on the user's short-term tool use are preferred over those based on the user's all-time tool use, as the former tend to be more contextually relevant. LiveClips similarly bases recommendations on the user's recent tool use.

5.2.4 Segmenting Videos to Make Them More Browsable

This work uses videos as the source for inspirational examples. Videos have the advantage over text and static images that they show the process of using a tool, rather than just the outcome, and they provide visual demonstrations that are directly relatable to the visual user interface [77].

Extensive prior work has explored automatically generating educational video clips from screencast videos of software use [11, 40, 125, 165, 184]. Though some methods rely on matching telemetry data for these videos [40, 78, 125], others have shown that computer vision alone can be used to detect tool selection events when such data is not available [11, 184]. In this work, we use telemetry data to segment live streamed videos into clips and rely on computer vision for the presentation and ranking of clips.

Given a long video with associated tool usage, prior work suggests techniques for extracting clips that demonstrate particular tools [40, 125, 184], and guidelines for selecting clips with the best learning value [125]. Lafreniere *et al.*'s recommended guidelines [125] include keeping clips short (15-25 seconds), using clips that show clear visual change to the document, and avoiding clips that show multiple unrelated actions. LiveClips builds on this approach to extract and rank clips from live streamed videos and explores how the characteristics of an inspiring clip

might differ from that of an instructional one.

5.3 Formative Work: Understanding Creative Live Streams

Creative live streams are a unique and rapidly growing source of data that have yet to be deeply studied. To understand the current landscape of creative live streams as well as their applicability to inspiration in software, we conducted content analysis and surveys, exploring the following two questions:

1. **What are creative live streams?** For a general sketch of creative live streams, we present a content analysis of a sample of live streams that illustrates the range of creative content people stream and the different types of creative live streams.

2. **Why do people watch creative live streams?** To understand the motivations behind creative live stream viewers, we present findings from three online surveys with 165 viewers that highlight learning and inspiration as key motivators.

We found that viewers often seek to learn and be inspired from creative live streams. Notably, inspiration is a much more prominent theme compared with prior work in other live streaming domains such as gaming. However, many streaming platforms are not designed to support these goals, and watching archived streams is tedious. These findings further motivate our goal to make creative live streams available to users in the context of their own workflows.

We were also interested in streamers' experiences producing creative live streams, so we conducted interviews with 8 streamers. These interviews did not directly inform the design of LiveClips as it does not address the streamer's experience, but our findings provide useful insights that future work may build on towards improving the creative streamer's experience. Appendix A reports our interview methodology and findings in detail.

5.3.1 What Are Creative Live Streams?

Live stream videos (Figure 5.2) typically show the artist's full screen (when working on a computer) or workspace (for physical work) and a camera view of their face. Live streams usually also feature a live chat, allowing viewers to communicate with each other and the artist.

There are two main things that make live streamed videos different from other types of videos demonstrating creative work, such as tutorials. First, live streamed videos usually show a real-world process, imparting both creative and procedural knowledge, while tutorials often show contrived tasks, and impart mainly procedural knowledge [226]. Second, artists often start without an exact goal in mind, and it can be inspiring to watch someone go from a blank canvas to a beautiful piece of work, though this also means watching the less "glamorous" parts of the creative process such as silent thinking or tedious, repetitive tasks.

To learn more about creative live streams, we studied two popular platforms: Twitch and YouTube. As these large platforms cover many types of content, we narrowed our investigation to the *Creative* category on Twitch and the *Adobe Live* video series on YouTube. Through this, we see how streams and communities differ across platforms. Our findings also inform the design

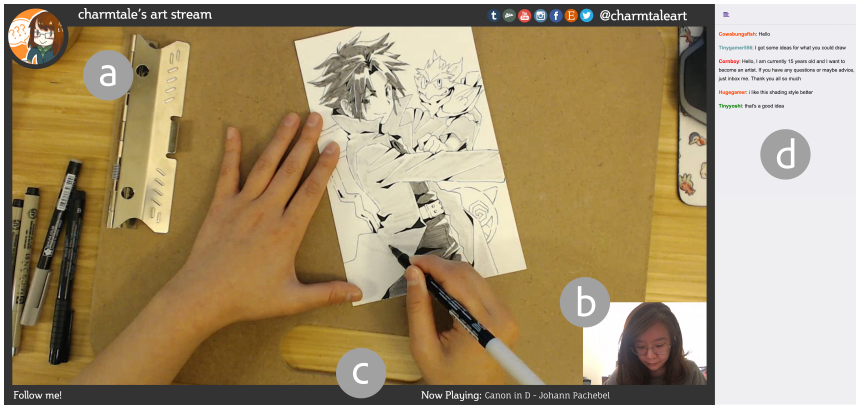

Figure 5.2: A typical creative live stream setup. (a) A camera or screencast displays the artist's workspace. (b) A second camera shows the artist's face, usually via their computer's webcam. (c) Graphical overlays provide ambient information about the artist (*e.g.*, social media pages) and display interactions with the audience (*e.g.*, pop-ups that appear when viewers subscribe or donate to the stream). (d) Live chat allows viewers to communicate with the streamer.

and implementation of LiveClips, which uses videos from these same two categories as its corpus for extracting short inspirational clips.

Creative live streams on Twitch: Content analysis

To better understand the format and content of creative livestreams, we analyzed a sample of videos on Twitch, one of the most popular platforms for live streaming. Twitch launched its *Creative* category in 2015 [160]. To deal with its explosion in popularity, Twitch replaced the *Creative* category with six more-specific categories in September 2018: *Art*, *Music & Performing Arts*, *Science & Technology*, *Beauty & Body Art*, *Food & Drink*, and *Makers & Crafting* [192]. For each category, we gathered aggregate metrics about streamers and viewers. We watched and took notes on a sample of 29 videos. We identified four common types of creative live streams that will appear throughout the chapter: *Teaching*, *Making*, *Socializing*, and *Performing*.

Although the current LiveClips system focuses on digital creative work, the platforms we analyzed include streams of both digital and physical work. This provides a more comprehensive look at the general creative live streaming space, to inform both LiveClips and other future work on creative live streaming.

Methodology: To measure the popularity and activity in each of the six creative categories, we queried the Twitch API 4 times a day for 7 days to obtain the number of currently-live streams and number of currently-watching viewers in each category.

We also used the Twitch API to download metadata about the videos in each category (limited to top 600) and randomly selected 50 archived English live stream videos. Four annotators (including the first author) watched each of these videos. Ten videos were not available for viewing and thus excluded (either because their archive expired between being downloaded and being annotated, or because they were only available to subscribers of a channel). Another 11 videos were excluded as they showed video games, TV show reruns, or live event coverage. While these videos were categorized as creative on Twitch, they did not reflect our definition of creative

Table 5.1: Summary of popularity of Twitch's creative live stream categories. The number of currently-live streams and currently-watching viewers were collected 4 times a day for a week and then averaged.

Category	Avg. # live streams	Avg. # live viewers	Avg. # viewers / stream
Art	339	6417	21
Beauty & Body Art	5	177	17
Food & Drink	19	1088	64
Makers & Crafting	40	680	16
Music & Performing Arts	286	6881	24
Science & Technology	91	1155	12

work, namely the creation of a novel artifact. This yielded 29 videos. For each, annotators took notes in a structured spreadsheet on the content presented, camera setup, overall structure of the stream, artist's presentation style, and chat activity.

Results: Most streamers focus on work & engage with viewers: Table 5.1 shows overall metrics for the creative categories on Twitch. The most popular categories by far are *Art* and *Music & Performing Arts*. The category with the most viewers watching per stream is *Food & Drink*, likely because there are fewer streams to choose from relative to the number of interested viewers. These communities are small relative to the most popular games; for example, the game Fortnite has between 5,000 and 10,000 streams live on Twitch at any given time, with around 100,000 total viewers watching.

The videos span a range of creative activities (Table 5.2). The average video length was 3h46m, not including time spent gaming – a few artists combined both creative work and video gaming into one stream, spending the first part on creative work then switching to gaming when they were finished. The shortest video was 1h3m; the longest was 7h56m. These videos are notably longer than most non-live-stream videos, which are already difficult to browse and navigate [40, 165, 184]. This further motivates LiveClips' goal of extracting shorter clips, as a multi-hour-long video is not likely to be helpful in context while the user is working.

Almost all videos contained either a screencast view for work being done on a computer (13/29) or a camera view for physical work (15/29). One showed a distant camera view of the

Table 5.2: Creative activities shown in a random sample of 29 live streams from Twitch's creative categories, and the primary type of structure each stream exhibits.

Category	**Activity (# videos if >1)**	**Primary type of stream**
Art	Multimedia production	Making
	Digital drawing (4)	Making
	Animation	Teaching
Beauty & Body Art	Makeup	Socializing
	Makeup (3)	Making
Food & Drink	Cooking	Teaching
Makers & Crafting	Making foam props	Teaching
	Sewing quilts	Socializing
	Bead art (2)	Making
	Assembling models	Making
	Assembling models	Socializing
	Woodworking	Making
	Pottery	Making
Music & Performing Arts	Music production	Performing
	Music production	Making
	Acting & improv games	Performing
Science & Technology	Building a computer	Making
	Programming (3)	Making
	Game development (2)	Teaching
	Talking about technology	Socializing

artist producing music in a studio. Most (26/29) showed the artist's face: in 10 as part of the main camera feed, and 16 as a separate feed overlaid in a corner (as in Figure 5.2). Of the 13 videos that showed screencasts of digital work, 10 showed the artist's face overlaid in a corner and 3 did not show the artist's face at all. Almost all artists (27/29) talked out loud while streaming; of the two silent streamers, one occasionally posted in the chat. Most artists talked about a mix of their work and other topics (18/29). Some talked only about their work (9), or only about other topics (1). One was a variety show, so the talking *was* the work. Many videos (19/29) included background music.

Most artists engaged with the chat at least sometimes (24/29). 18 artists engaged fre-

quently with the chat, and 6 occasionally. Three videos did not show a chat replay despite the artist referring to the chat; we assume it was not saved or had been hidden. In all 26 remaining videos, viewers asked questions at least occasionally, or in some videos (9/26) frequently. In half of these videos, all chat questions appeared to get answered; in the rest, some (7/13) or many (4/13) questions went unanswered. In 2 videos, most chat questions were answered by other viewers or moderators in the chat.

Four common types of creative live streams: We identified four common types of creative live streams. We also observed these in our interviews with streamers (see Appendix A). Sjöblom *et al.* [216] offer a similar characterization of video game live streams; we found some key differences and fewer overall types of structures. Table 5.2 shows the primary type of each stream in our sample set. These are general high-level trends; some streams bridge multiple types.

Teaching streams have an instructional focus, where the streamer is educating the viewers. These include step-by-step how-to demonstrations of tasks such as cooking a recipe, producing a photo-editing effect, or creating DIY costumes. Other examples include critiquing others' work, answering viewers' questions, or explaining a topic.

Making streams focus primarily on creative work and process, but not explicit teaching. These include an artist silently drawing, a streamer attempting a new task they have not tried before and talking their way through it, and an artist making pottery and describing *what* they are doing but not *how*.

Socializing streams feature the streamer chatting casually with viewers, often while working on a project, such as makeup or sewing (but the project is not the main focus). These are often described as "chill" streams. Socializing streams often have tight-knit communities; the streamer will recognize the names of viewers in the chat and ask them how they are doing.

Performing streams feature the artist performing their work. Naturally, these mostly include performative arts like music and acting (*e.g.*, as opposed to drawing). Like with Making, the focus is on the artist's work; in this case the artist does not talk about what they are doing,

they just do it. Performing streams differ from non-live recordings in that they often take a more casual improvisational form, rather than scripted performance (*e.g.*, musical "jam sessions" or improv acting).

Within each type, the amount of interaction between the streamer and the audience varies. Some streamers hold "request streams" or "Q&A streams", where the content and flow are determined by audience requests or questions, respectively. Some hold contests or games. A request stream could have audience members requesting songs for a Performing stream, a topic for a Teaching stream, or a particular artifact for the artist to make in a Making or Socializing stream. Finally, as Table 5.2 shows, the majority of live streams showing digital creative activities (*e.g.*, digital drawing, programming) were Making streams, and the rest were Teaching streams. Accordingly, LiveClips focuses on Making streams as its video corpus, though we believe our approach could extend to other stream types as well.

Creative live streams go professional

While many live streams are run by individuals, professionally-run streams are also growing in popularity. Adobe, a company that produces creative software, hosts live streams on a regular schedule multiple times a week[4]. These can be viewed on Behance or on Adobe's Creative Cloud YouTube channel. They host two live stream series: *Adobe Live* is a Making stream that happens for 6 hours (three 2-hour sessions) three days a week, and it features guest artists usually hosted by someone who works at Adobe. *Daily Creative Challenges* is a Teaching stream that happens for 30 minutes five days a week. It complements contests organized by Adobe to teach new skills. YouTube reports these streams having between 2,000 and 8,000 views each; it does not distinguish between live and replayed views.

Adobe Live differs from most streams by featuring two people. Typically there is a host and an guest artist (Figure 5.3), but sometimes two artists work together. Adobe hosts different artists

[4] `behance.net/live`

every week across a range of creative disciplines (*e.g.*, graphic design, UX design, photography, video editing). Typically, the host moderates the stream, responding to chat messages and passing on questions from viewers to the artist so that the artist can focus on their work. When the stream includes two artists, they trade off hosting. The featured artists are usually practitioners, not teachers; their work mainly serves as inspirational demonstrations, while also giving the the community a chance to engage with questions and comments.

5.3.2 Why do People Watch Creative Live Streams?

To understand the motivations and challenges of creative live stream viewers, we conducted three surveys over 1.5 years with 165 people: two with *Adobe Live* viewers; one with viewers of any creative live streams on the Web. All three surveys were voluntary. We found that creative live stream viewers watch streams primarily for *learning* and *inspiration*; community engagement and entertainment were also popular reasons for watching streams. Compared with prior work on live stream viewers in other domains, inspiration is a much more prominent theme in these survey responses. This finding motivates LiveClips' focus on contextual inspiration.

[5] `behance.net/dailycreativechallenge`

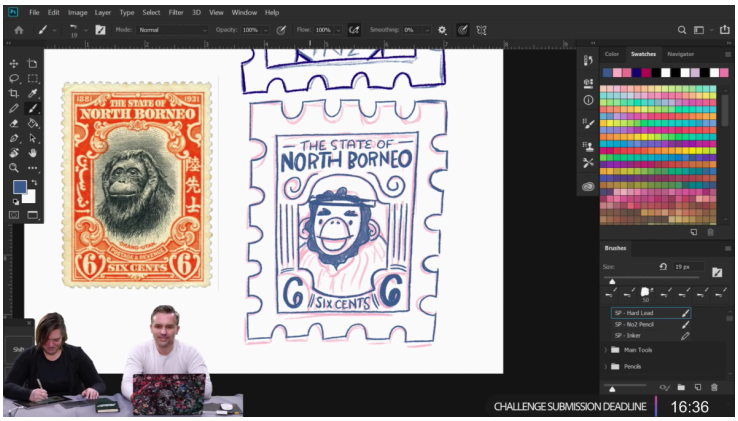

Figure 5.3: The *Adobe Live* series features a guest artist (bottom left) and a host (bottom right). The artist is working on a digital drawing, and the host is looking up at the live chat feed, engaging with the audience (`youtu.be/yYDmQhg_1uE`).

Survey methodology

The first survey with *Adobe Live* viewers (*S1*) was posted periodically in the chat and overlaid on the stream for four months (August - December 2017). It asked about viewers' experience with creative software, the reasons they watch creative live streams, what other creative live streams they watch besides *Adobe Live*, and how *Adobe Live* streams could be improved. 98 people completed this survey.

A year later, *Adobe Live* had changed considerably: more frequent streams, more audience interaction, and wider and more regular marketing. In January 2019, we conducted *S2* to gain additional insights about viewer motivations and challenges, focusing especially on the live chat experience. The survey was sent directly to previous winners of Adobe's *Daily Creative Challenges* who also showed up regularly in past chat logs of Adobe streams. 41 people completed this survey.

Finally, to zoom out and capture a broader range of creative live stream viewers, we conducted a third survey (*S3*) with viewers of creative live streams on *any* platform. The survey was disseminated with a snowball method, via the researchers' personal social media accounts. Participants were required to have viewed creative live streams before, which were defined as "activities such as visual art (drawing, painting, etc.), crafts, music performance, cooking, DIY projects, programming, etc." This survey asked viewers about the streams they watch, motivations for watching them, examples of things they learned from them, what else they do while watching, and on which platforms they watch. 26 people completed this survey.

What do people watch, and where?

The most popular platform overall was YouTube (74), with Twitch second (30) (Table 5.3). This is likely skewed by the fact that *Adobe Live* is on YouTube. *S3* had a smaller sample size but found YouTube and Twitch to be equally popular.

S3 respondents listed content genres they frequently watch live. Categories that came

Table 5.3: All platforms listed more than once in at least one survey by respondents when asked where they watch creative live streams.

	Survey 1 $n = 98$	**Survey 2** $n = 41$	**Survey 3** $n = 26$	**Total** $n = 165$
YouTube	40	17	17	74
Twitch	9	4	17	30
Facebook	5	4	4	13
Periscope	1	1	2	4
Instagram	-	-	6	6
Phlearn	2	-	-	2

up more than once were programming (11/26), cooking (6/26), digital art (*i.e.*, digital painting, photo editing) (5/26), music (4/26), physical artwork (i.e. drawing, painting) (3/26), 3D modeling (2/26), and DIY (2/26).

Viewers watch for learning and inspiration

Viewers responded similarly about motivations in all three surveys. Despite the differences in sample sizes and populations, this suggests that *Adobe Live* viewers' responses may often align with viewers more broadly. Across all three surveys, learning was the most common reason people chose for watching creative live streams (Figure 5.4). While learning has also been found to be an important goal for viewers in other domains such as gaming, the primary goal most often cited in prior work is entertainment [39, 52, 94, 142, 144, 235]. This difference may be due to the prevalence of Teaching live streams in creative communities.

Almost all free-form elaborations on viewer motivation mentioned learning. Unlike tutorials and lecture videos, live streams offer direct interaction with the streamer and other viewers, improving the learning experience [52, 142]. In this way they go beyond just learning content and catalyze "mentorship communities" of people with similar interests [52]. Learners can follow along like an apprentice in a studio, asking questions in the moment. This ability to see authentic, worked examples from start to finish reveals how the streamer makes decisions and recovers from errors [52]. Viewers often use the knowledge and techniques they learn from creative live streams

to inform their own work, as many *S1* respondents stated in free-form responses. *S3* asked for specific examples; 50% of respondents provided one. They include adopting new techniques such as photo editing operations, trying out a streamer's creative style for things like musical playing or code commenting, and learning how to achieve a specific goal like fixing a hole in a sweater.

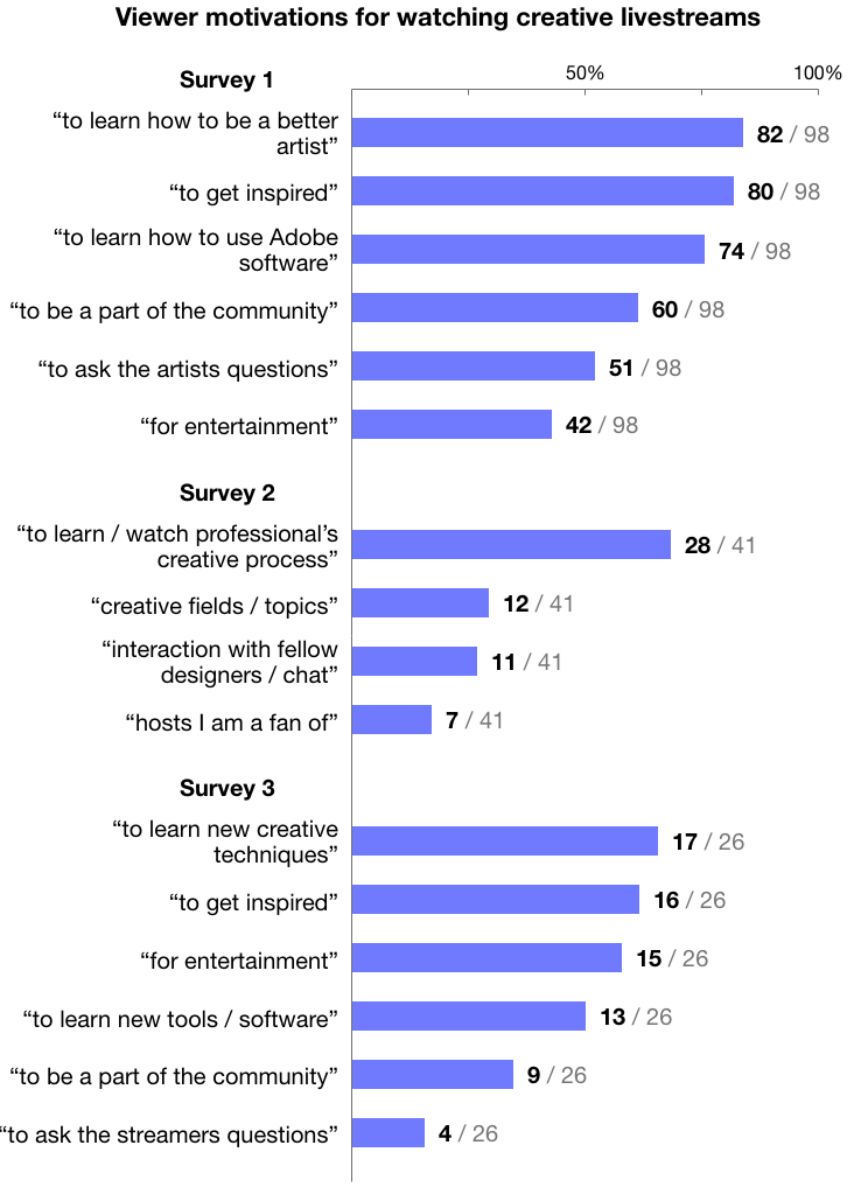

Figure 5.4: All three surveys asked why people watch creative live streams, allowing them to select all answers that applied from a list. This figure shows all responses chosen by at least 15% of respondents in each survey.

In addition to learning, many also reported watching for inspiration / motivation. With one exception [39], primary work has not reported inspiration as a goal. Cheung & Huang [39] describe "the Inspired" as one of nine personas for gaming live stream viewers; watching someone stream the game inspires them to play it themselves. However, a large majority of gaming stream viewers watch for entertainment, learning, or providing commentary. While inspiration can be beneficial in many genres, we believe it is especially salient in creative live streams due to inspiration's value for creative work [92].

In both *S1* and *S3*, inspiration was the second most popular motivation for watching creative live streams. In addition, 27% (26/98) of *S1* respondents specifically mentioned inspiration or motivation in free-form responses. 10% (10/98) also mentioned that the videos helped increase their own motivation and confidence as artists. As one respondent explained, *"[I] like watching artists work because it takes the mystery out of what they do."* Another said, *"Watching experts make mistakes gives me confidence."*

Creative work is often a solo activity, and its nebulous nature can make it hard to stay motivated as an artist, often causing creative "blocks" such as writer's block. Watching someone else work can motivate viewers to keep going, as well as give them new ideas to try. Respondents in all three surveys mentioned this in free-form responses. For example, one *S1* respondent said they watch live streams for *"getting myself inspired and hyped before I start working."* An *S3* participant said, *"It's fun seeing someone else's creative process, and usually motivates me to do my own side projects."* LiveClips therefore explores how seeing clips from creative live streams in context might help inspire and motivate people while they work on their own creative projects.

Viewers also watch for community and entertainment

People watch all kinds of live streams for entertainment [39, 52, 94, 144, 235]. It may be the streamer's personality or style, the chat, or the content itself. People also watch live streams for community. Viewers often feel emotionally attached to the streamer [95, 235], enjoy connecting

and conversing with other viewers [94,142,143], and enjoy being able to influence the streamer's content or process in real time [144]. Live stream communities often lead to longer-term chat groups on other platforms [52,144].

All three surveys found community and entertainment to be secondary motivations (Figure 5.4), showing that these are also important motivators for creative live stream viewers. Several *S1* respondents valued the company of other creative people while they worked alone. To investigate this further, Surveys 2 and 3 asked what people do while watching live streams (multiple choice). 68% (28/41) of *S2* respondents said they watch while doing creative work. 69% (18/26) of *S3* respondents said they watch while working on something, and 31% (8/26) said they work on a similar task as the streamer. In this way, creative live stream communities offer a virtual co-working space for people who would otherwise be working alone.

Respondents in all surveys specifically mentioned that the *combination* of learning and entertainment was what drew them to live streams. This echoes Lu *et al.*'s findings with knowledge-sharing streams [144]: they are appealing because they disseminate knowledge in a more relaxed, casual way than tutorials or lecture videos.

What are the challenges for viewers?

S1 and *S2* asked how the viewing experience might be improved. The most popular suggestions had to do with interactivity and engagement between the streamers and the chat. 17% (7/41) of *S2* respondents said their questions often get lost in the chat. Busy chat feeds are a problem in other types of live streams as well [158], but can be especially frustrating for viewers seeking to learn and ask questions. Two respondents in *S1* wished that hosts would interact more with the chat, and three others emphasized hosting skill, saying that the best hosts are able to keep the conversation interesting and interact meaningfully with the audience. Two respondents in *S2* wished there were more ways to involve the chat, *e.g.*, through quizzes or polls.

Several respondents also mentioned that the experience watching live stream replays could

be improved; one *S1* respondent said a summary document with important links and tips could help with reviewing the stream later, and three *S2* respondents wished they could view the chat and somehow be involved in the stream when watching replays. This agrees with Lu *et al.*'s findings [143] that it can be hard to learn from a stream after the fact, as navigation options are usually limited.

5.3.3 Summary of Findings

This section's analysis and surveys uncovered the many goals and motivations viewers have when watching creative livestreams. We also found that existing platforms do not support all these goals or offer help when goals conflict. Specifically, viewers often seek inspiration from creative live streams, but live stream viewing takes place out of context of the viewer's own work. Moreover, despite the wealth of expert knowledge and inspiration such videos contain, watching them as archives is tedious and difficult. Several survey participants mentioned the viewer experience for watching live stream archives is poor because the videos are long, have limited navigation, and include long periods of downtime and conversation with the then-live chat [143]. Twitch viewers can create "clips" and streamers can create "highlights" of interesting moments, but they must remember to do so, and such moments can seem out-of-context when viewed on their own.

Motivated by these findings, the remainder of this chapter explores how contextual video examples might help creative software users benefit from the wealth of expert knowledge hidden in creative live streams without having to watch all the downtime and unrelated conversation that comes with it. The LiveClips system does this by capturing moments of insight and inspiration from creative live streams and bringing these moments into the context of creative software users' workflows.

5.4 Design Space for In-Application Video Examples

To help inform our decisions in developing LiveClips and explore how users might interact with such a system during their creative workflow, we outlined a design space for systems that present contextual video examples (Figure 5.5). This design space is informed by our formative findings as well as prior work on contextual assistance in software. The following three sections present the LiveClips system, which includes three alternative interfaces that explore three different points in this space. This section also highlights where RePlay and ReMap (Chapters 3-4) fit in the design space as compared to LiveClips. Most notably, LiveClips differs from RePlay and ReMap in that it presents inspirational content rather than educational, and it presents short tool-focused clips rather than entire task-focused videos.

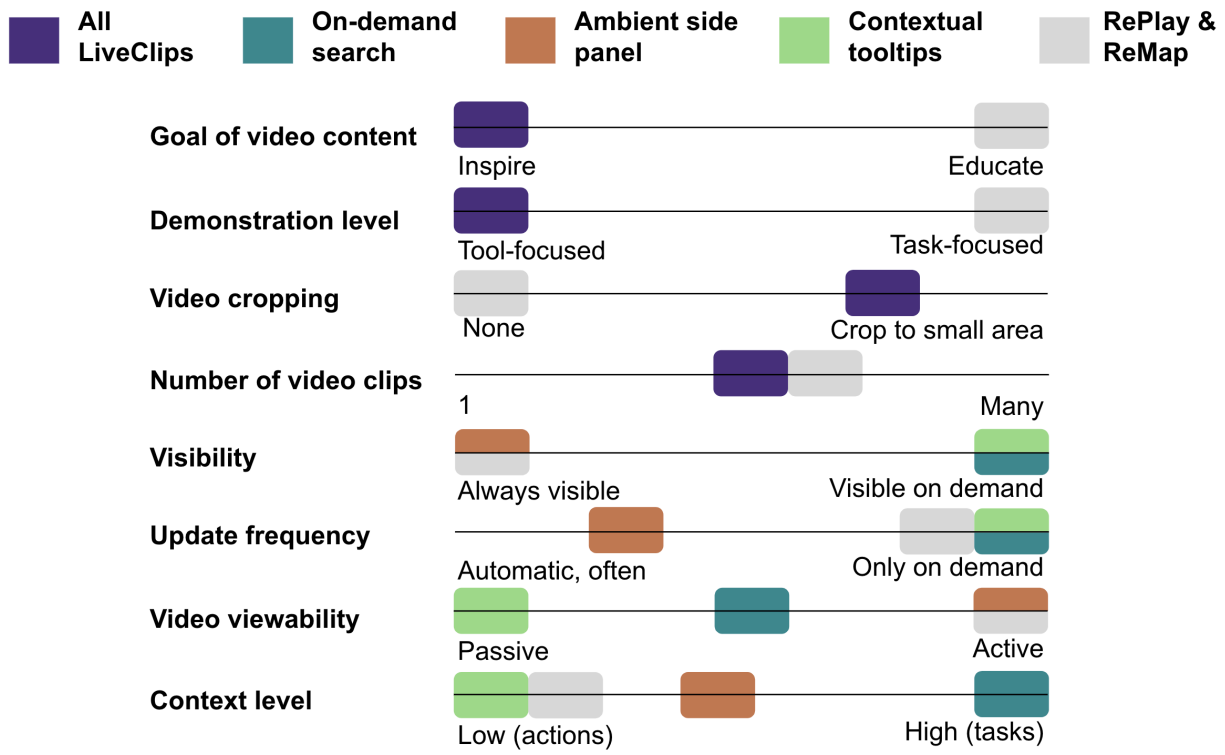

Figure 5.5: The design space for in-application video examples, as well as where LiveClips and RePlay/ReMap fit along each axis. LiveClips' prototype interfaces vary along the bottom four axes.

5.4.1 Goal of Video Content

Most prior work regarding contextual videos, including RePlay and ReMap, has focused on content that aims to educate the user, such as tutorials [184] and helpful tips [77]. Inspired by the known benefits of examples for inspiration [118] and our survey findings suggesting that live streamed videos are used for inspiration, this work explores how contextual live streamed videos might inspire users. These videos also have educational value, but this work focuses on them as a source of inspiration.

5.4.2 Demonstration Level

Videos of software use can be tool-focused or task-focused. A tool-focused video demonstrates using a single tool, while a task-focused video shows an entire task from beginning to end. Tool-focused videos are useful for quick contextual help that does not take away from the user's current work, for example reminding the user how a tool works [77] or showing alternative uses for a tool. Task-focused videos tend to be longer and are less useful in-the-moment, as they show a specific task that may or may not be relevant. Live streams tend to be task-focused, as they usually follow an artist through an entire task; this work focuses on extracting shorter, more generalizable tool-focused examples from task-focused live streams. RePlay and ReMap also extract relevant moments from within longer task-focused videos, but unlike LiveClips, they are meant for users who proactively seek help with a particular task, so they allow the user to browse the entire video and follow along with it.

5.4.3 Video Cropping

In-application examples are constrained by space; if content takes up too much space it becomes obtrusive, for example by blocking the canvas where the user is currently working. Therefore, contextual videos must be displayed at a relatively small size. For videos that were

created using full-screen video capture (as most live streams are), this can be problematic. Leaving a video un-cropped allows the user to see the full context of interaction but makes it difficult to discern any detail. Since RePlay and ReMap left videos un-cropped, they allowed users to open a single video in a larger window when seeing more detail was necessary. While this may have been practical for task-focused learning where users have other cues to indicate a video's relevance before selecting one (*e.g.*, caption previews), LiveClips aims to show users many examples at once of tool use that may only take place in a small part of the screen. Prior work has explored how to best present video demonstration clips when they cannot be viewed in full screen, by cropping or enlarging sections in the video where mouse movement and canvas changes occur [40]. Inspired by this, LiveClips crops videos to the area that shows the most visual change, in an effort to make examples focused and easy to browse.

5.4.4 Number of Video Clips Shown

Examples could be displayed one at a time, taking up the least amount of space but also providing less variety. Alternatively, displaying many videos gives users more options but potentially overwhelms them. Prior research on contextual videos recommends displaying multiple videos to demonstrate the range of uses for a tool and increase the likelihood that the user will find at least one useful [125,150]. Similar to RePlay and ReMap (which displayed five videos at a time), our prototypes display four videos at a time in an effort to provide some variety without overwhelming the user or taking up too much space.

5.4.5 Visibility

Examples can be always visible while the user is working (like with RePlay and ReMap), or visible only on demand. Examples that are always visible are more likely to be seen but can be distracting, while examples that are not easily visible and shown only on request are more likely to be ignored or missed [190]. Different users may have different preferences for how visible

they want their examples to be, just as some viewers like to watch creative live streams while they work, while others may not. Our prototypes explore three variations along this axis.

5.4.6 Update Frequency

Recommended examples can update in response to an explicit user action (*e.g.*, a query), an implicit user action (*e.g.*, being idle for some time or opening a new document), or automatically at a regular time interval. Updating in response to explicit user actions was appropriate for RePlay and Remap, which are intended to be used when users have a specific question. But for a system like LiveClips that aims to elicit inspiration, the ideal approach is less clear. The trade-offs between these strategies have been widely explored in the literature, and it seems there is no globally optimal solution [34, 190, 213]. Automatic updates can be useful because they require zero effort from the user and can highlight an example in the moment [190], but they can also distract the user during periods of focused work [34] or make the interface feel uncontrollable [170]. Updates in response to an explicit request are less distracting and give the user more control over their attention but rely on the user to know when an update would be useful [190, 213]. Update frequency is also tied to Visibility; recommendations that are only visible on-demand only update (visibly) when the user requests them. As with Visibility, the ideal solution may depend on the user's preferences; our prototypes explore three variations.

5.4.7 Video Viewability

When presenting non-static content such as videos in-app, the way in which the user interacts with and views the content may affect its usefulness. On the passive end, videos could play automatically when the content appears, which removes the need for the user to decide whether or not to watch a video but runs the risk of being distracting. On the other end, videos could be shown as a static thumbnail that only animates when the user intentionally interacts. Most prior work [40,77], including RePlay and ReMap, has adopted the latter approach, however

such work also suggests that in-context content should have a low threshold for engagement; the user shouldn't have to break too much from their task to interact with recommended content [77]. LiveClips explores three variations along this axis.

5.4.8 Context Level

Examples can vary in how contextual they are to the user's behaviour. Low-level examples respond to the user's individual actions, such as the tools they use. High-level examples respond to the user's task or intent, which can be inferred from sequences of commands or by analyzing the document being edited. RePlay and ReMap use low-level context to augment queries, but the queries themselves can be either low- or high-level depending on the user. LiveClips mostly focuses on lower-level examples, but one of our three prototypes explores a potential way to measure task-level similarity between the user's actions and the examples clip's source video.

5.5 LiveClips System Overview

LiveClips takes as input a live streamed video and telemetry data for the software usage in the video. It extracts short clips showing bite-sized chunks of the artistic process, crops clips to thumbnail size, and ranks clips for recommendation based on their properties as well as the user's context. The current implementation of LiveClips extracts clips from a corpus of Making live streams, though we believe its approach can extend to other types of creative live streams as well. As our formative work showed, Making streams focus on the work being created, making them an appropriate candidate for exploring the potential of inspirational clips. In the current implementation, audio is removed to focus solely on the visual component of the videos, as our formative work showed that streamers often talk about irrelevant topics while working.

Our approach is tool-centric: each clip focuses on one tool and LiveClips recommends clips based on the user's tool use in the application. We focus on tools as a starting point, because

prior work has demonstrated that short tool-focused clips can be effective for contextual learning [77] and creative software tasks are often centered around the use of different tools. The next section describes LiveClips' three user interface prototypes, and the following section describes the backend system.

5.6 User Interfaces

Based on the design space outlined in section 5.4, we present three alternative methods for displaying and recommending the video clips generated by LiveClips in creative software (Figure 5.6). All three methods provide a link to the original video source underneath each clip, so that users can easily access it if desired. Each interface was implemented as a prototype HTML/Javascript extension to Adobe Photoshop.

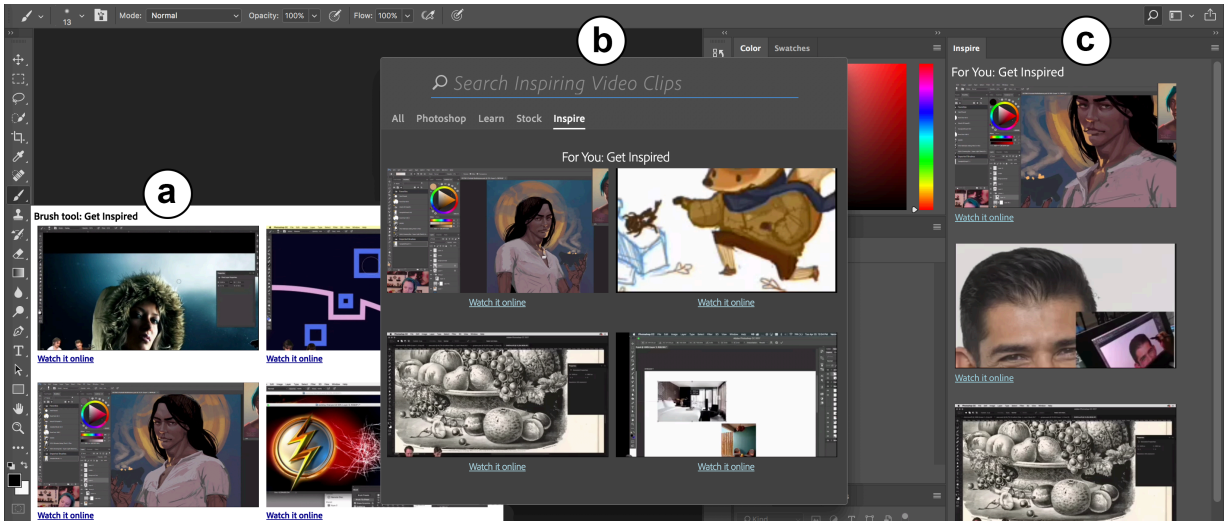

Figure 5.6: LiveClips' three interface prototypes demonstrate how video clips taken from creative live streams can be embedded in software as inspirational examples, implemented here in Adobe Photoshop. a) Contextual tooltips: When hovering over a tool icon, clips of that tool being used are shown. b) On-demand search: Pressing Photoshop's search button or Ctrl-F brings up an extended search interface with task-level clip examples. c) Ambient side panel: an always-visible panel updates periodically with clip examples based on the user's recent tool use.

5.6.1 On-demand Search

Many software applications provide in-application search, which allows users to search for application features, help, and/or recent documents. We augmented Photoshop's in-application search interface to present four clips when the search window is opened (Figure 5.6b). Clips are selected based on the user's tool use during the entire work session. Users can also search the entire library of video clips from this window. This interface is the closest in concept to RePlay and ReMap's interfaces. As such, we expect this interface to be most useful in moments where the user is stuck and wants new ideas. Each video is displayed as a thumbnail showing the first frame and plays when clicked.

This interface has low visibility, and updates only in response to an explicit user action. It uses higher-level context to recommend examples. Playback requires intentional mouse clicks.

5.6.2 Ambient Side Panel

Since creative applications tend to include many panels, a natural location for examples is in a side panel (Figure 5.6c). This interface explores examples that update ambiently as the user works. To avoid distracting the user during periods of focused work [34], the examples only update after the user has been idle for 10 seconds, indicating that they might be taking a break or trying to think of new ideas [213]. The panel shows four clips based on the user's recent tool use. Mousing over a clip plays it, providing a low threshold for interaction without distracting the user by playing video clips automatically.

This interface has high visibility, updates in response to implicit user actions, and uses mid-level context to recommend examples. Playback requires mouse movement without clicks.

5.6.3 Contextual Tooltips

For an interface that lies between an on-demand window and an always-visible panel, our third approach shows examples in tooltips that appear when the user hovers over a tool icon for 3 seconds (Figure 5.6a). Inspired by ToolClips [77], we believe tooltips may be a useful location for video examples. They can be activated by the user with minimal effort, are easily dismissed, and will not display at all if the user is working quickly or activating tools with keyboard shortcuts. Tooltips are a natural place for tool-centric examples; the video clips shown are based on the tool selected. Our implementation shows four clips when the user hovers over a tool, and clips play automatically when the tooltip appears.

This interface has mid-level visibility, updates on demand, uses low-level context (tools) to recommend examples, and requires no interaction for video playback.

5.7 LiveClips System for Generating Clips

The LiveClips system comprises three stages: 1) extracting short 25-second clips based on tool use from long videos, 2) cropping clips so that they can be displayed at thumbnail size, and 3) ranking clips for the three interfaces described above (Figure 5.7). The main goal for these clips is to expose users to what is possible in their creative software, and inspire them with new ideas and ways to use the software's tools.

Our approach uses a hybrid of telemetry (recorded usage data) and computer vision to understand and analyze the videos. We note that this could be accomplished entirely with telemetry (as in [78, 125]) by instrumenting the artists' software with detailed logging. Conversely, it could be accomplished entirely with computer vision, as prior work has shown that mouse movement and tool selection can be detected from videos alone [11, 184]. In our case (which is not uncommon), we have some usage data that is not fully detailed. Our approach aims to be a catch-all that allows working with a mixed or inconsistent set of data. LiveClips removes all

audio from clips during processing, focusing only on the visual component. This decision was motivated by our formative work finding that streamers talk about both relevant and irrelevant topics while working. While removing the audio risks removing potentially relevant narration, it avoids the challenges of properly segmenting narration that may or may not align with the work being shown.

5.7.1 Available Data

To develop and test LiveClips, we collected a sample dataset of creative live stream videos from the same two communities we analyzed in the formative work: the *Creative* category on Twitch and *Adobe Live* on YouTube. In an effort to make the sample as representative as possible, we collected videos from 17 different artists, and for two different software applications: Adobe Photoshop and Illustrator. Videos ranged in length from 30 minutes to 3 hours, giving us a total of 30 hours of video content from 17 different videos.

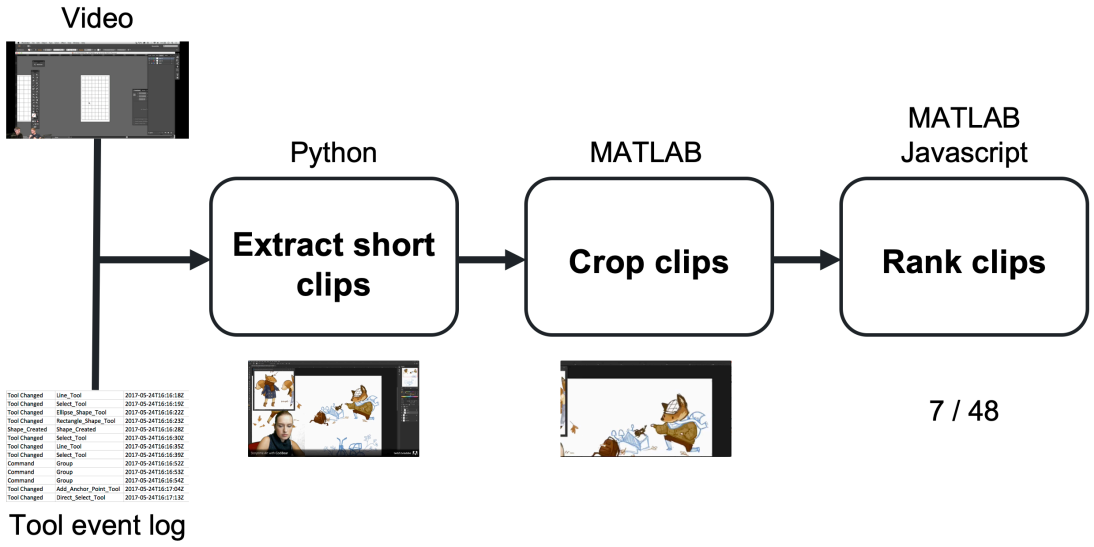

Figure 5.7: The LiveClips system takes as input a live streamed video and telemetry data for the tool usage in the video. It then: 1) extracts short 25-second clips based on tool use from long videos, 2) crops clips so that they can be displayed at thumbnail size, and 3) ranks clips based on their inspirational value and relation to the user's context.

All videos in our dataset were streamed either by Adobe (*Adobe Live*) or by an artist supported by Adobe (*Twitch Creative*). Therefore, for each video we had access to telemetry data produced by the artist's software at the level of tool usage, which includes time-stamped events for every selection and invocation of a tool (Figure 5.8). Mouse clicks and canvas manipulation details were not available. Our dataset included the use of 36 different tools from the toolbars in Photoshop and Illustrator.

5.7.2 Extracting Clips

To extract short clips, we use a heuristic approach based on work by Lafreniere *et al.* [125], who segment videos to create instructional clips of tool use. Our focus is on creating inspiring clips, which are different from instructional clips in that they require less attention to specific details (such as setting a tool's parameters) and more attention to the content being created. For the purposes of segmenting initial clips, our method is very similar to Lafreniere *et al.*'s method. We focus on inspirational value in the cropping and ranking stages.

Given a tool, our goal is to create clips showing its use. First, we group all consecutive invocations of the tool and include the tool selection event if it happened within 10 seconds of the first invocation. We add 2 seconds of padding to the beginning so that the viewer can see the tool being selected or invoked. We then trim clips to 25 seconds, as prior work has indicated that

Tool Changed	Line_Tool	2017-05-24T16:16:18Z
Tool Changed	Select_Tool	2017-05-24T16:16:19Z
Tool Changed	Ellipse_Shape_Tool	2017-05-24T16:16:22Z
Tool Changed	Rectangle_Shape_Tool	2017-05-24T16:16:23Z
Shape_Created	Shape_Created	2017-05-24T16:16:28Z
Tool Changed	Select_Tool	2017-05-24T16:16:30Z
Tool Changed	Line_Tool	2017-05-24T16:16:35Z
Tool Changed	Select_Tool	2017-05-24T16:16:39Z
Command	Group	2017-05-24T16:16:52Z
Command	Group	2017-05-24T16:16:53Z
Command	Group	2017-05-24T16:16:54Z
Tool Changed	Add_Anchor_Point_Tool	2017-05-24T16:17:04Z
Tool Changed	Direct_Select_Tool	2017-05-24T16:17:13Z

Figure 5.8: A sample excerpt of the time-stamped usage data we had for each live stream video.

15-25 seconds is a desirable length for in-application video clip recommendations [125]. We also ignore the use of navigation tools (*e.g.*, zoom, scroll, select) as these tend to happen frequently in visual software as part of other tasks. We shorten clips where the context likely changed before it was over, *i.e.* if a document was closed or opened. If a clip is shorter than 15 seconds, we remove it.

This heuristic approach sometimes fails. For example, live streaming artists often work meticulously with one tool for long periods of time (*e.g.*, a brush tool to create artwork), making incremental changes that over time create a more drastic and impressive change. The first 25 seconds of this may not be very inspiring, so we also explore an alternative method for creating clips that brings the focus away from the specifics of a tool's use, and toward the content being created: for instances where a tool is used consecutively for longer than 25 seconds, we extract the entire section where that tool is used, and speed it up to be 25 seconds long. We refer to these as timelapse clips.

In our sample set of 30 hours of video, the above two methods combined produced 1,727 clips, 484 of which were timelapses. This is comparable with Lafreniere *et al.* [125], who generated approximately 2500 clips from 25.5 hours of footage. LiveClips generated between 1 and 939 clips for each of the 36 tools in our telemetry dataset.

5.7.3 Cropping Clips

To present examples within an application, they have to be easily viewable by the user. A main design challenge with contextual clip recommendations is the space constraint: prior research has shown that in-application video clips should be unobtrusive and not take up too much of the user's screen [77]. Since live streamed videos typically include the artist's entire screen, simply resizing them to a small thumbnail size will make most of the interesting detail hard to see. Existing methods for cropping videos to a good thumbnail size include cropping it to only the region of the document that changes in the clip [78] or cropping to only relevant

UI regions [40]. While more animated effects such as "pan and zoom" might be appealing for providing both context and detail, Chi *et al.* [40] found that too much animation is disorienting for brief video clips.

Since prior work has shown that visible change is an important factor affecting the usefulness of a video clip [125], LiveClips attempts to crop each video clip to the area that changes most in the clip, so that the viewer's attention is drawn to the changes that are happening. For inspiration, we are mainly interested in visual change on the canvas. However, there are many other types of visual change that happen in these videos, such as switching windows, opening dialogs, and zooming in and out, and movement of the streamer in the webcam view of their face. Some of these changes (*e.g.*, zooming) are simply uninteresting, and others (*e.g.*, setting parameters in a dialog) may be helpful in a tutorial but are less interesting for short inspirational clips. LiveClips therefore excludes these types of change before identifying the area that exhibits the most visual change. This leaves canvas changes and other UI changes. Although we are primarily interested in canvas changes, we leave the task of separating these from UI changes to future work.

To filter out these "uninspiring" visual changes, LiveClips uses computer vision to exclude visual change caused by navigational movement, changing application windows, and the artist moving in the webcam view. To detect navigation events, LiveClips does simple feature detection and point feature matching in MATLAB across consecutive frames to identify and exclude frames where there is a significant amount of motion (likely due to panning or zooming). To detect application window changes, LiveClips identifies moments where more than 90% of the pixels change between two consecutive frames, and ends the clip before this change occurs. To detect change caused by the artist's webcam view, LiveClips uses face detection in MATLAB to locate faces that are consistently present in a bottom corner of the screen throughout the video, and mask out the corner containing those faces. The bottom corner is the customary place for streamers to place the webcam view of their face, and all the videos in our formative analysis as well as our

implementation dataset either did not show the artist's face or showed it in a bottom corner, so this is a reasonable assumption for creative live streams.

To determine the area of most change after filtering out uninspiring changes, LiveClips first calculates the pixel-wise difference between each pair of consecutive frames in the video clip. It then computes the average difference over all of these difference frames. There are many changes around the edges of a clip where artists open menus and panels, but these are not very interesting. The interesting changes are on the canvas, which is typically in the middle of the screen. Thus, LiveClips trims off 100 pixels from all four sides. Next, LiveClips further trims off all sides where the pixel values in the average difference frame are less than a given threshold (which we set to 1/4 of the maximum value in the frame). This allows us to avoid specifying a desired crop size, since some clips may be already zoomed in on the part of the canvas being changed, requiring minimal cropping, whereas others may be zoomed out, requiring substantial cropping to highlight the changing area. Figure 5.9 shows an example of the average difference frame from a clip and the crop that results from it.

5.7.4 Ranking Clips for Recommendation

The set of candidate clips generated is much too large to be useful, as only a few videos will eventually be shown to the user at any given time. As Lafreniere *et al.* [125] found, over 50% of automatically selected clips are of poor quality, so choosing good clips from the candidate set is an important step. This section outlines the criteria LiveClips uses to rank clips.

Time in the original video

Live streamed videos are several hours long and often show an entire project happening from start to finish. The closer an artist is to the end of their project, the more finished content they are likely to have on their canvas, and thus the more inspiring it is likely to be. For each clip, LiveClips divides the start time of the clip by the total length of the video to obtain a number

between 0 and 1 indicating how far along in the video the clip occurs. Larger numbers are better because they indicate that the clip occurs closer to the end.

Amount of visual change

Lafreniere *et al.* [125] found that showing clear visual change in short clips was most closely correlated to user preference. To determine how much visual change a given clip shows, LiveClips takes the cropped difference frame generated in the previous stage (Figure 5.9), and averages it across the *x* and *y* dimensions to obtain a numeric value representing the average amount of change in that clip (a larger number = more change).

User context

User context is the final factor for selecting which video clips to present to the user. This metric varies in each of our three prototypes, to explore different levels of user context. LiveClips' current implementation uses tool use to measure context. LiveClips first calculates a "total" score

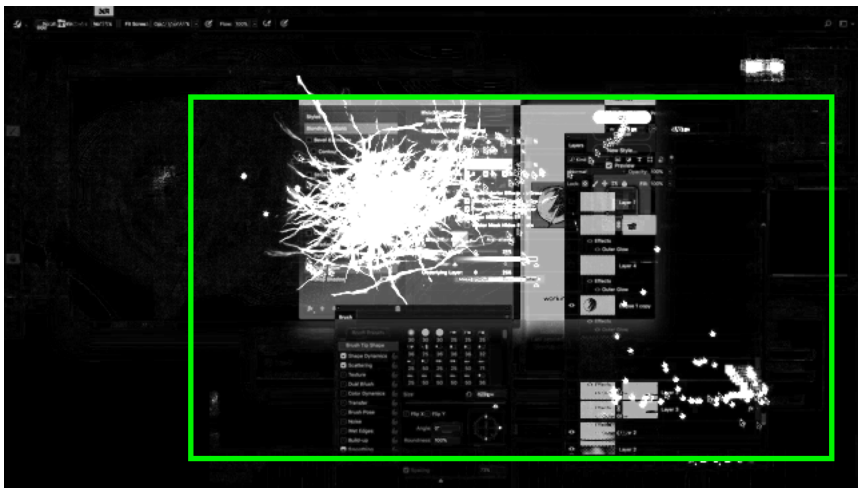

Figure 5.9: An example of a clip's visual change averaged across time (with change caused by navigation, application window switching, and the artist's face moving removed). From the brightness values of the pixels, we can see that the artist opened some dialogs, and drew some lightning strokes. The green box shows how LiveClips crops the clip to focus on the area that is changing.

for each clip by normalizing the time and visual change scores from above to be between 0 and 1, then taking the average. (For simplicity, we weight the two metrics evenly.)

Low-level context: The contextual tooltips interface uses low-level context to recommend examples; video clips are selected based on the tool that the user hovers over. For a given tool, LiveClips orders all clips of that tool by total score, then picks the top four clips. LiveClips requires that all examples come from different source videos, to ensure a variety of content. If some of the top examples are from the same original video, the system goes down the list until it has a set from four different source videos.

Mid-level context: The ambient side panel uses mid-level context to recommend examples; video clips are selected based on the last four tools the user has used. For each of those tools, LiveClips chooses the clip with the highest total score. If some of these clips are from the same original video, LiveClips instead picks the next top clips that are from different videos, to ensure variety.

High-level context: The on-demand search interface uses high-level context to recommend examples; the user's entire session of tool use is taken into account as well as the entire video from which each clip was extracted. For each full-length video, we store the overall percent of time spent using each tool, and compare this with the overall percent of time the user has spent using each tool in their current session. To measure the "difference" between a video and the user's session, we sum the absolute differences between percentages for each matching tool (each is a number between 0 and 1), and for every tool used in the video that the user has not used at all, we add 1 to the sum. The video with the smallest difference sum therefore represents the closest match to the user's task. LiveClips chooses the live stream videos with the four smallest difference sums. From each video it picks the clip with the highest total score.

5.8 Evaluation

To evaluate whether our ranking metrics are effective at finding inspiring clips, we conducted a study on Amazon Mechanical Turk. This study did not evaluate clips in the context of creative software. Instead, it focused on whether timing and visual change are good general predictors for inspiring content.

As described in the System section, LiveClips generated 1,742 clips from our dataset of 30 hours of video of Photoshop and Illustrator use. We randomly sampled 129 of these clips to include in our evaluation set. To ensure we had coverage of all tools in our dataset, we randomly chose between 2 and 10 clips per tool. Since some tools are much more popular than others (*e.g.*, Photoshop's brush tool had 939 clips whereas the burn tool only had 2 clips), this method allowed us to have a smaller sample while still representing every tool.

Workers compared video clips in pairs. Each Mechanical Turk task started with an overview page describing the task and showing an example pair of clips followed by five rounds of paired comparisons. Each comparison involved viewing two clips of the same tool and answering which clip they think would inspire them to be more creative (Figure 5.10). After completing all five comparisons, workers completed a short survey asking about their experience with Photoshop and Illustrator, and what types of creative tasks they do. Workers' results were only included if they completed all of the above steps. In an effort to ensure that they actually watched the videos, each comparison page only allowed workers to answer the question once both videos had played through once. Workers were paid $1.50 per task, which took an average of 10 minutes and 20 seconds to complete. The same worker could do multiple tasks and would see five new pairs of videos (randomly selected) every time.

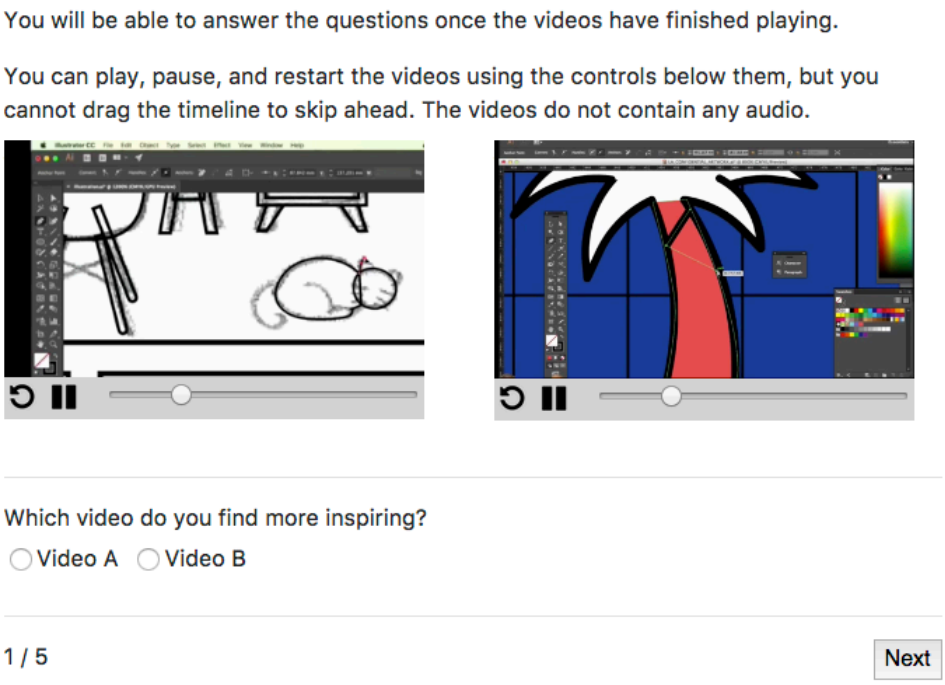

Figure 5.10: An example of a pairwise comparison completed by Mechanical Turk workers. Each task involved five such pairs. In the introduction to the task, workers were asked to consider which 25-second video would inspire them to be more creative.

5.8.1 Results

In total, 481 workers completed 628 tasks, giving us a total of 3,140 paired comparisons. Most pairs were compared by 9 unique workers, though some ended up being compared more times. To balance results across clips, we considered only the first 9 comparisons for each pair. Table 5.4 shows the distribution of workers' prior experience with Photoshop, Illustrator, and various types of creative tasks. Notably, 73% and 35% of participants had at least some experience using Photoshop and Illustrator respectively.

Since only clips showing the same tool were compared, our analysis only examines ranking agreement within tool groups. Each tool group had between 2 and 10 video clips. For groups with only 2 clips, we determine the human ranking of these two clips by ranking whichever clip was chosen more often first, and the other second. For all other groups, we use the Bradley-

Table 5.4: The distribution of workers' experience with Photoshop and Illustrator, and the types of creative tasks they have experience with.

		# Participants
Photoshop experience	None	130 (27%)
	Beginner	242 (50%)
	Intermediate	91 (19%)
	Expert	18 (4%)
Illustrator experience	None	311 (65%)
	Beginner	114 (24%)
	Intermediate	46 (9%)
	Expert	10 (2%)
Creative experience	Physical drawing/painting	247 (51%)
	Digital drawing/painting	147 (31%)
	Photography	386 (80%)
	Photo editing	290 (60%)
	Design	144 (30%)

Terry model as implemented in [154] to infer a ranking of clips within that group based on the paired comparisons. Within each group, we use Spearman's ρ to compute the correlation between human rankings and LiveClips rankings. LiveClips rankings are based on the total score for each clip (time and visual change combined).

Overall agreement between rankings ranges between very weak and moderate. Since the groups have different sizes, we cannot compute one overall measure of correlation, as it is unclear what the null distribution would be. Table 5.5 (top) shows the average ρ values for each group size. Note that p-values are not appropriate for such small group sizes, however the ρ values still accurately represent the correlation between rankings. Figure 5.11 (left) shows an aggregate comparison of all clip rankings.

Agreement between rankings for less-disputed clips ranges between weak and very strong. The main goal of LiveClips' ranking is not to establish an overall ranking of *all* clips, but rather to ensure that bad clips are discarded, and that only the best clips are shown to the user. Therefore, any clips that had a large amount of disagreement between workers are likely not among the best. Indeed, if we restrict our selection of video clips to only those that either won or lost over 65% of all comparisons they were a part of, this set consists of video clips that

Table 5.5: Average Spearman ρ correlation between human ranking and LiveClips ranking for all clips (top) and all clips with >65% agreement among turkers (bottom), compared within each tool. "# Groups" for column *i* refers to the number of tools that have *i* clips. Positive ρ values indicate a positive correlation, with correlations above .2 considered weak, above .4 considered moderate, and above .6 considered strong.

All clips (n = 115)

# Clips	2	3	4	5	6	7	9	10
# Groups	18	5	2	1	1	1	2	2
Mean ρ	0.11	0.3	-0.2	0.6	0.09	0.54	0.21	0.48

Clips with >65% agreement (n = 64)

# Clips	2	3	4	5
# Groups	18	3	3	1
Mean ρ	0.33	1.0	0.73	1.0

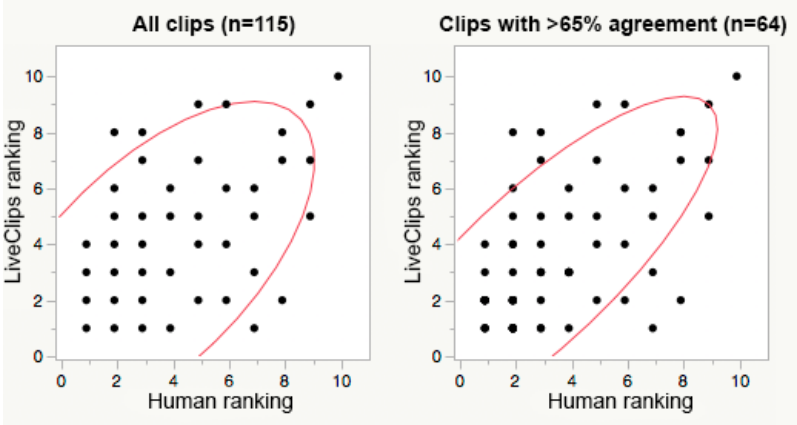

Figure 5.11: Human ranking vs. LiveClips ranking for all video clips (left), and all video clips with >65% agreement among turkers (right). Note that there is more overall agreement between rankings in the subset on the right.

likely have a more obvious objective value. Table 5.5 (bottom) shows the average ρ values for each group size when restricted to this smaller set of clips (in our sample set, 64/115). Overall we see stronger correlations. Figure 5.11 (right) shows an aggregate comparison of all these clips' rankings; we see fewer large disagreements here than in Figure 5.11 (left).

Timelapse clips did not perform significantly better or worse than standard clips. However, there were far fewer timelapse clips than standard ones in our dataset (19 vs. 96), due to the fact that longer sequences of a single tool use were less common overall than shorter sequences.

A larger dataset is needed to determine whether or not timelapse videos may be preferred over standard ones.

5.9 Early User Feedback

The Mechanical Turk evaluation explored the viability of LiveClips' ranking algorithm outside the context of an application. To get some initial user feedback on placing inspiring examples *inside* an application, we recruited two casual Photoshop users, both male, to work on projects of their choice. They used Photoshop with all three interfaces enabled at once (on-demand search, ambient side panel, and contextual tooltips) (Figure 5.6), and gave feedback while they worked.

Overall feedback was positive. Both participants found seeing video examples in the application useful and said that they wanted to see how others use tools and set parameters. The two participants differed in how they wanted to see examples. One participant preferred the tooltips, explaining that they were a nice level in between the on-demand search (which was easy to forget about) and the ambient panel (which they found distracting). They liked that if you happen to hover on a tool for an extra moment (which could happen by accident), it pops up and serves as a reminder that these examples exist, without taking away too much from the process. The other participant preferred the panel, and wanted to be able to refresh recommendations on demand and see a large variety of content. Both participants clicked on the videos and wanted to watch them in the browser where they could see them larger.

This early user feedback is encouraging and shows the opportunity around embedding inspirational examples in creative software.

5.10 Discussion

5.10.1 Can We Really Predict What Will Be Inspiring?

The results from the ranking evaluation presented in section 5.8 should be taken with a grain of salt. The workers who participated in this study were not all users of creative software, and the video clips were presented out of context. The evaluation was intended as an initial baseline to determine whether LiveClips' ranking algorithm can reasonably identify inspiring clips. The subjective nature of the question workers were asked meant that we could not include a "ground truth" comparison to filter out "lazy turkers". We did however enforce that the entire video clips played at least once before workers could select an answer.

Aside from the questions that using crowd-worker participants raise, the idea of "inspiration" is subjective and hard to predict. What one person finds inspiring another may not. Whether someone finds something inspiring could even change depending on the time at which they see it. Despite these challenges, we had reasonable agreement among workers overall; the average percent of agreement across all pairs of clips was 74% (SD 14.5). Even if Mechanical Turk workers *were* a 100% reliable source, we would still expect some disagreement, due to the subjective nature of the question.

The LiveClips algorithm can reasonably identify clips that most workers agree are either good or bad. It is important to keep in mind that the main purpose for this algorithm is to address the "cold start" problem. Once people start using an interface with video examples, LiveClips could continually improve the ranking algorithm based on user behaviour and preferences, exhibited through how much people interact with the videos.

5.10.2 How Diverse Should the Set of Examples Be?

Research on examples and creativity is rather divided regarding whether diverse examples that are distantly related to the user's task are better than a narrow set of examples that are more

closely related to the user's task. Some work has shown that more diverse, far-off examples improve creativity by encouraging people to think more broadly and try new things [33, 212], while other work has shown that similar examples are more useful as they are more relevant to the user [32]. More recently, Benjamin *et al.* [16] propose letting the user decide by providing an adjustable slider that determines the diversity of recommended examples, based on the idea that the need for more diverse or more similar examples may differ depending on where the user is in their process.

Another option could be to let users search by example, a feature that is now common in search engines. Rather than having to specify an exact query, users could provide an example video and ask for "more like this". However, even in this case the diversity of results is an important factor to consider.

5.11 Limitations & Future Work

This chapter presented initial work exploring ways to bring inspiring examples into the creative process, and a new type of content from which to draw such examples: creative live stream videos. We provide suggestive results that our algorithm can select potentially inspiring clips, and some initial user feedback on our prototype interfaces indicating that this is a promising avenue for future work. Rather than conduct a rigorous controlled study (which is difficult and often inappropriate for evaluating goals like creativity and inspiration [210]), this chapter aims to introduce this space and lay out the possibilities for future work to explore. In this section, we discuss a few main directions for such work.

5.11.1 More Robust Segmentation and Change Ranking

Our current methods for detecting visual change use simple computer vision approaches, and exhibited some failure cases. We have yet to try more sophisticated deep learning methods

to segment this data but this is a promising direction of future work that could help improve the classification of panning, zooming, and parameter setting. Having more robust and available usage data could also improve the detection of such actions. For example, Lafreniere *et al.* [125] used instrumented software to gather both videos and detailed usage, which allowed them to calculate visual change directly by counting the number of pixels on the artist's document that change during a video clip. As of recently, streamers on Behance (`behance.net/live`) can use a plugin that logs their tool use while they stream in creative software. This data can be used to segment live stream archives [62], but such segmentations would still benefit from additional visual analysis to identify events that are not captured in usage logs and understand visual properties of the artist's work. Finally, to make use of the large amount of video content that already exists online with no available telemetry, a deep learning system could be trained on those videos that do have usage data, and then applied to videos that do not.

5.11.2 Making Use of Audio and Chat Logs

Live streamed videos come with additional data that this work did not make use of: audio in the form of the streamer's narration and responses to questions (which can be transcribed to text) and chat logs from viewers of the stream. Making use of audio narration in software tutorial videos is an open problem [40] as narrations don't always align exactly with the artist's actions. Live streamed videos have a similar problem that is exacerbated by the fact that artists are not always narrating what they are doing; as our formative work found, sometimes streamers answer questions from the chat and other times they talk about unrelated things as they work. Future work should explore ways to detect when those different types of narration are occurring, and make use of their content as appropriate, for example highlighting moments where artists answer questions, or building a mapping of ways in which artists describe their software actions in natural language.

Chat logs may also be a useful data source to include in future work. Though the content

of live chats tends to be very noisy, especially on popular channels [83], the frequency of chat posts at a given time can indicate exciting or interesting moments [174]. Chat post frequency could therefore be used as another criteria for ranking clips. In addition, our formative work found that questions asked in the chat often go unanswered, usually because the streamer does not see them. Finding and highlighting such questions after the fact and encouraging the streamer (or other viewers) to answer them later could help keep viewers engaged once streams are archived (which several viewers in our formative surveys desired), as well as surface valuable feedback and ideas for the streamer.

5.11.3 Other Ways to Generate Short Clips from Long Videos

This work focused on generating tool-focused clips, inspired by prior work [77, 125] and the natural mapping between tool use and user behaviour. However, while tool-focused clips are good for showing users how a tool works [77], they may not be the most inspirational types of clips one can generate from long videos. Other methods could include breaking videos into higher level sub-tasks (based on pauses in tool activity and narration content), or extracting clips where the artist describes a particular technique or answers a question. As one of the exciting parts of watching a live stream is seeing an artist go from a blank canvas to a finished work, another option could be to shorten the entire video down to a short summary. This could be done by removing sections where the artist takes pauses, talks with no actions, and switches applications; and by adapting existing techniques for creating short summaries from long videos (*e.g.*, [227]).

5.12 Conclusion

This chapter explored a growing form of creative videos very different from traditional tutorials: live streams. We found that creative live streams are a good source of both learning and inspiration, and introduced an approach for bringing inspiration into the context of creative

software users' workflows.

RePlay, ReMap, and LiveClips all leveraged **visual media** in the form of screencast videos for providing contextual support towards understanding creative processes. But for users who just want to get a task done without worrying about the process behind it, videos can be too tedious and detailed. The next two chapters explore how two different types of resource – **executable code** and **written text** – can help people quickly and easily achieve creative outcomes.

5.13 Acknowledgements

We thank Tricia Ngoon, Kandarp Khandwala, and Nicolas La-polla for their help with live stream analysis, and our study participants for their insights. This work was supported in part by NSERC, Adobe Research, and NSF award #1735234.

This chapter, in part, includes portions of material as it appears in *Sharing the Studio: How Creative Livestreaming can Inspire, Educate, and Engage* by C. Ailie Fraser, Joy O. Kim, Alison Thornsberry, Scott Klemmer, and Mira Dontcheva in the Proceedings of the 2019 on Creativity and Cognition (C&C '19). The dissertation author was the primary investigator and author of this paper.

This chapter, in part, includes portions of material coauthored with Andy Edmonds and Mira Dontcheva. The dissertation author was the primary investigator and author of this material.

Chapter 6

DiscoverySpace: Suggesting Actions in Complex Software

When using complex software for the first time, sometimes even video assistance can be too overwhelming. RePlay and ReMap found that people don't always know what help to search for to achieve their goal, and novices may not even know what goals are possible. LiveClips proposed contextual examples to help users explore potential outcomes, but it does not help them *achieve* these outcomes. Even when following tutorials, novices often get lost in the many steps required to achieve their goals. To address these challenges, this chapter introduces *DiscoverySpace*, a contextual panel for Adobe Photoshop that suggests action macros to apply to photographs. DiscoverySpace harvests these one-click actions from the online Photoshop user community. A between-subjects study indicated that action suggestions may help novices maintain confidence, accomplish tasks, and discover features. This work demonstrates how interfaces can leverage user-generated content to help novices achieve expert-level outcomes in complex software.

6.1 Introduction

Accomplishing one's goal with software generally requires composing multiple operations into a sequence. The burden is on the user to know which operations to combine and how to execute them. This work aims to reduce the execution gap between users' high-level task goals and the low-level steps needed to achieve them, by suggesting action macros. These aggregated

action suggestions move the user's focus from individual *operations* to human-understandable, goal-driven *activities* [64, 137].

We investigate the efficacy of action suggestions through DiscoverySpace (Figure 6.1), a recommendation interface for action macros recorded by the user community. Action macros encapsulate a sequence of operations that is executed as a batch, and are widely shared online[1]. The DiscoverySpace prototype is a Photoshop panel comprising action macros that can be applied in one click for quick and easy exploration. We hypothesize that action suggestions help users

[1] A Google search for "free photoshop actions" brings up over 39 million results, such as `creativebloq.com/photoshop/photoshop-actions-912784` and `fixthephoto.com/free-photoshop-actions`.

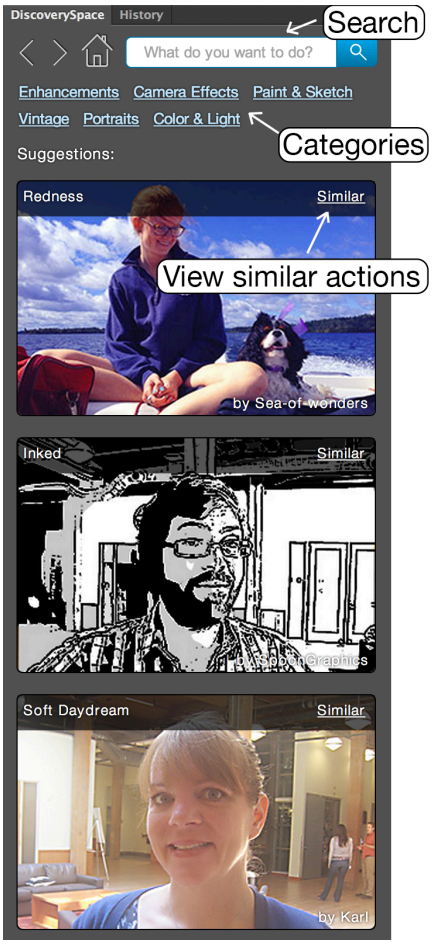

Figure 6.1: DiscoverySpace as a panel in Photoshop. Users can apply a suggested action by clicking on its image.

get started in complex applications by enabling them to easily explore creative possibilities and achieve quick results.

This chapter contributes a prototype action suggestion system, DiscoverySpace, and the results of a preliminary experiment to examine its efficacy. This between-subjects study found that action suggestions may help prevent novices from losing confidence in their abilities, and help users to accomplish tasks and discover new features. This chapter also proposes design guidelines for a suggestion interface based on these observations and results.

6.2 Related Work: Helping Novices Use Software

The literature offers several strategies for helping novices effectively use software: some focus on high-level tasks, others on lower-level tool use. Figure 6.2 describes how our approach relates to this prior work.

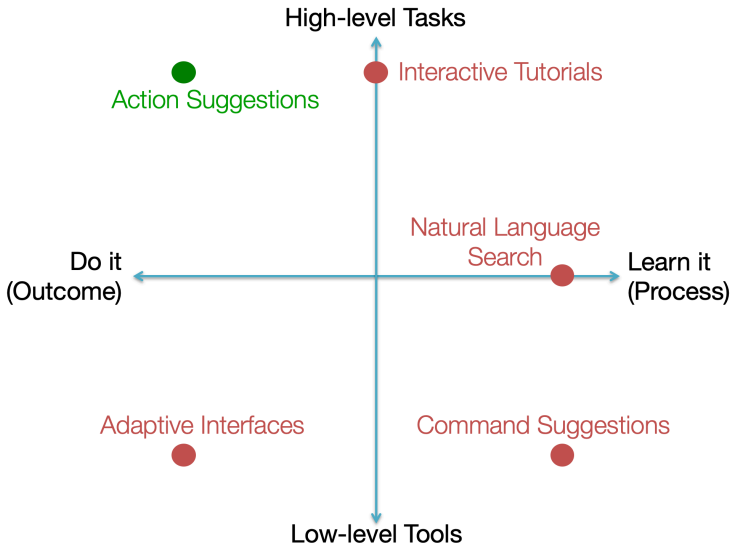

Figure 6.2: Two dimensions along which interventions to help novices can vary. *X-axis*: the goal of the user (whether they want to get things done quickly, or learn the application and its tools). *Y-axis*: the approach to assisting the user (providing high-level tasks or low-level tools).

6.2.1 Interactive Tutorials Provide Step-by-step Guidance

Online tutorials are a popular resource for users of complex software. However, they present difficulties such as switching back and forth between the browser and the application, and mapping screenshots or videos of the application to the user's own version [106,184]. Interactive tutorials address these challenges by guiding users step-by-step through hands-on example tasks inside the target application [106, 123, 184]. Such tutorials can even be generated automatically as a user demonstrates them [73]. However, there are still far more user goals than authored tutorials, and automatically making static tutorials interactive remains a challenge [61, 126]. In addition, users must translate between their own goals and the available tutorials.

Tutorials teach users to use applications step-by-step. Other systems focus on speeding up holistic tasks rather than teaching individual steps. For example, TappCloud introduced "semi-automated" tutorials that can be applied in one step, like macros, but also allow for the user to interact with each step if desired [126]. However, TappCloud users do not interact directly with the application's tools, but instead with visual previews of different parameter settings. DiscoverySpace also focuses on smoothing the execution process of achieving outcomes rather than helping users learn the application.

6.2.2 Adaptively Disclosing Interface Functionality

Adaptive interfaces initially provide a simplified interface and progressively disclose additional features either automatically [63, 155, 179], or by allowing users to move between predefined interface stages [30, 132, 155, 207]. Adaptive interfaces are most successful when they are controllable, predictable, efficient, and promote feature discovery [156,209]. However, achieving all of these is a well-recognized challenge [54, 155]. Interfaces that disclose or move features automatically based on user behaviour (*e.g.*, adaptive toolbars in Microsoft Word [63]) can improve efficiency, but often feel unpredictable, uncontrollable, and distracting [63, 179, 209].

Photoshop Elements is an example of an interface with predefined stages: quick, guided

and expert. Users can choose a mode appropriate for their task and skill level, such as quick mode for easily accomplishing basic edits. However, because each mode is associated with a different level of complexity, the interface layout and grouping of tools differs between the modes, making it difficult to transition from one to another.

Adaptive interfaces have mainly focused on task completion efficiency as the performance metric [63, 132]. However, for open-ended creative tasks, speed is often less important than discoverability and final quality. Progressive disclosure impedes discoverability by hiding interface elements [54]. DiscoverySpace addresses discoverability by suggesting actions users may not have known to be possible.

6.2.3 Command Suggestions and Previews

Most software users are only aware of a small percentage of software features [156]; there may be potential results they could achieve but do not think to try. CommunityCommands introduced the idea that recommending commands to users can improve an application's discoverability [136, 153]. It uses collaborative filtering to suggest AutoCAD commands that are unknown to the current user but frequently used by others in tandem with the current user's frequent commands. Though command recommendations improve discoverability of new features [136], suggesting individual commands still requires the user to compose and apply them. Inspired by this, DiscoverySpace uses suggestions to promote exploration and discovery, but in the form of task-level actions rather than individual commands.

Previewing results helps users predict what a command will do without having to execute it first. Side Views demonstrated that previews are especially valuable when shown as multiples with different parameter settings [223]. Building on this, suggesting and previewing alternative courses of action provides users with ideas they may not have considered, speeds the iterative design loop, and shows the contextual effects of changes. Inspired by DesignScape's suggestion interface for graphic design layouts [170], DiscoverySpace provides both minor (refinement) and

major (radical) suggestions.

Popular recommendation algorithms, like collaborative and content-based filtering [180], rely on users' behaviour and preferences as input. For suggestions within a user interface, the user often begins with a document, 3D model, or image. Suggestions should be content-dependent, since different documents will benefit from different types of operations. For example, DesignScape generates layout suggestions by altering existing elements in the document [170]. Since visual editing operations can have vastly different outcomes on different images [17], it is important to suggest effects that make sense for the content in question. Our algorithm therefore takes into account the content of the image to produce relevant suggestions.

6.2.4 Natural Language Search

To help users find functionality, applications like CommandSpace integrate search directly into the interface [5]. Natural language search reduces the gap between user language and application language [5]. Inspired by this work, DiscoverySpace includes a search capability in addition to automatic suggestions. DiscoverySpace also provides simple faceted browsing, the ability to refine a collection based on metadata characteristics [115]. Facets are especially beneficial for "exploratory searchers" who tend to have only a partial idea of what they are looking for [88]. Faceted search and browsing were designed for information retrieval tasks; DiscoverySpace extends this underlying idea to executable *actions*.

6.3 What Kind of Help do Novices Need?

To unearth novice-expert differences in creative software usage strategies, we conducted nine 30-minute sessions in which three expert photo editors helped seven novices edit their photos using Adobe Photoshop and Lightroom. Each session included one novice and one expert (Figure 6.3). The novice described their goals while the expert controlled the program and com-

municated with the novice to help achieve them. One-on-one tutoring like these pair discussions is highly effective for teaching [18], and consequently a valuable model for what software could achieve. We observed these interactions to see what novices asked, how experts translated these requests into image editing operations, and whether there were novice-expert language differences or communication challenges. Our intuition is that good software should enable novices to solve challenges like these for themselves. After all the sessions, we reviewed our notes and looked for recurring behaviour patterns. The following four insights were most prominent:

1. Novices often did not know what they wanted to do with a photo, and could not picture how it might be improved. In these cases, the experts would provide their own suggestions to get things started.

2. Novices had very high-level goals (*e.g.*, "I want to make this person stand out more"), and often did not know what tools or techniques were needed to accomplish them. Experts were able to translate these goals into concrete operations.

3. A common way to show a novice what an effect will do was to execute it on the photo and compare the result with the original photo, sometimes exaggerating the effect to illustrate

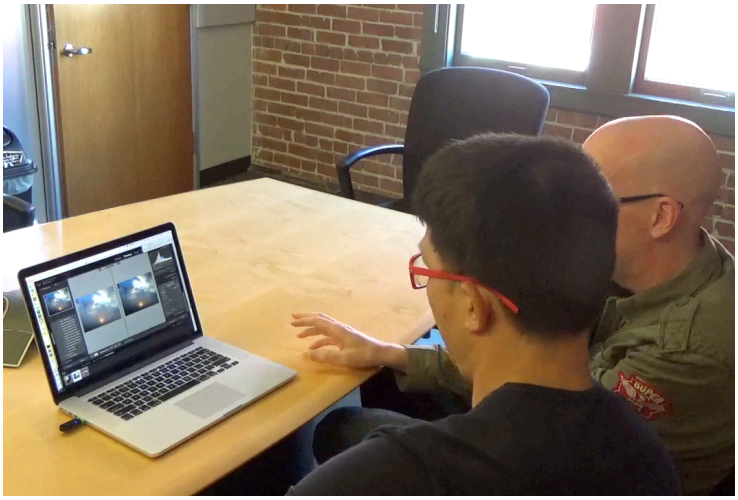

Figure 6.3: A novice-expert session. The expert (right) compares the edited photo with the original to illustrate what the edits did.

the difference before dialing it back down. Novices were sometimes hesitant about an expert's suggestion, but after seeing its effect on the image would become more excited about it.

4. Novices appreciated both suggestions that were relevant to their goals, *and* suggestions for different effects they would not have otherwise thought of.

6.4 Design Goals

Presenting suggestions relevant to the user's goal can help them accomplish it, as the experts did. Presenting suggestions the user might not have otherwise thought of can help them to discover what is possible and achieve creative results. Combining these insights with prior recommendation research (*e.g.* [88, 89, 115, 136]), we developed five main **design goals** for suggestion interfaces:

1. *Help users get started*: Make suggestions available as soon as the user begins a task. A frequent observation throughout our formative study and prototyping process was that users often did not know where to start in Photoshop.

2. *Use human language*: Allow users to search using goal terminology, and describe suggestions in a language novices can understand by including a descriptive non-technical title for each. This reflects the growing popularity of natural language or "semantic" search [89].

3. *Show previews* of what a suggestion does, and allow users to easily compare before and after, for example as in Side Views [223].

4. *Offer faceted browsing* in addition to search to help users explore. This is a well-documented concept in the information retrieval literature (*e.g.*, [88, 115, 238]), and we believe it to be applicable to software applications as well.

5. *Suggestions should be relevant* to the user's current task, but should also alert the user to new or unknown possibilities. CommunityCommands users were found to prefer "contextual" suggestions related to their short-term history [136], and users of complex software often desire the ability to discover new features [156].

We iterated on the DiscoverySpace design based on feedback from users with varying levels of Photoshop proficiency. Because DiscoverySpace harvested actions from multiple online sources, they had widely varying amounts of user interaction: some automatically applied effects, while others paused at key points with dialogs for user interaction. We removed such pauses and dialogs when present because they tended to lack sufficient accompanying explanation, and therefore confused some non-expert pilot testers. Users often desired basic adjustments such as brightness/contrast, exposure, and saturation. The actions in our collection generally comprised several steps rather than just one, and so did not include many of these basic edits. We manually created 11 actions for these basic adjustments, which would simply initialize the values to auto, and leave the sliders visible for the user to adjust as they wish.

6.5 DiscoverySpace: Action Suggestions

This section describes the DiscoverySpace interface (Figure 6.1) and its keyword-based suggestion algorithm.

6.5.1 User Experience

When a user first opens a photo, DiscoverySpace prompts them to enter a goal and select features their image contains from a list (*e.g.* "contains people", "outdoor"). Next, the DiscoverySpace home page appears, containing a search bar, category buttons, and suggested effects (Figure 6.1). Each suggestion is displayed by showing the effect applied to an example image with appropriate content. For example, the image for a "skin smoothing" effect is of a close-up face.

The user can mouse over the image to compare *before* and *after*. Clicking a suggestion applies that action to the photo. As given, the user has no control over the settings for each step of the action; they can see the sequence of steps that was done after applying it by opening the History panel (Figure 6.4), but cannot step through it or adjust the parameters of individual operations. These actions are therefore intended to allow users to quickly achieve an effect without needing to manually complete all of the intermediate steps. The user can easily undo or redo the effect once it has been applied. If the user scrolls down on the main page, more suggestions appear. To browse suggested effects in a specific category, the user can click a category button. To search the corpus of effects, the user can enter a query in the search bar. The user can also browse effects that are similar to a selected effect by clicking the "*Similar*" button on the effect.

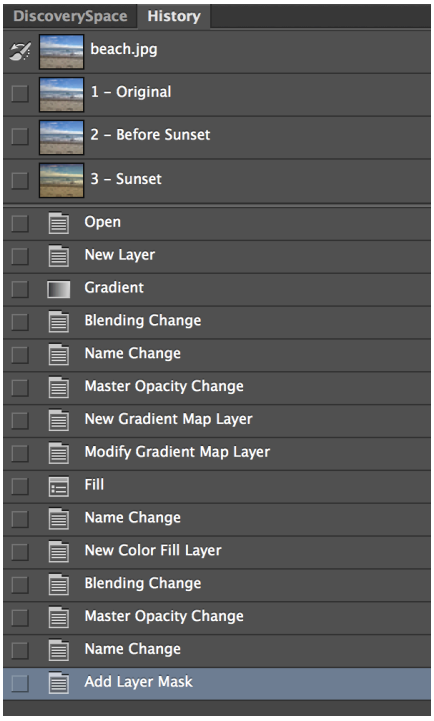

Figure 6.4: In the History panel, users can review the operations an action has performed.

6.5.2 Implementation

DiscoverySpace is a Photoshop extension panel, written in HTML, Javascript, and Adobe ExtendScript. It retrieves a manually-curated corpus of 115 actions stored on an Amazon S3 server, providing the flexibility to update the actions to reflect popular trends. We manually defined the action categories by reviewing and clustering free actions found online. We created the corpus by downloading 2-3 actions from each subcategory. Paid actions could be added in the future by allowing users the option to pay for effects from within the panel. DiscoverySpace automatically augments search queries with synonyms to maximize the number of relevant results.

6.5.3 Keyword-based Recommendations

DiscoverySpace recommendations take the user's photo as input (Figure 6.5). Each action has been assigned a number of descriptive keywords. When opening a new photo, users must select features that describe the image; image analysis could automate this step in the future. We map the selected features to potentially relevant keywords, and assign each action a weight based on the number of matching keywords.

To select and present suggestions, our algorithm randomly samples from the collection of actions based on these weights: actions with a larger weight have a higher probability of being selected. The randomness encourages discovery by presenting new suggestions when the user refreshes the page. Each action has also been manually classified as either *refinement* or *radical*. An action is deemed *radical* if it produces a drastic effect, does not necessarily maintain realism, and/or takes multiple steps to accomplish. Examples include painting effects, HDR, replicating old cameras, and drastic color effects. An action is deemed a *refinement* if it involves a small adjustment that is meant to improve the photo while maintaining realism, or could be accomplished in one step. Examples include brightening, softening skin, sharpening, and slight color casts. We prefer that both types of effects are suggested, to promote variety in the options. To ensure this, we run the prioritization algorithm separately on the two collections of actions, and

then, for every four actions that are suggested, we sample two from the refinement pool and two from the radical pool. This is because on a 27-inch desktop display, DiscoverySpace in its default position shows four suggestions at a time.

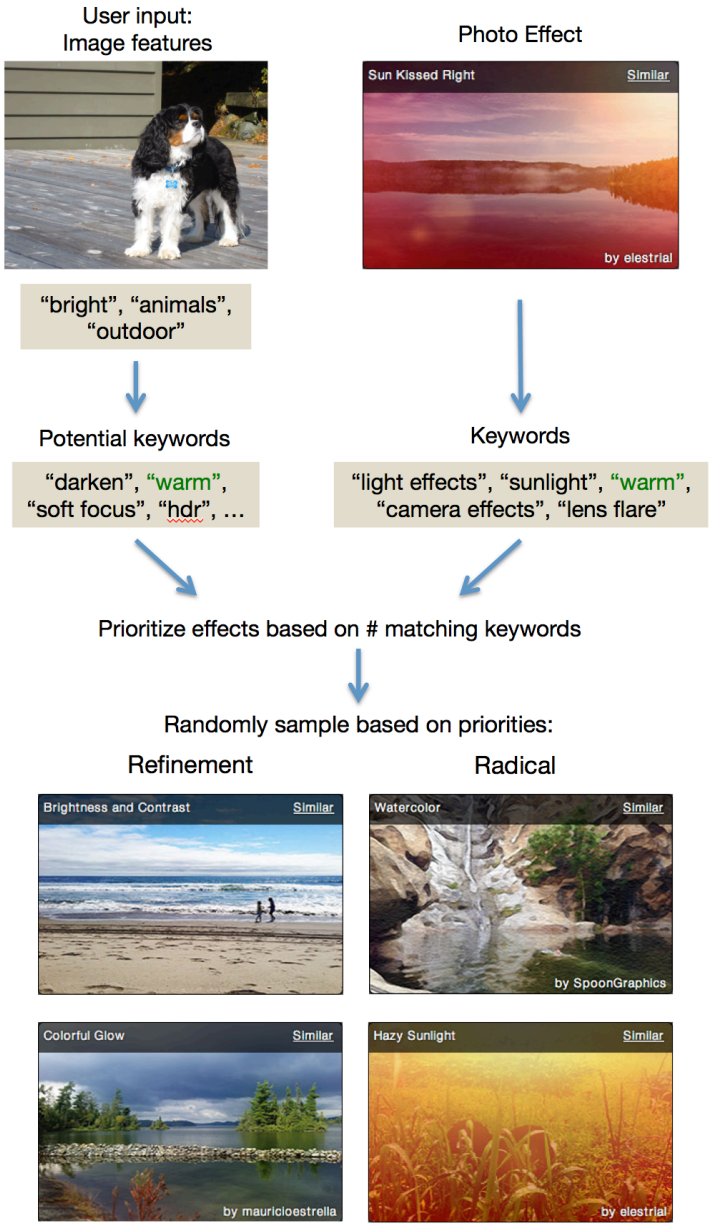

Figure 6.5: An overview of our keyword-based recommendation algorithm. User-selected image features are mapped to keywords and matched against the keywords for each action in the corpus.

6.6 Study

To investigate the efficacy of action suggestions, we conducted a between-subjects experiment in which participants edited their own photos in Photoshop. In the experimental condition, subjects had access to DiscoverySpace; in the control condition they did not. We hypothesized that participants exposed to the suggestions in DiscoverySpace would find the editing process easier, feel more creative, and become more confident using Photoshop.

6.6.1 Method

To sign up, participants completed a brief initial survey about their Photoshop experience. Participants were then scheduled for a 30-minute session and assigned to one of two conditions: participants in the DiscoverySpace condition used Photoshop with the DiscoverySpace panel open, and participants in the Control condition used Photoshop with a blank panel that instructed them to save their photo when finished (Figure 6.6). The study held constant the length of the session (30 minutes), the number of photos edited (two), the questions participants answered when opening and closing a photo, and the availability of a web browser (yes).

Each session comprised the following steps: consent form, background questions, edit first photo, edit second photo, short interview, and final online survey. The background questions allowed participants to elaborate on their initial survey responses regarding their prior experience with Photoshop and photo editing. Participants were instructed to bring one photo containing at least one person, and one photo without people, to ensure some consistency across participants' photos. The editing order of the two photos was randomly assigned with balancing in each condition. We chose to allow participants to bring their own photos, rather than provide sample photos, so that participants would be more likely to come up with their own goals for their photos, and would be motivated to do a good job.

In both conditions, upon opening a photo, the study panel prompted the user to describe

their goal for the photo and select its image features. The panel also prompted the user to answer a few questions about their editing experience every time a photo was closed (Figure 6.7). The panel was in a prominent location and the participants were made aware of it by the experimenter. The task was open-ended: participants could define their goal however they wanted and were simply instructed to work until they were satisfied (or until time ran out). We used an open-ended task as opposed to a more directed one because participants would be more likely to perform as they would in a real-world setting when working on personally meaningful tasks. In the DiscoverySpace condition, participants were given a quick overview of the DiscoverySpace interface and were shown what each button in the panel did, so that time would not be wasted figuring out how to use the interface. The History panel was located behind DiscoverySpace, and was only visible when clicked on. Participants in the DiscoverySpace condition were shown the History panel only if they asked explicitly how to go back in time by more than one action.

In both conditions, a web browser was open on the Google home page just behind Pho-

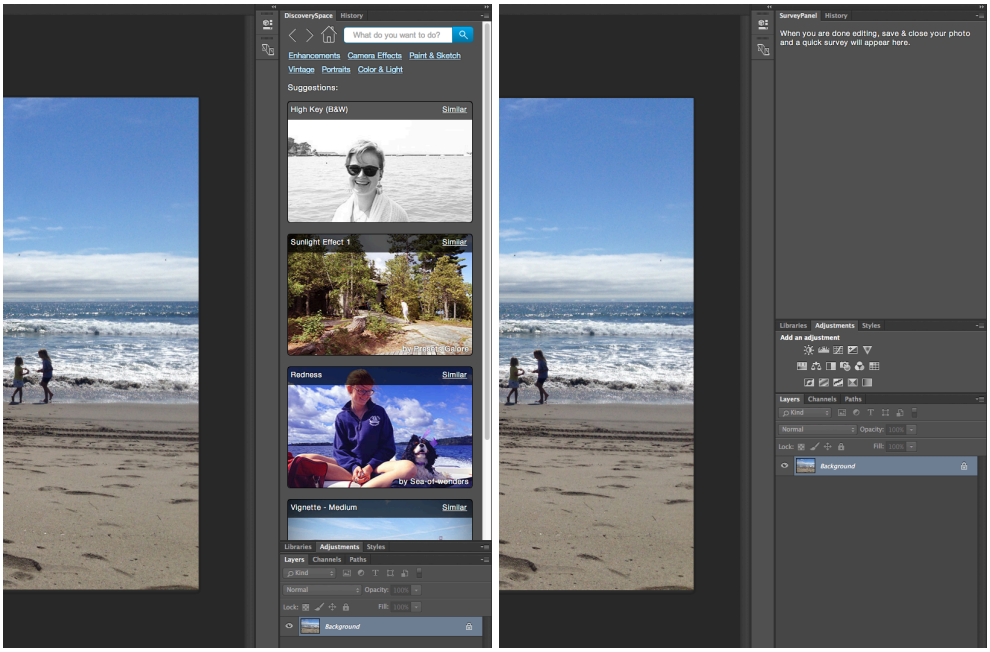

Figure 6.6: Photoshop as it appeared in the DiscoverySpace condition (left) and Control condition (right).

toshop; participants were told they could use it at any time. This was included to reflect the ready availability of web resources in creative workflows. A short interview and final survey took place in the last five minutes of the session. It comprised follow-up questions on the participant's overall editing experience and their perceived difficulties.

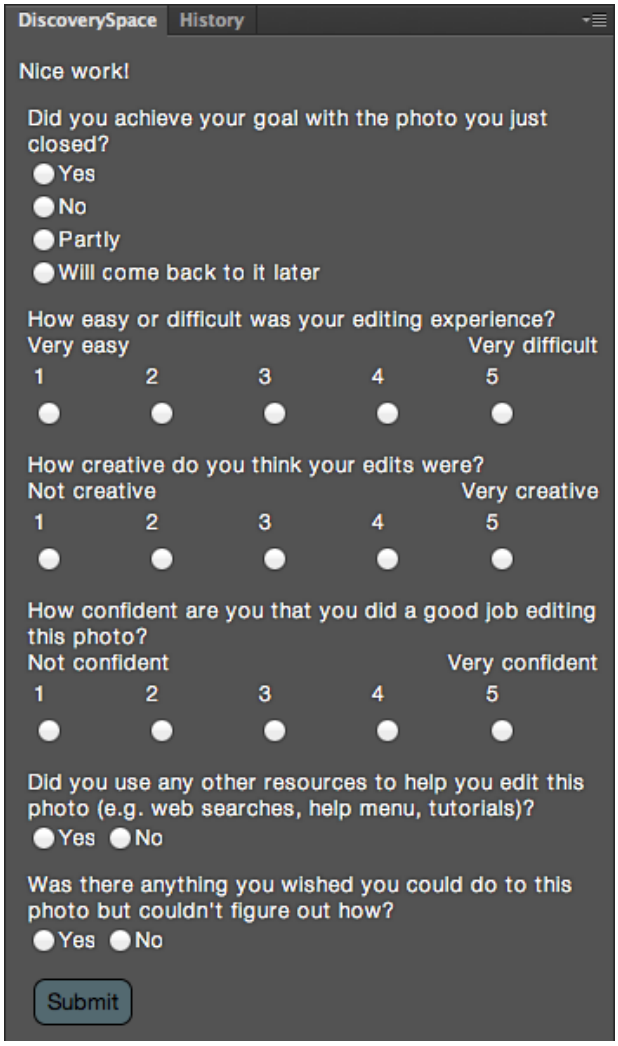

Figure 6.7: Questions answered by participants after editing each photo. The measure for "How confident are you that you did a good job editing this photo?" is referred to as "confidence about performance" in the Results.

Table 6.1: Participants' gender and Photoshop expertise.

	Female	Male	**Total**
Beginner	6	5	**11**
Intermediate	3	9	**12**
Expert	1	4	**5**
Total	**10**	**18**	**28**

6.6.2 Participants

28 students were recruited from four undergraduate classes at a Southern California university (one photography class, three design classes). None of these classes provided formal training in Photoshop, though many participants had experience with other photo editing software and/or basic photography principles. Stratified randomization was used to balance gender and Photoshop expertise across conditions. Expertise with Photoshop was measured as follows: in the initial survey, participants were asked to rate their skill level with Photoshop from 1 to 5, and their amount of experience with Photoshop from 1 to 5. The answers to these two questions were added together to produce a score between 2 and 10, and participants were grouped according to their score into Beginner (2-4), Intermediate (5-7) and Expert (8-10). See Table 6.1 for the distribution of participants' expertise and gender. 15 participants were assigned to the DiscoverySpace condition, and 13 to the Control condition (the uneven balance was due to the stratified randomization).

6.6.3 Measures

Both surveys asked participants to rate their Photoshop confidence by answering two questions: "How confident are you with your ability to edit photos in Photoshop? (1-5)" and "How confident are you that you can achieve your goals in Photoshop? (1-5)". We measured change in confidence as the difference between the sum of both responses (between 2 and 10) in the final survey and the initial survey. For each of the three Likert-scale questions answered after every photo (Figure 6.7), we added the responses from both photos to obtain one score between 2

and 10 for each participant ($N = 28$). For the questions with yes/no answers, we considered each photo separately ($N = 56$ as each participant edited two photos). We also recorded the number of Photoshop tools used (not including those from DiscoverySpace) and all clicks and pages loaded in DiscoverySpace.

6.6.4 Results

For the numerical measures, we used analysis of variance to analyze Condition, Expertise, and their interaction. For yes/no measures, we used logistic regression with the same variables.

Participant goals

As expected with such an open-ended task, there was a large variety of participant goals. The most frequent goals focused on color/brightness (*e.g.*, "correct exposure and color balance" or "experiment with color"), or a particular emotional effect (*e.g.*, "make it surreal looking" or "give a lonely, solemn feel"). As expected, participants more often (72% of the time) used non-technical terms to describe their goals rather than photography-specific terms such as "saturation" and "sharpness". This was most prominent among Beginners and Intermediates.

Quantitative results

Beginner participants in particular were significantly more likely to lose confidence in the Control condition ($M = -2.0$) and gain confidence in the DiscoverySpace condition ($M = 0.5$), $t(22) = -2.16, p = 0.04$ (Figure 6.8). Across all Expertise levels, DiscoverySpace had only a marginal effect on participants' confidence: the confidence of Control participants decreased ($M = -0.77$) while the confidence of DiscoverySpace participants increased ($M = 0.27$), but this was not significant, $F(1, 22) = 2.97, p = 0.10$. No effect was found for Expertise.

DiscoverySpace participants were significantly less likely to state that there was something they couldn't figure out ($M = 50\%$) than Control participants ($M = 80\%$), $\chi^2(1) = 5.52, p = 0.02$.

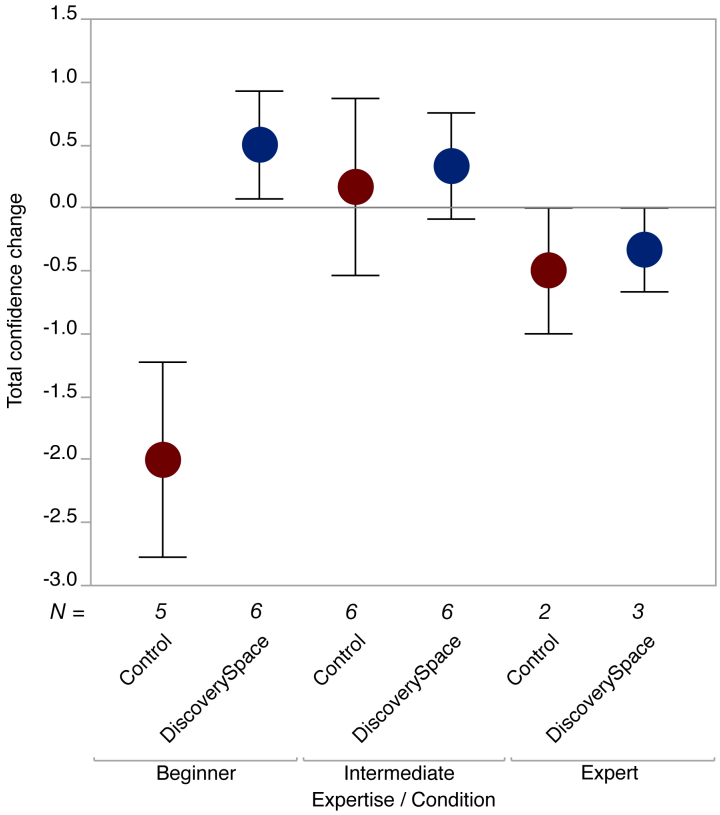

Figure 6.8: Confidence of Beginner Control participants decreased while confidence of Beginner DiscoverySpace participants increased. Error bars show 1 standard error from the mean. Note that due to small sample sizes in the sub-conditions, this is intended only as a descriptive illustration.

All measures compared between conditions are summarized in Table 6.2. Responses to many of the post-editing questions varied widely. Within the Beginner group, DiscoverySpace participants rated difficulty as marginally but not significantly lower ($M = 5.3$) than Control participants ($M = 6.8$), $t(22) = 1.62, p = 0.12$.

Across both conditions, Experts rated difficulty significantly lower ($M = 4.4$) when compared against Intermediates ($M = 5.8$) and Beginners ($M = 6.0$) together, $t(22) = -2.14, p = 0.04$. Experts also used significantly more tools ($M = 26$) than Intermediates ($M = 16$) and Beginners ($M = 16$), $t(22) = 2.37, p = 0.03$. In the DiscoverySpace condition, 51% of the actions used were radical and 49% were refinement, which suggests both types of actions were useful.

Table 6.2: Summary of mean values for measures compared between conditions. DiscoverySpace participants had marginally *higher* confidence change (this effect was significant within the Beginner group), and stated they could not figure something out *less often* than Control participants.

	DiscoverySpace	Control	p
Total confidence change (-8 – 8)	**0.27**	**-0.77**	**0.10**
Couldn't figure something out = yes	**50%**	**80%**	**0.02**
Achieved goal = yes	47%	31%	0.72
Used web browser	33%	54%	0.26
Difficulty (2 – 10)	5.3	6.0	0.37
Creativity (2 – 10)	4.5	4.1	0.86
Confidence about performance (2 – 10)	5.9	5.2	0.68
# Photoshop tools used	15	20	0.38

Overall, there was a strong negative correlation between difficulty and confidence about performance, $r(28) = -.70, p < .0001$. Interestingly, there was also a strong positive correlation between creativity and confidence about performance, $r(28) = .67, p < .0001$. This effect seems to hold most strongly for Beginners, as the correlation was weaker for Intermediates and Experts only, $r(17) = 0.48, p = 0.05$. Further exploration revealed that using the web was negatively correlated with achieving one's goal, $\chi^2(1) = 8.09, p = .005$, and positively correlated with having trouble figuring something out, $\chi^2(1) = 6.93, p = 0.01$.

Preliminary analyses found no effects for type of class (photography vs. design), so this was excluded from further analyses. Gender was confounded with Expertise (Table 6.1) so Gender was also excluded from analyses; however within the Beginner group where Gender was roughly balanced, no Gender effects were found.

Qualitative results

Participants were asked at the end of the session if they had discovered any new tools. All DiscoverySpace participants replied yes, most of them adding that DiscoverySpace had exposed them to effects they had not seen before. 9 of 13 Control participants also responded yes, that they had discovered a new tool, mainly by clicking around or from an online tutorial. Two added

that while they had discovered a new tool, they were unsure how to use it.

DiscoverySpace participants were asked two additional questions about the kinds of tasks for which they might use DiscoverySpace, and about the capabilities they wish it had. The most popular response to the task question was "for making quick and fun edits", such as preparing a photo to post on social media or editing personal photos for fun, as opposed to aiming for professional-looking or detailed edits. The capabilities most participants wished DiscoverySpace had were more control over the result of an effect, such as sliders to dial it down or to adjust its parameters; and the ability to apply the effect to a selected part of the image only. Other requests included an explanation as to how the effect was achieved so that the user could easily repeat it and experiment with it, and better interaction with the document history. Though users could go back and forward through the applied operations using the History panel, it is separate from DiscoverySpace, and does not allow editing of the previous steps.

DiscoverySpace participants were also given the opportunity to share their general thoughts about the panel. Most stated either that they found it helpful, or that it had the potential to be helpful if some of the aforementioned improvements were made. Novice participants in particular seemed on the whole excited by DiscoverySpace as it meant they were able to achieve effects they otherwise would not have thought possible.

6.7 Discussion

Despite the relatively small sample sizes when participants are broken up into sub-conditions, our results suggest that action suggestions can be beneficial for novice users of complex software. Further research should explore these possible effects in more detail.

It may not be that surprising that the confidence levels of participants in the Control condition dropped after using Photoshop. Novice participants were quickly frustrated with the time it took to find functionality and get accustomed to the interface, and more experienced

participants struggled to locate forgotten features. We believe that DiscoverySpace, while it may not currently support all possible goals, provided a friendlier starting point that showed or reminded participants what types of edits are possible. Again, it is not that surprising that this effect was more pronounced for Beginners, as they are the most likely to misjudge their abilities, having never or rarely used Photoshop.

We assume that participants who answered "no" to the question, "Was there anything you wished you could do to this photo but couldn't figure out how?" were able to accomplish the tasks they set out to do. Participants who answered "yes" to the question were prompted to describe what they could not figure out. Most responses named a task the participant had been trying to accomplish (*e.g.*, removing redness from a person's face), but either gave up on or ran out of time. It is likely that DiscoverySpace provided quicker ways to accomplish many of these tasks, and thus participants in the DiscoverySpace condition were more likely to complete them in the time allotted than those in the Control condition.

As the study involved using the full Photoshop interface, expertise had a great impact on user performance. Observations during the sessions confirmed that Expert users were comfortable with Photoshop's interface, whereas Beginners tended to find it overwhelming because of the large number of menus and buttons. Ratings of creativity and confidence about performance varied widely, likely because the task was open-ended, and these measures depended on each individual's goal. The correlation between confidence in performance and creativity indicates that participants in both conditions did tend to include creativity as a criterion for doing a good job, supporting the motivation for an interface that encourages creative exploration. Participants who used the web achieved their goals less often, likely because these are users who had more trouble with their tasks and turned to the web for help. However, using the web did not in fact seem to help them. The following section provides some suggestions as to why that may be.

6.7.1 Common Difficulties

The same researcher conducted all 28 sessions, and observed participants' recurring errors and difficulties. The most prominent observations, outlined here, support our motivation for action suggestions as well as for the previous systems in this dissertation, and provide directions for future work.

1. Failing to find the best way to accomplish a task

Several participants used Google to search for help and often followed the top result without realizing that there was an easier, more efficient, or more effective way to accomplish the task. For example, the automatically generated Google summary listed at the top for "how to crop photoshop" suggests using the Rectangular Marquee tool, rather than the more relevant Crop tool. In addition, many tutorials included screenshots from older versions of Photoshop, making it difficult for participants to find the tools mentioned. Version-specific help is a challenge for all software. Systems like RePlay and ReMap that augment queries with the user's context could help with this by including the user's software version as additional context (Blueprint [22] does this for programming frameworks). For example, the top search result shown when "cc 2015" is appended to the above Google search is recent and demonstrates the Crop tool. However, this only works when the results also mention the software version explicitly, which many online tutorials do not. Future work should investigate how systems might infer this information from online resources based on available metadata.

2. Not noticing when commands cause state changes or side effects

A common issue participants ran into was creating new layers without realizing, and then having the wrong layer selected when trying to apply another effect. For example, performing an adjustment from the Adjustments panel creates a new Adjustment layer, but does not explicitly notify the user of this. Most actions in DiscoverySpace also create at least one layer. In this study,

many participants in both conditions struggled with having the wrong layer selected when trying to apply other edits, which meant those edits did not work as expected. This is an instance of a more general problem in complex software: commands often have side effects that the user is not aware of, which may cause confusion later on. This suggests that programs should 1) make the user aware of what they are doing, and 2) encourage better development of a correct mental model of the program, so that such side effects are not so surprising when they occur. Support for better interactive history may be an important improvement for DiscoverySpace as it would allow participants to work through and explore the steps that have been completed.

3. Difficulty finding out what tools do

For novices, complex software has poor information scent [183]: we observed users sequentially browsing menus, toolbars and panels to find something specific or see what is possible. Even with exhaustive search, Beginners seemed to have difficulty understanding what the tools did, given only their names and brief tooltip descriptions. When participants did find the desired tool, they often had trouble figuring out how to use it. This emphasizes the need for tool-based contextual assistance like RePlay, ReMap, and LiveClips provide. RePlay and ReMap enable users to quickly and easily search for videos about a tool when they click on it, and LiveClips shows users multiple examples of how a tool can be used. Both strategies likely would have helped DiscoverySpace participants. While RePlay and ReMap participants rarely made tool-based queries in Canva, such queries would likely be more helpful in more complex software like Photoshop.

6.7.2 Study Limitations

Having an open-ended task facilitated more realistic usage than a pre-defined task, and having participants work on their own personal photos likely increased their motivation to do a good job. However, participants' widely varying goals resulted in widely varying results regarding

their editing experience and likelihood of achieving their goals. Future studies might consider more directed tasks to reduce the variability in participants' goals.

Using self-reported expertise ratings has the potential for inaccurate assignments based on differences in how participants perceive their own skill. Our observations found their self-report sufficiently accurate. However, future studies might consider a more objective expertise measure, such as a pre-test. The pool of "novices" is larger than one might think: many Photoshop novices were experienced with photography principles and/or simpler photo editing software. They had domain knowledge, just not expertise on the specific task at hand. Especially with feature-rich software like Photoshop, nearly everyone is a novice in some regard.

Participants' widely varying expertise levels yielded small sample sizes for measuring interaction effects. However, for this preliminary study, this variety unearthed valuable differences in participant behaviour.

Finally, many participants ran out of time and stated this as the main reason they could not achieve their goal; future studies should allow participants to work for longer. A longitudinal study would allow for more realistic usage data, as participants could use the application on their own time and more goals could be collected per person.

6.8 Future Work

Participants' experiences with DiscoverySpace suggest several ways to improve the robustness and usability of action suggestions in complex software.

6.8.1 Improving DiscoverySpace for Photo Editing

An important limitation of the current DiscoverySpace prototype is that most actions in its corpus produce *global* edits, not *local* ones. A few actions intended for faces (*e.g.* skin smoothing) allow the user to brush on the effect, but the majority apply to the entire photo.

This is representative of the selection of free Photoshop actions available online; most are global edits because these are much easier to create than ones that allow for user input such as selection. However, users often desire local editing capabilities, and so providing actions that allow for this would be valuable. One way of providing support for user input would be to leverage the interactive steps provided by TappCloud [126].

Based on participants' responses to the question of what tasks they would use DiscoverySpace for, it seems to be effective for quick exploration. In order to help users make more professional-looking edits, DiscoverySpace could take better advantage of the properties of the user's input photo and apply effects in a content-adaptive way [17] or suggest automatic fixes based on properties of the photo such as its histogram (*e.g.* as in [29]).

Other immediate improvements to DiscoverySpace for photo editing will include automatic image analysis to replace manual user descriptions of photos, and replacing the default preview images of effects with previews of the user's actual photo.

6.8.2 General Improvements to Action Suggestions

Many programs have the capability to record and save actions (*e.g.*, Adobe Photoshop, Illustrator, and Acrobat), also referred to as presets (*e.g.*, Adobe Lightroom), action macros (*e.g.*, Autodesk's AutoCAD), and macros (*e.g.*, Microsoft Excel and Word). Even more programs allow for task automation via scripting (*e.g.*, Adobe InDesign, Mac OS). DiscoverySpace and the following proposed improvements can apply not only to Photoshop but to any complex software that provides some way of combining operations into actions.

End-user control

The most frequent piece of feedback received throughout the study and prototyping process was the desire for more control over the effects. One of the main advantages complex programs have over simple ones (*e.g.*, Instagram) is the ability to have fine-grained control over

operations, and so providing this control to users would be a valuable component of an interface like DiscoverySpace. This could be accomplished by examining current usage data of the software in question to determine what parameters users most often edit for common operation sequences, and making only those parameters available when an action is executed. Alternatively, DiscoverySpace could make the action history more interactive by allowing users to select, edit, and delete any step that was performed. To provide an intuitive way of setting parameters, visual previews like the variations in Side Views [223] and TappCloud [126] could be incorporated.

Recommendation algorithm

Given the diverse goals that users have in complex applications, a technique like collaborative filtering could improve recommendations by personalizing the suggestions to the user and their goals. Li *et al.* recommend an "item-based" collaborative filtering algorithm based on their work with CommunityCommands [136]. This algorithm bases recommendations on the similarity between commands, rather than between users. They also found that users preferred recommendations based on their behaviour within the current session, rather than long-term. The goals users enter into DiscoverySpace should also be included as input to the suggestion algorithm, so that suggestions are not only relevant to the user's photo, but also to the goal they have in mind. This natural language description could also be used to annotate the operations executed by users. With this extra information, we could build a better model of the relationships between goals, the language used to express them, and the tools used. Such descriptions would otherwise be difficult to collate, as users would be required to do extra work to describe the operations they perform.

Building a scalable corpus of actions

User-generated actions are posted across multiple sites and carry differing amounts of metadata and detail, and differing distribution licenses. Curating our corpus of actions required

careful attention to these differences, and we had to manually define keywords and names for most actions. Scaling up such a corpus or building it automatically would require accounting for these issues. This could be done by creating a centralized repository inside the software itself or as part of DiscoverySpace, wherein users can create and share actions, and are required to provide keywords and agree to a distribution license. An existing example of this is AdaptableGIMP, an interface for the photo editor GIMP that provides user-created actions referred to as "task sets" [121]. Users can create and share task sets from within the interface, and search the corpus for ones to apply.

Action suggestions need not be user-created. They could instead be generated by automatically recording and segmenting user behaviour along with the type of document being worked on, or by mining and analyzing the large collection of online tutorials that are available for most applications. Work has been done on generating interactive tutorials from static online ones [61,126], and incorporating such work into a search and suggestion interface like DiscoverySpace is a promising direction for improving both the initial usability and long-term learnability of complex software applications.

6.9 Conclusion

This chapter introduced DiscoverySpace, a task-level action suggestion interface, and found that easy-to-execute task-level action suggestions may help prevent novices from losing confidence with complex software. DiscoverySpace allows users to more easily discover what is possible in an application by abstracting away individual operations in favour of understandable and goal-driven actions. By suggesting actions relevant to the user's input document, DiscoverySpace can provide useful assistance toward accomplishing the user's task. The experiment described in this chapter confirmed the need for interventions like this in modern software, and provided results suggestive of the fact that action suggestions are beneficial to users. The next

chapter explores how contextual suggestions can help novices accomplish tasks in a different domain: providing feedback.

6.10 Acknowledgements

We thank Nancy Reid & Catherine Hicks for their help with statistical analyses. This work was supported in part by a Powell Fellowship from UC San Diego.

This chapter, in part, includes portions of material as it appears in *DiscoverySpace: Suggesting Actions in Complex Software* by C. Ailie Fraser, Mira Dontcheva, Holger Winnemöller, Sheryl Ehrlich, and Scott Klemmer in the Proceedings of the 2016 ACM Conference on Designing Interactive Systems (DIS '16). The dissertation author was the primary investigator and author of this paper.

Chapter 7

CritiqueKit: Interactive Guidance Techniques for Improving Creative Feedback

All of the systems presented so far have aimed to support people doing creative tasks in complex software. But with creative tasks, the software is not always the only complex part. Creative tasks can themselves be complex and difficult by virtue of being subjective and open-ended. One example of such a task is giving feedback. Good feedback is critical to creativity and learning, but many people do not know how to actually *provide* effective feedback. This chapter contributes empirical evidence that two interactive techniques – reusable suggestions and adaptive guidance – can help people provide better feedback on creative work. We present these techniques embodied in the *CritiqueKit* system to help reviewers give specific, actionable, and justified feedback. Two real-world deployment studies and two controlled experiments with CritiqueKit found that adaptively-presented suggestions improve the quality of feedback from novice reviewers.

7.1 Introduction: Feedback's Hidden Potential

Feedback is one of the most powerful influences on learning and achievement [86]. Both giving and receiving formative feedback encourage self-reflection and critical thinking on one's

work [134, 167], especially in creative and open-ended domains such as design and writing [86, 196]. The growing scale of many educational and professional settings increases both the importance and difficulty of providing sufficiently descriptive and personalized feedback. Good feedback can be hard to generate, and people are not consistently skilled in doing so [119, 240]. Feedback is often too short, vague, and not actionable [120, 217, 236]. Even experienced reviewers don't always recognize when they are providing poor feedback or why it is ineffective [217].

This chapter contributes two interactive techniques that improve feedback in the moment, their embodiment in the CritiqueKit system, and their evaluation through two deployments and two experiments.

Interactive guidance of feedback characteristics. CritiqueKit features a guidance panel with checkboxes that update as the reviewer gives feedback. A text classifier categorizes feedback as Specific, Actionable, and/or Justified as the reviewer types, providing them with an ambient awareness of their feedback quality and guiding them to improve their feedback while they write it.

Suggesting prior feedback for reuse. CritiqueKit enables reviewers to reuse expert feedback, reducing experts' labor by scaling their feedback to similar work. These suggestions update and adapt based on the feedback's categorization to give reviewers targeted ideas for how to improve their comment and provide inspiration.

Two deployment studies and two controlled experiments investigated the efficacy of these interactive techniques on the quality and characteristics of feedback. The first deployment examined how experienced reviewers (teaching assistants) reuse feedback in an undergraduate course. The second deployment examined how undergraduate students reuse feedback. The first experiment examined the impact of statically presented suggestions and interactive guidance on novice feedback. Finally, the second experiment examined the efficacy of adaptively updating suggestions in tandem with interactive guidance on novice feedback. We found that adaptively-presented suggestions improved feedback quality. Reviewers found suggestions useful for inspi-

Table 7.1: Two deployments (DEP) and two between-subjects experiments (EXP) examined the efficacy of feedback reuse and interactive guidance. We found that interactive suggestions and guidance were most helpful for improving feedback.

Study	Adaptive Reuse	Static Reuse	Interactive Guidance	Participants	n=	Main Finding
DEP1		X		TAs	8	TAs used suggestions as inspiration
DEP2		X	X	Design Students	29	Students reused vague suggestions
EXP1		X	X	Design Students	40	Static suggestions and guidance were not helpful
EXP2	X		X	General Students	47	Adaptive suggestions and guidance were helpful

ration, and the interactive guidance reminded them to ensure their comments met the criteria for effective feedback. This work provides evidence that interactive techniques such as suggestions and guidance can effectively scaffold the feedback process, improving the feedback people give without needing them to spend extensive time learning good techniques (See Table 7.1 for details).

7.2 Related Work

7.2.1 Good Feedback is Actionable, yet Rare

Rapid iteration is critical to the success of creative projects, from essays, to visual design, to buildings [48,196]. Receiving feedback early on is important for learners to test alternatives and course-correct [48,224]. Effective feedback is especially important in educational settings where novices are learning new skills and developing expertise. However, giving effective feedback is rarely taught [167]. As physical and digital classrooms increase in size, the demand for feedback outgrows the ability to adopt the ideal learning model of one-to-one feedback [18]. Instead, a one-to-many approach is utilized, where an expert provides feedback for multiple learners.

Although learners most value expert feedback [66, 237], the one-to-many approach is highly demanding on experts, and specific, actionable feedback for individuals becomes increasingly rare.

In general, effective feedback is specific, actionable, and justified. Specific feedback is direct and related to a particular part of the work rather than vaguely referent [117, 196, 240]. Specific positive feedback also highlights strengths of the work and provides encouragement, so the recipient can tell they are on a good path [107, 228, 240]. Actionable feedback is important because it offers the learner a concrete step forward [196, 217, 228, 240]. Simply pointing out a problem is not sufficient to help one improve [187, 196, 217, 224]. Actionable feedback is often most helpful early in a project [42, 228, 240] because it may help people self-reflect and self-evaluate their work [65], prompting more revisions for improvement [49, 225]. Lastly, justification is an important characteristic of feedback [117, 164, 240], but is arguably one of the hardest to understand or recognize [66]. Justified feedback contains an explanation or reason for a suggested change, which helps the learner understand why the feedback was given.

7.2.2 Rubrics & Examples Usefully Focus Feedback

Rubrics [8, 240] and comparative examples [117] are effective in structuring feedback because they beneficially encourage attention to deep and diverse criteria. Novices otherwise tend to focus on the first thing they notice, often surface-level details [74, 93, 120, 240]. Viewing examples of past designs can lead to greater creativity and insights [118, 149]; thus, showing examples of good feedback may spark ideas reviewers would not have otherwise considered [74, 119, 146]. Also, adaptive examples curated to match design features are more helpful than random examples in improving creative work [130].

Rubrics and other scaffolds require significant upfront manual work by experts who must carefully design a comprehensive rubric, curate a thorough set of examples, or decide how else to structure the feedback process. This chapter investigates leveraging existing feedback to dynami-

cally create rubric criteria. We hypothesize that showing reviewers previously-provided feedback can guide their attention to important aspects of the design.

7.2.3 Is Feedback too Context-specific for Practical Reuse?

Schön persuasively argues that effective feedback should be context-specific and expert-generated [198]. He offers a vignette from architecture where the teacher suggests an alternative building to the student as an example of situated wisdom and its transfer. If Schön is right that this exchange requires both wisdom and context, does that mean that feedback reuse is infeasible? Within a given setting, project, or genre, common issues recur. Hewing to the principle of recognition over recall, we hypothesize that suggestions and guidance can increase novices' participation in context-specific exchanges.

7.2.4 Prior Systems & Approaches for Scaling Feedback

Existing approaches for scaling personalized feedback include clustering by similarity (*e.g.*, for writing [24] and programming [70, 87]). Gradescope [214] and Turnitin [3] allow graders to create reusable rubric items and comments to address common issues and apply them across multiple assignments. Gradescope binds rubric items to scores, which emphasizes grades rather than improvement.

Other methods include automating the reuse of the solutions of previous learners. These methods work best when correct and incorrect solutions are clearly distinct, such as in programming [68, 85] and logical deductions [53]. Automated methods have also found success with the formal aspects of more open-ended domains such as writing [24, 193]. However, assessing the quality and effectiveness of creative work – the strength of a design, the power of a poem – is intrinsically abstract and subjective and lies beyond current automated analysis techniques. Also, little automated analysis exists for media other than text. For domains like design, human-in-the-loop analysis will remain important for quite some time.

7.2.5 Automatically Detecting Feedback Characteristics

Although feedback is often specific and contextual [198], general characteristics can be automatically detected and used to help reviewers improve their feedback. For example, PeerStudio [120] detects when comments can be improved based on the length of the comment and the number of relevant words. Data mining and natural-language processing techniques can also automatically detect whether a comment is actionable or not, and prompt the reviewer to include a solution [166, 236]. Krause *et al.* use a natural-language processing model to detect linguistic characteristics of feedback and suggest examples to reviewers to help them improve their comment [117]. These methods require a reviewer to first submit their comment so it can be analyzed, and then improve their comment after submission.

7.3 CritiqueKit: Interactively Guiding Feedback

Based on these methods and insights, CritiqueKit (Figure 7.1) categorizes feedback and provides prompts and suggestions to reviewers. It differs from prior work by providing feedback to reviewers as they type rather than after they submit. We hypothesize that this contextual assistance may provide a just-in-time scaffold that changes how reviewers' thoughts crystallize, yielding feedback that is more specific, actionable, and/or justified.

Much like the previous systems in this dissertation, CritiqueKit's interface embeds assistance in the context of the user's task, by displaying interactive guidance and suggestions in a panel next to the work being reviewed. Like DiscoverySpace, CritiqueKit prioritizes ease of use for achieving outcomes; users can reuse a feedback suggestion by simply clicking on it. However, unlike DiscoverySpace, CritiqueKit also helps users improve the feedback they come up with from scratch. As our deployments will show, reviewers are often reluctant to directly reuse feedback suggestions, preferring instead to write their own ideas. While DiscoverySpace does not provide support for users who choose to use Photoshop's tools directly, CritiqueKit explores

how simple guidance might help reviewers go beyond direct reuse while still achieving outcomes quickly and easily.

7.3.1 Interactive Guidance as a Form of Scaffolding

CritiqueKit features an interactive guidance panel (Figure 7.1b) with checkboxes that update based on which of three attribute categories the feedback fits: *Is Specific* [117, 196, 217, 240], *Is Actionable* [49, 65, 120, 146, 196, 228], and *Is Justified* [66, 117, 164, 240].

The prototype assesses the feedback's fit with the following heuristics. The heuristic for the *specific* category merely requires that comments be at least five words long because vague comments tend to be short, such as "good job" or "needs work." Perhaps surprisingly, we observed that the five-word nudge was sufficient to garner specific feedback in practice. (Some websites, like Etsy, also use a five-word minimum heuristic for reviews). For the *actionable* and

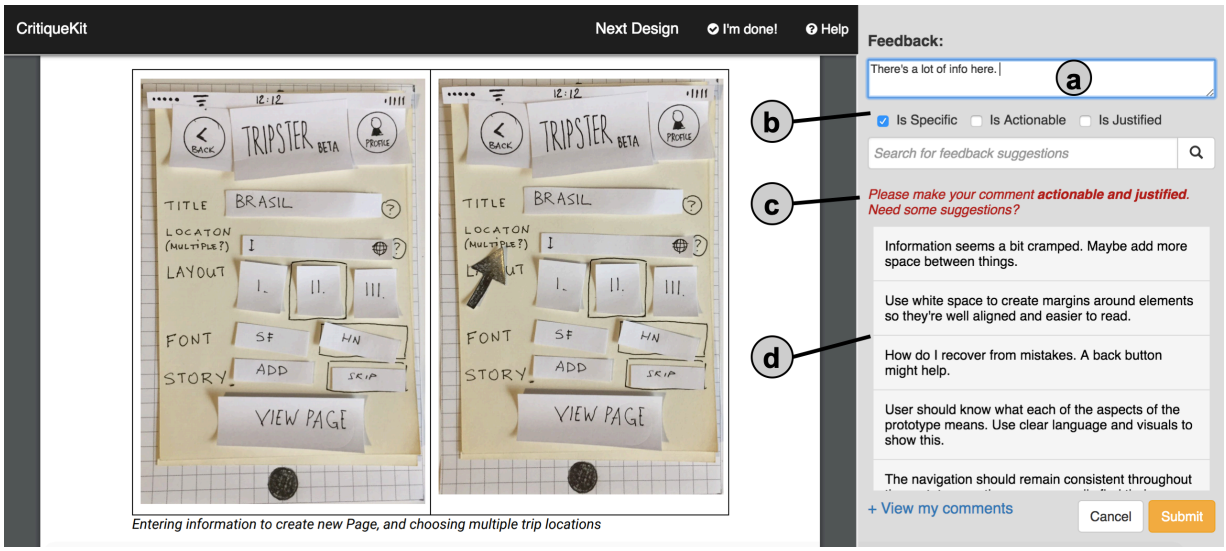

Figure 7.1: The final CritiqueKit interface (used for EXP 2). a) The reviewer can type their feedback in the textbox. b) The checkboxes in the guidance panel update based on the characteristics of the reviewer's comments. c) CritiqueKit explicitly prompts reviewers to ensure their comment fits the checkboxes in the guidance panel. d) The reusable feedback suggestions in the suggestions box update based on the unchecked characteristics in the guidance panel, adapting specifically to the reviewer's feedback.

justified categories, we manually combed feedback that had been hand-labeled as meeting these categories and observed that specific keywords (i.e., "maybe try" and "you should" for actionable; "because" and "so that" for justified) were strong cues of these categories. Consequently, the prototype implementation simply checks for the presence of these keywords and phrases in feedback comments.

A comment is considered complete once all checkboxes are checked. Reviewers can manually check and uncheck the checkboxes if they feel the checkboxes did or did not add a category in error. For example, if a reviewer's comment states, "Use a 2-column grid layout," and the "Is Actionable" checkbox remains unchecked, the reviewer can manually check the checkbox to note that their comment does indeed contain an actionable suggestion.

7.3.2 Adaptive Suggestions for Greater Specificity

The suggestions box (Figure 7.1d) contains a list of previously given feedback from experts. These suggestions dynamically adapt based on how the reviewer's feedback is categorized in the guidance panel. For example, if a reviewer's comment does not yet satisfy the actionable and justified categories (as in Figure 7.1), the suggestions box would contain examples of feedback with these characteristics. Suggestions appear in the order they were added to the corpus.

7.3.3 The CritiqueKit Review Workflow

When a reviewer first opens CritiqueKit, a prompt asks them to provide specific feedback on something they like about the design and something that could be improved. The suggestions box contains general feedback snippets [119] pertinent to the review criteria to give reviewers a starting point, providing suggestions that are broadly applicable and fit within the specified criteria. The "Submit" button at the bottom of the interface is red to indicate that the comment text box is either empty or does not fit any of the categories in the guidance panel.

Once a comment is sufficiently long, the "Is Specific" checkbox will check, and the re-

viewer will be prompted to make their comment actionable and justified. The "Submit" button turns yellow to indicate that their feedback is not yet complete, though they can still submit if desired. The feedback suggestions then change to present comments that instantiate both actionable and justified feedback. The suggestions continue to adapt depending on the characteristics of the comment, showing reusable examples of feedback that satisfy the unchecked categories in the guidance panel. The reviewer can insert a suggestion directly into their own feedback by clicking on it. They can then edit the suggestion if they wish in the text field. Once all checkboxes are checked, the "Submit" button turns green as an indication of completeness.

Using prior feedback as suggestions can give inspiration and highlight common issues. The presence of the structured guidance panel reminds reviewers of attributes that feedback should have.

7.3.4 Implementation

CritiqueKit is a client-server web application implemented using Node.js; it assumes that all content to be reviewed is available on the web. The corpus of reusable feedback comments is stored on the server in JSON format.

CritiqueKit uses web sockets for communication between each client running the application and the main server, implemented using the socket.io module. Feedback classification happens on the client-side using JavaScript. Feedback suggestions are also generated on the client-side after retrieving the corpus from the server; the suggestions box adaptively shows and hides comments using JavaScript.

Users access CritiqueKit by navigating to its URL in a web browser. The first time the browser loads the website, a unique ID is generated for the user and sent to the server. A cookie is also saved on the client-side so that the server can identify and differentiate users. The review content is loaded within the page as an iframe.

7.4 Deployments: (How) Is Feedback Reused?

To understand how feedback is reused in educational settings and evaluate the CritiqueKit approach, we conducted two deployments and two experiments. All studies took place at a research university.

7.4.1 DEP 1: How Do Teaching Assistants Reuse Feedback?

Eight teaching assistants (TAs) (two female) for an undergraduate design course used Gradescope to grade and critique seven weekly assignments that varied in content from storyboards to written explanations to high-fidelity web application prototypes. TAs set rubric items for each assignment and wrote comments for each. We deployed CritiqueKit to first understand how TAs might reuse feedback and made iterative improvements to the design throughout the quarter based on TA input.

Method: Integrating CritiqueKit with Gradescope

To integrate with the TAs' existing workflow, we implemented CritiqueKit as a Google Chrome extension that augments the Gradescope interface with a suggestions box (Figure 7.2). This version of CritiqueKit contained only the suggestions box to explore feedback reuse. The suggestions box contained a manually curated set of feedback provided by former TAs in a previous iteration of the course, stored in a Google Sheet online and retrieved by the Chrome extension using the Google Sheets API. Suggestions were categorized into three feedback categories: Positive, Problem, and Solution. TAs could select feedback suggestions to directly copy into the textbox for further editing. Each rubric item contained its own suggestion box interface, providing suggestions specific to that rubric item.

We curated the reusable suggestions corpus as follows. Given all feedback from the previous quarter, feedback that was 25 or fewer words in length was kept, because longer feedback

was both too long to be skimmed in a suggestion display and tended to be overly specific. Feedback of 26-30 words was truncated at the sentence level to fit within the 25-word limit. Longer comments or duplicate comments were discarded. In total, 526 comments were provided as suggestions throughout the course for seven (of ten) assignments. Suggestions were manually categorized into the Positive ($n = 92$), Problem ($n = 312$), and Solution ($n = 122$) categories.

Result: TAs Used Feedback Suggestions as Inspiration

Across seven assignments, four of the eight TAs reused 51 distinct suggestions from the 526-element corpus (9.7%). 75 of 583 designs received a reused suggestion for feedback. 60% of reused suggestions were categorized in the Problem category. These numbers omit any reuse

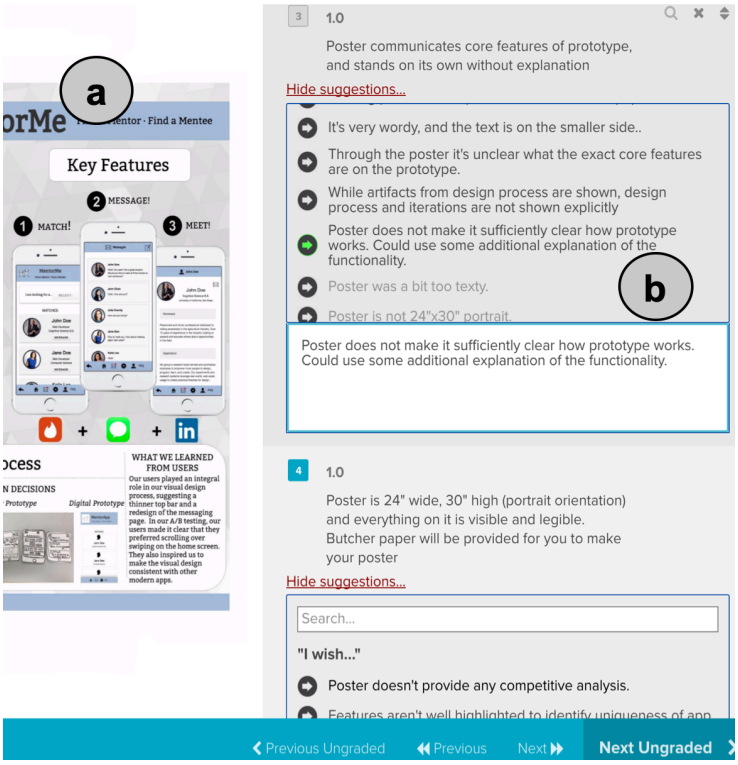

Figure 7.2: CritiqueKit implemented as a browser extension in Gradescope for DEP 1. a) Reviewers provide feedback on a student design. b) The suggestions box under each rubric item provides reviewers with a list of reusable suggestions and a comment box for providing feedback on a submission.

occurring entirely inside Gradescope without CritiqueKit. (Gradescope provides an interface for reusing entire comments within an assignment rather than for individual parts of the comment.)

An end-of-course survey asked TAs about their CritiqueKit use. One commented that he would "*skim the comments in the [suggestions] to see if something was accurate to my thoughts*." Another mentioned that the prototype helped him "*[find] ways to better explain and give feedback about specific points*." TAs also mentioned that suggestions sometimes reminded them to comment on more diverse aspects of students' work. For example, one mentioned that seeing positive suggestions reminded her to give positive feedback, not only critiquing areas for improvement. TAs mentioned using the suggestions as inspiration rather than the exact wording, taking the underlying concept of a suggestion and tailoring it.

7.4.2 DEP 2: How Do Students Reuse Feedback?

The first deployment examined teaching staff usage; this second deployment examined student usage to understand how novices interact with guidance and suggestions. We deployed CritiqueKit as a standalone web application with 29 students in an undergraduate design course for five weeks. Students gave anonymous feedback on two randomly assigned peer submissions for each of seven assignments.

Method: Integrating Interactive Guidance for Scaffolding

Novice students are less experienced in giving feedback and may benefit from interactive scaffolding [189]. This version of CritiqueKit included an interactive guidance panel to help reviewers provide more specific and actionable feedback (Figure 7.3). The categories on the guidance panel were "Is Positive," "Is Specific," "Identifies a Problem," and "Presents a Solution" with checkboxes next to each. These categories stem from recommendations in the literature for both positive and critical feedback [107]. Similar to the final version of CritiqueKit, these checkboxes updated as a reviewer typed by classifying their comment into the three categories.

The categories differed from the final version, focusing on specific and actionable feedback.

The suggestions box was seeded with feedback from the course TA. Similar to the first deployment, the suggestions were categorized in the Positive, Problem, and Solution categories. When a student submitted a comment, it was classified into one of these categories, shortened to 25 words if it was longer, and fed back into the corpus to appear as a suggestion, enabling students to reuse their peers' as well as their own comments. The suggestions were ordered first by frequency used, then by shortest length first, and updated as these values changed and more comments were added. Compared to the final version of CritiqueKit, suggestions were static, meaning they did not change as the reviewer typed.

Results: Positive Feedback Common; Reuse Rare

For seven assignments, 898 comments were submitted. Independent raters classified each comment into the five categories of Positive Only, Positive and Specific (Positive + Specific), Problem Only, Solution Only, and Problem with a Solution (Problem + Solution). 45% of these

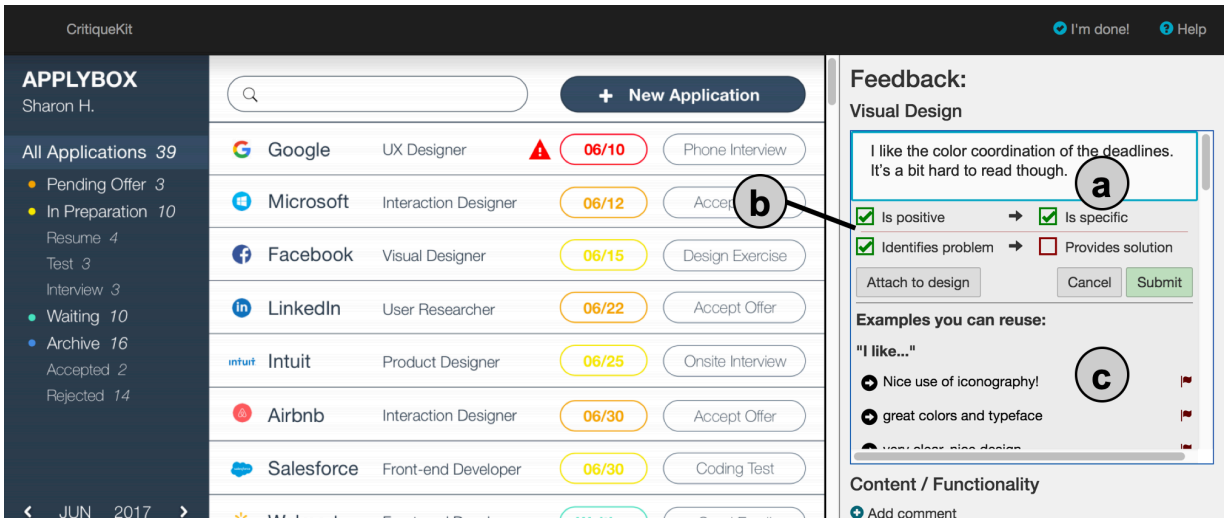

Figure 7.3: The CritiqueKit user interface for DEP 2 and EXP 1. a) The reviewer types their feedback into the text box. b) Checkboxes in the guidance panel update as the reviewer types to show how well the comment fulfills high-quality feedback criteria. c) The reviewer can browse and reuse previously given feedback.

comments contained positive feedback; 30% contained a Problem + Solution statement.

Students rarely selected feedback suggestions for reuse. Over the five-week deployment, 14 distinct suggestions were reused on 27 student designs for four of the seven assignments. These suggestions were mostly short, vague comments such as "I wish this was more visually appealing." This may be because students often left feedback specific to individual designs that did not easily generalize to other contexts. Students' comments in a post-survey confirmed that the suggestions did not always seem applicable. Students also did not regularly use the interactive guidance panel; 15 of the 29 students engaged with the panel a total of 120 times over five weeks.

In contrast to how TAs reused feedback, students may not have recognized common issues. TAs paid attention to common errors between designs and mainly reused Problem feedback, whereas students may not have noticed or attended to underlying issues between designs. For instance, one student mentioned that they did not use the feedback suggestions because they "*rarely pointed out the same things when critiquing interfaces*."

This exploratory deployment investigated how students reuse feedback and respond to interactive guidance in the classroom. To understand how a system with these features compares to a standard feedback system, the next study was a controlled between-subjects experiment.

7.5 Experiments: Scaffolding Feedback

Following our deployments, we conducted two empirical studies to investigate the impact of suggestions and guidance on feedback quality.

7.5.1 EXP 1: Do Static Suggestions Improve Feedback?

In an online between-subjects study, 40 undergraduate design students were asked to review three restaurant website homepages using CritiqueKit. The task emulated peer review tasks often required in creative courses. This study's suggestion corpus came from a design feedback

task on CrowdCrit [146] and was categorized in the Positive, Problem, and Solution categories. We hypothesized that suggestions and guidance would help reviewers provide more specific and actionable comments.

Method: Reviewing Restaurant Websites

40 participants were randomly assigned to either the CritiqueKit condition or the Control condition (20 in each). CritiqueKit participants used the same version of CritiqueKit as DEP 2 with all features available (Figure 7.3). Control participants used an otherwise identical version consisting solely of a comment text box. Upon landing on the homepage of either version, participants were provided with a scenario explaining that three restaurant owners are seeking feedback on their new website design. Participants were given a brief tutorial of CritiqueKit's features and an explanation of what makes for good feedback. There were no restrictions or requirements on time spent or amount of feedback to provide. We compared the percentages of comments in five categories. Comments including a supportive element were labeled as *Positive Only* or *Positive + Specific*. Comments including a critical element were labeled *Problem Only*, *Solution Only*, or *Problem + Solution*.

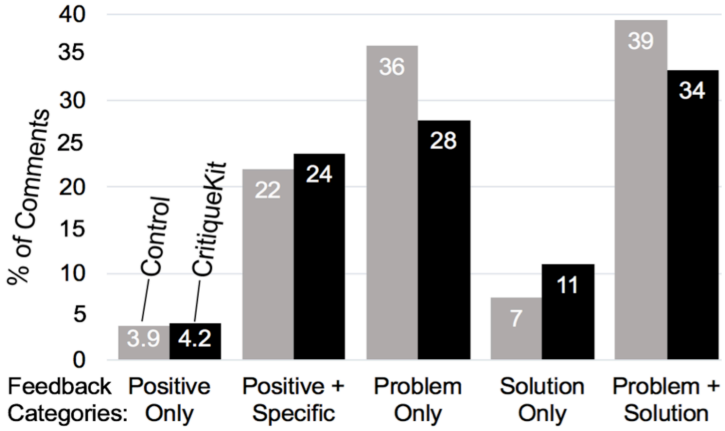

Figure 7.4: A plurality of feedback in both conditions in EXP 1 identified both a problem and solution (*i.e.*, was actionable). Feedback that was only positive was the rarest. There were no significant differences between conditions for these categories.

Results: Static Suggestions Were Not Helpful

With static suggestions and interactive guidance, there were no significant differences between conditions. (To foreshadow, we will see differences in EXP 2, which adds adaptive suggestions). Participants provided a total of 323 comments (168 for control, 155 for CritiqueKit). The average number of words per comment was not significantly different between conditions (Control: $m = 29.07, SD = 23.64$; CritiqueKit: $m = 23.22, SD = 17.3$) ($F(1,38) = 2.52, p = .11$).

Suggestions & Guidance Did Not Affect Type of Feedback

The distribution of the five category types did not vary significantly between conditions ($x^2 = 4.80, df = 4, p = .31$) (Figure 7.4). In both groups, participants provided mostly Problem + Solution feedback (39% in Control; 34% in CritiqueKit).

Most CritiqueKit Participants Corrected Category Labels

65% of CritiqueKit participants actively used the guidance panel, making a total of 85 corrections to categories. Interaction with the guidance panel may have indicated attention to the feedback characteristics. As the study was online, we don't know how many of the other 35% were influenced by the guidance panel.

Unfortunately, People also Reused Vague Suggestions

11 distinct suggestions from the corpus were reused. 8 of these were vague; 3 were specific. 15 of 155 reviews included a reused suggestion. This seems especially low given the high engagement with the guidance panel. We see two reasons for this: First, the suggestions came from CrowdCrit [146], where participants provided feedback on a weather app design. The study task was different than the task for which the suggested feedback was originally given, and novices may have had a limited ability to see the deep structure behind a suggestion and reapply it in a new context. Second, the suggestions were created by crowd workers and of uneven quality.

The suggestions selected were typically short, positive comments, perhaps because students did not know how to apply them in the specific context. For example, the most commonly reused suggestion was "great use of color" (reused 3 times). This result is similar to DEP2 in which students did not find feedback provided by other peers or novices to be useful and generalizable. Feedback suggestions may require more curation or quality control to be most useful.

Suggestions & Guidance Should Work in Concert

While this version of CritiqueKit contained both feedback suggestions and interactive guidance, these features functioned independently. Regardless of the categories checked in the guidance panel, the suggestions remained static and presented in the same order for each participant, potentially making them easy to ignore if they were irrelevant to the context. Participants may have paid attention to only one feature at a time. The next study investigated the question of whether adaptively presenting feedback suggestions along with interactive guidance improves feedback.

7.5.2 EXP 2: Do Adaptive Suggestions Help?

The second experiment used the final version of CritiqueKit described in the system section to test the hypothesis that adaptively-presented suggestions combined with guidance would improve feedback by increasing the fraction of feedback that is specific, actionable, and/or justified.

Method: Reviewing Paper Prototypes

We conducted a between-subjects in-person study with 47 (27 female) participants. Participants were recruited from an undergraduate subject pool within the Psychology and Cognitive Science departments. Participants were randomly assigned to either the CritiqueKit ($n = 24$) or

Control ($n = 23$) conditions. 44 of these participants had no design course experience; 3 participants had taken at least one design course. 28 spoke English as a second language.

Participants were asked to provide feedback on two designs from students enrolled in an online course who volunteered to receive more feedback on their work. These designs were PDFs of mobile application paper prototypes. The review criteria included whether the prototype supported the student's point of view and whether it seemed easily navigable. Participants were first shown the design instructions and review criteria and then given a short tutorial of CritiqueKit as well as an explanation of what makes for good feedback. CritiqueKit participants had all features of CritiqueKit available to them (Figure 7.1), while Control participants used a version that consisted of only a textbox for their feedback. The task took about 30 minutes to complete. After providing feedback on both designs, participants were interviewed about their feedback process and use of CritiqueKit.

Presenting Feedback Suggestions Adaptively

The categories on the guidance panel and their definition used for coding participants' responses were the following:

Specific: relates directly to the review criteria

Actionable: gives a concrete suggestion for improvement

Justified: provides a reason for why something is done well or should be improved

For DEP 2 and EXP 1, the guidance panel categories sought to encourage specific and actionable feedback (Figure 7.3). Examining the feedback from our previous studies, we found that "Is Positive" and "Identifies a Problem" did not provide significant guidance as reviewers were generally aware of whether their feedback was positive or critical. In addition, the guidance panel did not explicitly check for justification of feedback. For EXP 2, we revised the categories to "Is Specific," "Is Actionable," and "Is Justified" to also encourage the explanation or reasoning behind feedback. As described in the system section, the checkboxes update as the reviewer types

to reflect the categories present in their comment, and the suggestions adapt to show feedback examples from categories not yet present in the comment.

Results: CritiqueKit Participants Provided More Specific, Actionable, & Justified Feedback

Participants provided 158 total comments (79 control, 79 CritiqueKit). The percentage of comments that contained all three categories (specific, actionable, and justified) was significantly higher in the CritiqueKit condition (53%) than in Control (30%) ($x^2 = 8.33, df = 1, p = .01$) (Figure 7.5). As an example, this comment meets all three: "The 'more questionnaires' section (Specific) should be made smaller (Actionable) because it is not the main focus of the page." (Justified). The system's heuristic for checking specificity of a comment was quite simple: five words or greater in length. Feedback raters blind to each condition used a more sophisticated and holistic assessment, taking specific to also mean related to the review criteria. With this assessment, 98% of CritiqueKit comments were labeled by raters as specific whereas only 83% of Control comments were. These raters also rated comments from EXP 1 within the specific, actionable, and justified categories to provide a comparison between the two experiments. Interestingly, the percentage of comments containing all attributes in the Control condition was relatively consistent between EXP 1 (35%) and EXP 2 (30%). The percentage of comments with

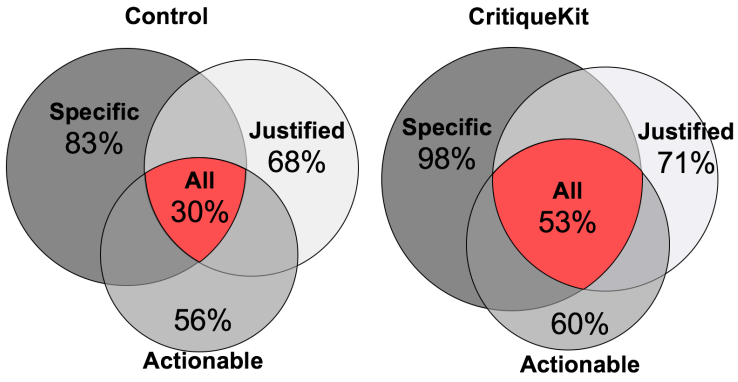

Figure 7.5: In EXP 2, a significantly higher percentage of feedback in the CritiqueKit condition (53% versus 30%) contained three attributes of good feedback: Specific, Actionable, and Justified.

all attributes in the CritiqueKit condition greatly increased between the two experiments (26% versus 53%). Having the checkboxes may have explicitly reminded CritiqueKit participants to ensure their comments satisfy the specific, actionable, and justified categories.

Because longer comments were more likely to contain all three categories, each comment was also scored on a point scale and averaged per participant. Comments received one point for each specific, actionable, and justified idea (Figure 7.6). A MANOVA with category points as dependent variables shows a significant difference between conditions ($F(1,3) = 3.21, p < .005$). CritiqueKit participants provided more specific ideas than Control participants (Control $m = 3.87$, CritiqueKit $m = 5.17$, $F(1,156) = 14.04, p < .05$). This may be because the suggestions provided examples of relevant ideas and led CritiqueKit participants to address more. CritiqueKit participants also provided more ideas that fit all three categories than control (Control $m = 1.0$, CritiqueKit $m = 2.2$, $F(1,156) = 8.78, p < .005$). Given that most participants did not have any design experience, the combination of adaptive suggestions and guidance may have been most useful for these reviewers. The suggestions may have provided a starting point while the guidance panel helped them understand how to apply the attributes of good feedback. There were no significant differences in the average number of actionable and justified ideas in comments.

On average, Control comments were 39.3 ($SD = 30.3$) words long and 43.7 ($SD = 31.4$) words for CritiqueKit comments. There was no significant difference in comment length ($F(1,156) =$

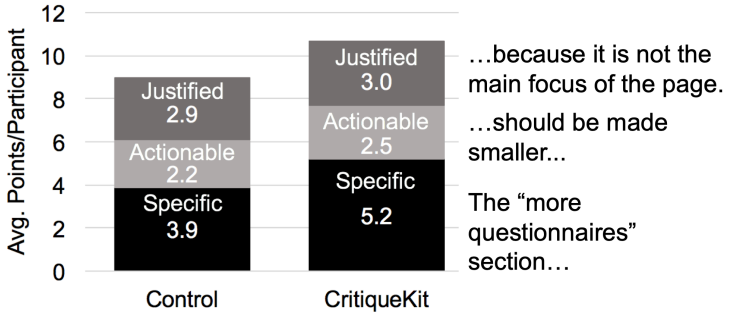

Figure 7.6: CritiqueKit participants provided more *specific* and *all-three* category ideas than control participants.

$1.77, p = .19$). Unfortunately, we don't know how the feedback improved students' work because they received feedback from both Control and CritiqueKit participants. A longitudinal deployment with the final version of CritiqueKit would likely be more useful in determining the helpfulness of feedback.

Suggestions Helped Reviewers Describe Their Thoughts

Participants rated the suggestions as being generally helpful ($m = 4.29, SD = 0.95$, 1-5 Likert Scale). When asked to elaborate on their rating, many participants noted that the suggestions helped them describe their thoughts. One participant remarked, "*I was a bit lost at first because I didn't know how to describe my thoughts. The suggestions helped me figure out how I should describe what I was thinking*." Similarly, another mentioned that "*when I [didn't] know how to put my feedback in words, I could look at the suggestions*." Particularly for participants without any design experience, suggestions helped with appropriate language to use in their feedback. For example, one noted that "*seeing actual wording from a designer's point of view was good so you know how to say what you want to say*." Though a few participants did not directly select suggestions, it is likely that they were inspired or influenced by them as they used similar wording in their own comments.

Still, some participants felt the suggestions were too general and not entirely relevant to the specific design they were reviewing. One participant felt constrained by the suggestions, stating that she ignored them because she wanted to write her own opinions. Suggestions seemed most helpful for participants who used them as a starting point for their own thoughts rather than solely relying on them. Participants who simply selected suggestions tended to list issues without adding their own elaboration. This behavior not only led to incomplete feedback, but also produced depersonalized and scattered comments. For example, one comment that solely relied on suggestions reads "User immediately knows the purpose of the prototype. Good use of grid layout to keep items aligned. Icons should be immediately recognizable to the user." A

consideration for future work is to develop feedback suggestions tailored to help reviewers provide more cohesive and contextual comments.

Most participants in this experiment did not have any design experience and may have benefited most from the suggestions. Many participants noted using the suggestions as a way to find ideas whereas students with design experience may already have heuristics and processes in mind when providing feedback. Future work should examine how suggestions and guidance might improve feedback for more experienced learners as well.

Interactive Guidance Helped Remind & Focus Reviewers

Participants were mixed on the helpfulness of the guidance panel ($m = 3.67, SD = 1.2$, 1-5 Likert Scale). Those who did find it helpful noted that the categories helped guide their feedback process. For instance, one participant noted that he "*went in order of the checkboxes. First, I provided something specific, then something actionable, then justified it*." Another noted that the categories helped her know whether her feedback was actually useful or helpful, and one noted that the guidance panel "*[made] sure the feedback is complete and not vague*."

Anecdotally, we observed that when participants said the categories were not useful, it was because they believed them to be inaccurate in their classifications. The accuracy (compared to human raters) for the actionable category was 67% and 75% for the justified category. A participant stated that "*[the checkboxes] didn't always check when I thought they should, so I would just do it myself*." Another thought the checkboxes were "*quick to judge, it felt like it wasn't reading what I was saying*." Three participants, who were not native English speakers, found the categories confusing because they weren't sure what they meant. Future iterations of CritiqueKit could include the definition of these categories in the prompts to make the meaning clearer. Interestingly, a couple participants noted that they used the categories as reminders rather than for active guidance. For instance, a participant mentioned that though he felt the interactive guidance was not that accurate, "*[the checkboxes] reminded me to make sure my comment contained specific, actionable,*

and justified parts, so I'd go and reread through my comment."

Some participants commented on the adaptive presentation of the suggestions with the guidance panel. For one participant, the suggestions helped him understand what the categories meant. He noted, "*The whole actionable and justified thing, I didn't know what that meant, so the suggestions helped with that*." Observations of participants showed that some clicked on the checkboxes simply to see the suggestions under each one. When asked about how useful they found CritiqueKit in general, participants varied widely in their ratings of usefulness. A more precise measure would allow participants to compare across conditions, which was not possible with this between-subjects design.

7.6 General Discussion

This chapter empirically investigated two techniques for scaffolding feedback: reusable feedback suggestions and adaptive guidance. This work complements this dissertation more broadly, highlighting the benefits of adaptive guidance for interfaces that support creative tasks. Here we discuss and synthesize the findings.

7.6.1 Generating Reusable Feedback Suggestions

This work investigated whether suggestions and guidance can scaffold the feedback process. For this strategy to work, an eye towards reuse and adaptive feedback must be adopted. As Schön argues, experts may be most capable of recognizing common patterns and giving useful feedback [198]. However, while feedback should be specific, underlying concepts can generalize across contexts. In the studies that used expert-generated feedback suggestions (DEP 1 and EXP 2), participants cited the same reason for why the suggestions were useful: as inspiration. Participants reported that the suggestions helped them find words for their thoughts or helped direct their attention to issues they did not originally notice. This suggests that reusable suggestions

should focus attention to common issues rather than specific instances. Our approach demonstrates how expertise on creative work can be scaled by providing feedback on a few to apply to many [119]. This extends work on reusable feedback in coding and writing [24, 85, 87] while keeping the human in the loop, enabling novices to learn and reuse expert insights.

It is possible that more general suggestions can lead to less personalized feedback, particularly in abstract domains like visual design. We observed this in 7 of the 79 comments from the CritiqueKit condition in EXP 2, in which the four participants simply selected suggestions without further elaboration. A consideration for creating and presenting reusable suggestions is how these suggestions can be both general yet personal to be more helpful to the recipient.

7.6.2 What is the Best Way to Guide Feedback?

Prior empirical work on feedback (*e.g.*, Kulkarni *et al.* [119] and Krause *et al.* [117]) has not compared static and adaptive suggestions. In this chapter, we found that people rarely used static suggestions and did not find them helpful; adaptive suggestions were used more and found more helpful. This reinforces prior work demonstrating that adaptive presentation of examples can improve learning [130, 163]. By presenting feedback suggestions that directly addressed missing characteristics of a reviewer's feedback, reviewers were prompted on where they could specifically improve, and explicitly shown examples of how to do so.

The second experiment adapted feedback suggestions based on whether their feedback was categorized as specific, actionable, and/or justified. Though some of the prototype's categorizations were misleading or inaccurate (for example, the comment "user flow is simple" was categorized as "Is Actionable" because of the word "use", even though it lacks a concrete suggestion), participants still referenced the three categories when composing their comments. The guidance panel was useful as a reminder to include the attributes of good feedback in their comments. A more sophisticated method for categorization would likely be helpful, though our naïve approach performed reasonably well overall.

The guidance panel focused on three important attributes of good feedback. A consideration is to also provide guidance for emotional content in feedback, as emotional regulation is important to how learners perceive feedback [117, 230]. In addition, other characteristics may also contribute to perceived helpfulness, such as complexity or novelty [117], that could be further explored through adaptive guidance.

7.6.3 Creating Adaptive Feedback Interfaces

In order for adaptive guidance to be most effective, the interface should be suitable for adaptation. In the two deployments and first experiment, the suggestions were not curated in any way: more than 1,400 comments were supplied as suggestions, but only 76 of these were reused by reviewers. Having more suggestions available was not beneficial because the suggestions were not sufficiently adaptable and were potentially irrelevant and difficult to browse. EXP 2 introduced a curated approach: experts provided the suggestions with generalizability in mind. Of the 47 suggestions created, 29 were reused. Though fewer suggestions were available, they were more general and adaptable, potentially making them more useful.

Suggestion presentation shares many properties with search interfaces. Like with search, a good result needs to not only be in the set, but toward the top of the set [89]. The second experiment contained fewer suggestions, enabling easier search and browsing. Effective curation and display of suggestions should take into consideration the quality of feedback suggestions and how likely they are to be selected, potentially using frequency or some measure of generalizability as a signal.

7.7 Conclusion

Looking across the deployments and experiments, adaptive suggestions and interactive guidance significantly improved feedback while static suggestions did not offer significant im-

provements. These techniques were embodied in the CritiqueKit system, used by 95 feedback providers and 336 recipients. Although CritiqueKit's main goal was to help reviewers produce better outcomes quickly and in the moment, we suspect that the combination of interactive guidance and illustrative examples may also help reviewers learn and retain procedural about writing good feedback. Future work should evaluate this hypothesis and further investigate how best to create, curate, and display helpful suggestions.

Much knowledge work features both underlying principles and context-specific knowledge of when and how to apply these principles. Potentially applicable feedback and review areas include domains as disparate as hiring and employee reviews, code reviews, product reviews, and reviews of academic papers, screenplays, business plans, and any other domain that blends context-specific creative choices with common genre structures. More generally, this work, together with DiscoverySpace, showed how contextual suggestions and guidance can help improve creative outcomes in the moment by leveraging and reusing existing expert work. We hope that creativity support systems of all stripes will find value in the ideas and results presented here.

7.8 Acknowledgements

We thank Kandarp Khandwala and Janet Johnson for help rating feedback. This research was supported in part by a fellowship from Adobe Research.

This chapter, in part, includes portions of material as it appears in *Interactive Guidance Techniques for Improving Creative Feedback* by Tricia J. Ngoon, C. Ailie Fraser, Ariel S. Weingarten, Mira Dontcheva, and Scott Klemmer in the Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). The dissertation author was one of the primary investigators and authors of this paper.

Chapter 8

Conclusion and Future Work

This dissertation demonstrated how expert help and demonstrations can be reused, repurposed, and curated to provide personalized contextual assistance to people doing creative work. The development and evaluation of four interactive systems showed that curating existing expert help and demonstrations and presenting them to users in context helps novices build confidence, accomplish tasks, and produce higher-quality creative work. This chapter proposes future directions based on the challenges and open questions that emerged from this research.

8.1 Future Directions

8.1.1 Multimodal Interaction for Integrating Resources Into Creative Work

This section discusses how future systems might integrate help resources more closely into the user's workflow by leveraging multimodal interaction. Specifically, our experience developing and testing ReMap surfaced some challenges with using speech and pointing for complex search tasks. However, with careful design, multimodal systems could enable more natural interaction with help resources so users can more easily understand processes and achieve complex outcomes.

Conversational Interaction Leverages the Benefits of Speech

Designing an effective speech-based interface requires careful attention to how speech is used [96]. Simply speaking a search query out loud may not be the best way to maximize users' cognitive abilities, as it gives the user only one chance to describe their intent. Interacting with an expert in-person typically involves back-and-forth dialogue to clarify and update intentions; a conversational system (*e.g.*, [96, 147]) may therefore be a more effective design for systems like ReMap. For example, Manuvinakurike *et al.* [147] propose a conversational approach for editing photos where users can incrementally update results through dialogue. This allows users to achieve complex effects without having to figure out complex interfaces or learn technical terminology.

Several ReMap participants found that typing their query was easier than speaking it as they could refine and reword it while typing. Speech as a parallel input modality while working on software tasks may be better suited for short commands (*e.g.*, "rotate the image") rather than long, descriptive queries (*e.g.*, "how do I add a title on top of a photo and position it in the center and make the text stand out"). Fittingly, people who use voice assistants on desktop issue short commands more often than search queries [157]. Speaking is convenient when one's hands are busy with a motor task that requires little focus (*e.g.*, washing the dishes), but speaking at the same time as doing other text-based activities is often difficult because they both use verbal working memory [15, 206]. A conversational approach would allow users to interact more naturally with the system, giving details when they think of them. As smart assistants grow in popularity and ability, systems that converse with users while also leveraging relevant context will be increasingly useful and valuable, as they provide a familiar form of interaction with gentle learning curves [114] that brings us one step closer to the ideal experience of one-on-one tutoring [18].

How to Better Leverage the Specificity of Deictic Resolution

Arguably the most exciting piece of ReMap was its ability to process deictic references by the user. This enabled a form of "chunking," [28] allowing users to point at relevant elements instead of having to describe them in words. However, in our study (Section 4.4), people rarely made deictic references, and the ones they did make did not seem to have a significant impact on the usefulness of search results. Why?

I believe the lack of utility for deictic references is largely due to most queries being goal-oriented (rather than tool- or object-oriented) and a concern of being *too* specific in a search query. First, the evaluation of RePlay (Section 3.5) as well as previous research [21] found that people mention *actions* or *goals* more than tools in search queries. While deictic resolution might be more useful in software with more complex terminology than Canva, it still does not help users describe what they *want* to do. When users have a high-level goal, they often do not know what tools they even need, so referencing tools in a query is not helpful. In the ReMap study, participants who made deictic references mostly used them to refer to objects on the canvas, but in most cases the names of such objects were straightforward (*e.g.*, photo, text) and so the feature did not save users much time.

Second, adding more specificity to a search query is not always helpful, as it relies on there being existing resources that match the specifics of the search query. For example, when pilot testing ReMap with iMovie, we would often make deictic references to various pieces of the project timeline, which added terms like "video filmstrip" and "audio waveform" to the query. However, help resources do not typically use these terms (opting for simpler words such as "timeline"), and so the results from these searches were fewer and less useful. People often resort to tutorials that show how to do a similar task to their own because an exact match does not exist [123]. So, describing a task in more exact terms (*e.g.*, "how to align the center of heading text with a rectangle below it") may bring few results, when a more general description (*e.g.*, "how to align objects") would be better.

Adding specificity via deictic resolution may be more beneficial for systems where the user is either issuing commands directly to the system (*e.g.*, PixelTone [127]), or asking questions to other humans (*e.g.*, CodeOn [38]). However, relying on other humans will inevitably result in a delayed response when someone is not immediately available, and the solution the user desires may already exist on the web somewhere else. And a system that responds automatically to commands will inevitably encounter situations where it does not understand the user's command. One way to mitigate this is to have the user teach the system: for example, the mobile system SUGILITE [135] allows users to issue voice commands like other mobile voice assistants, but when it does not know how to do what the user asks, the user can demonstrate and the system learns for next time. But what about when the user does not know how to do what they are asking?

A Multimodal System That Adapts to the User's Goal

Ideally, a system for contextually presenting expert resources could leverage the strengths of both speech and deictic interaction by adapting based on the goals of the user and the available information. As the line between search engines and smart assistants continues to blur, people use such systems for both executing commands and issuing natural language search queries [5, 21, 57, 169]. A system that supports both would allow users to better leverage the benefits of deictic resolution for specifying their requests.

If the user makes a request the system has not seen before, it could search the web for related tools or instructions and automatically adapt results to the user's particular request and context. If no relevant web results exist, the system could send the user's question to available experts (from discussion forums or other community connections) for help [4, 38, 111]. Over time, the system could learn from the user's actions in response to web search results and expert responses to build up a library of user requests and corresponding actions, thus leveraging the knowledge of online experts and communities in a more robust and extensible way. Then, if a user makes a request the system has seen before, it could automatically execute the command (if the

user wants a quick outcome) or walk the user through instructions (if the user wants to understand the process). In the latter case, the system could help the user follow along with instructions by highlighting mentioned tools in the software's interface or even overlaying images and videos on top of relevant parts of the interface. The system could engage the user in conversational dialogue to confirm parameter settings or clarify the request.

To inform the design of such a system, future work should first explore in more detail how people ideally expect a multimodal help system to function and behave. This could be accomplished by conducting Wizard-of-Oz studies where participants use a multimodal "system" that responds to any request they make. A researcher could serve as the "wizard" and control the system behind the scenes to accomplish whatever participants ask for. Another reason ReMap participants may not have used deictic resolution as much as they otherwise would have could be that they lost trust in the feature after trying it once or twice and failing to make a successful deictic resolution (due to the system's limitations). A Wizard-of-Oz study would mitigate this concern by not relying on a system to understand user's deictic references or speech commands, instead exploring how users would behave with an "ideal" system. One could also vary the tasks required of participants from open-ended tasks (like creating abstract art) to specific tasks that require participants to produce a given outcome (like the design re-creation task used to evaluate ReMap). This would help us understand how users' needs vary depending on their task, to inform the design of an optimally helpful creativity support system.

8.1.2 Enabling Better Understanding of User Context and Expert Resources

The systems presented in this dissertation all rely on having some semantic knowledge about the expert resources used (in order to curate and present them in a useful way) as well as contextual knowledge about the user's environment and task (in order to find relevant expert resources). Each system had limitations in how such knowledge was gathered and used automatically; most available data was low-level (*e.g.*, usage logs) but higher-level knowledge about the

user's task and intent might be more useful for connecting users with relevant resources. Indeed, understanding task-level semantic information about both user behaviour and online resources such as videos is still a largely unsolved problem in the research community. This section discusses the challenges and possibilities for obtaining such knowledge, as well as the trade-offs that emerge between application-general and application-specific solutions.

Obtaining Task-Level User Context

Like with deictic resolution, too much context can sometimes add too much specificity or noise to a search query and degrade the quality of results [50, 55]. RePlay and ReMap used context not just for augmenting search queries, but also for determining relevant moments within videos, inspired by Ekstrand *et al.* [50]. But the only type of context RePlay and ReMap used were the user's recent tools. As both studies found, tool-level context is often not helpful for novices. There are many more types of context that may be useful for search, such as application state, details about the user's document, and context beyond the application such as system-level settings or the user's prior knowledge [50]. Higher-level semantic context such as the type of document open or the type of task being attempted might be more useful. Future work should explore how to identify such context and add it to search queries when appropriate.

Automatically obtaining a task-level understanding of the user's context and intent is a longstanding and difficult problem, but some recent work has begun to explore solutions. Liu *et al.* [139] proposed a machine learning approach for identifying a hierarchy of subtasks from low-level usage logs of image editing tasks. This data-driven approach could likely be extended to other types of creative software tasks. One could also compare low-level usage logs to a large corpus of usage logs with accompanying videos, like Wang *et al.* [233]. If such videos also include descriptions or audio narration, this information could be used to generate a mapping between low-level usage and natural language activity descriptions, which could in turn be used to improve users' search queries. Another promising method for inferring higher-level context from usage

might be computer vision, which has been successfully used to identify interface elements [36,45, 97]. Comparing the visual properties of the user's document and interface with those in existing online resources could help identify the most similar resources to the user's activity.

Understanding the Contents of Expert Resources

The expert resources we gathered for DiscoverySpace and CritiqueKit were manually labeled with semantic information (tags describing actions in DiscoverySpace and presence of feedback characteristics in CritiqueKit). RePlay, ReMap, and LiveClips leveraged pre-existing metadata (video captions for RePlay and ReMap and time-stamped usage logs for LiveClips), but even with these data sources, there were several important limitations in what we could infer from the videos. Addressing these limitations might help such systems provide more useful results to users and improve the ways people navigate within results.

A challenge we encountered in the development of RePlay, ReMap, and LiveClips was how to understand the contents of screencast videos. Most online screencast videos do not include accompanying usage logs, and although some platforms (like YouTube) automatically generate captions, we found that they often include mistakes, especially for domain-specific terminology that is uncommon in everyday speech. Many other video platforms do not generate captions at all. Prior work has demonstrated how computer vision can extract usage information from screencast videos [11, 184] but these methods miss invisible operations, such as keyboard shortcuts and changes of state that do not cause visible change. Similarly, extracting tool information from text-based tutorials is possible [59, 60, 61, 176] but misses any operation not explicitly described.

Usage logs and tool mentions are helpful for identifying specific moments of tool use, but what users more often need when browsing tutorials is higher-level information about the steps in the task [40, 84, 112, 184]. While text-based tutorials are often structured based on steps (*e.g.*, `instructables.com`), in videos this structure is often implicit and therefore not surfaced in most video-viewing interfaces [178]. Determining task-level structure from videos with or

without usage logs is still a challenge [40, 78, 184], even with the help of audio transcripts [62]. Some systems leverage the crowd to generate higher-level labels for videos [112, 234], but doing this for every new video that gets uploaded takes time, resources, and moderation.

What if instead of reverse engineering expert resources, we could give the expert creators simple low-cost ways to annotate and label their own resources *while* they make them? No automated method will ever match ground truth perfectly, but if adding structure to help and demonstrations was as easy as making them in the first place, perhaps more creators would do so. For example, video creators could say certain keywords out loud every time they start a new subtask, and video browsing interfaces like RePlay could use these keywords to display higher-level segmentations of videos to viewers. Or, people recording action macros could describe what they are doing out loud while they record, and action recommendation systems like DiscoverySpace could analyze those transcripts to determine descriptive keywords for each action, and even provide users with explanations of particular steps if they are interested. A major challenge with implementing such a strategy would be coming up with a standard that multiple resource-sharing platforms would leverage, so that enough segmented expert resources are available. A major advantage of leveraging YouTube videos for RePlay and ReMap or action macros on the web for DiscoverySpace was that these are existing corpora with a large variety of expert resources.

Application-General vs. Application-Specific Solutions

The systems presented in this dissertation illustrate the trade-offs between system-wide solutions that provide contextual support across multiple applications (RePlay and ReMap) and application-specific solutions that are integrated into a single application for a single domain (LiveClips, DiscoverySpace, and CritiqueKit).

Using accessibility APIs to build an application-general solution for RePlay and ReMap meant that important contextual information was often missing, for example when users tried to make deictic references to objects that did not have accessibility labels. It also required a more

complicated implementation. However, RePlay and ReMap enabled consistent support for cross-application workflows, and their general solution of indexing into videos using captions was able to find relevant moments reasonably well.

On the other hand, integrating LiveClips and DiscoverySpace directly into Photoshop meant that they had more information about the user's behaviour and context. We were able to tailor LiveClips' approach for segmenting videos more specifically to the domains of photo editing and illustration, making assumptions about the types of actions that were present in videos (*e.g.,* zooming and panning). However, both LiveClips' algorithm for segmenting videos and DiscoverySpace's algorithm for recommending actions would need to be modified to apply to other types of creative tasks, and the user loses access to both systems' contextual support as soon as they leave Photoshop.

As efforts to improve accessibility labeling continue to grow (*e.g.,* [79, 80]) and methods for automatically understanding user interfaces and behaviour via computer vision and machine learning become more sophisticated (*e.g.,* [45,139]), system-wide support will likely become more reliable and robust in the future. However, there will always be some benefit to tailored, domain-specific solutions that leverage an application's particular features. This dissertation demonstrated how in either case, targeting help to the user's context and leveraging expert resources from online communities are key strategies for supporting creative software users in their work.

8.1.3 Improving Navigation and Browsing of Video Resources

Presenting videos in context has limitations, especially space-wise. There is only so much detail one can show when displaying videos alongside the user's work, as they risk obstructing important information if they take up too much space. This was a challenge when designing RePlay, ReMap, and LiveClips. We had to either shrink the entire video to fit inside the contextual interface, which risks making important details too small to see; or crop videos to a smaller area, which risks removing potentially useful content. In both cases, our systems allowed users

to optionally open the video in a larger window, but we did not explore how to take advantage of this extra space to ease navigation and browsing. We have so many useful ways of structuring, organizing, and searching through text resources, but videos still remain difficult to skim and navigate. How might we make videos easier to watch when we are *not* restricted by space, for example in a web browser?

As discussed in Chapter 2, prior work has developed approaches for navigating videos using timeline markers [11, 78, 110, 112, 150], thumbnail images [11, 40, 77, 110, 178, 184], transcript text [110], and clickable elements overlaid on the video [165]. For other types of videos beyond software screencasts, even more creative solutions exist, such as navigating videos by direct manipulation of objects in the video [72], and summarizing videos as concept maps [138] and tapestries [12]. How might we adapt these ideas to creative software videos while leveraging their associated metadata? Are there more useful ways to summarize and display videos than as a single image with a linear timeline? As videos become an increasingly popular medium for consumption in all kinds of domains, future work should continue to explore these questions.

8.1.4 Applicability to Domains Beyond Creative Software Activities

This dissertation focuses specifically on creative software tasks, but the proposed approaches may also apply to any complex task for which expert examples or resources exist. For example, productivity applications like Microsoft Excel are not typically described as creative but exhibit many of the challenges of complex software, and also have an abundance of resources online, including tutorials and macros[1]. Systems such as Wrangler [104] have shown how contextual suggestions can benefit users of data processing software; enhancing such assistance with expert help and demonstrations may therefore also be helpful.

Beyond software, online help and demonstrations also abound for physical tasks, such as crafting and DIY construction (*e.g.*, `instructables.com`, `pinterest.com`). The methods

[1] For example, `excelchamps.com/blog/useful-macro-codes-for-vba-newcomers`

presented in this dissertation for searching and curating online resources may also hold for such tasks, but the remaining challenge would be to understand the user's context and activity when it takes place outside of a computer. Prior research has introduced methods for detecting physical tool use by augmenting tools with sensors [9, 200], detecting physical user activity with sensors worn on the body [145], tracking physical objects using head-worn cameras [90], and detecting user movement using depth sensors [7]. Future work should explore how we might leverage such information for finding relevant expert resources and presenting them to users in context.

8.2 Closing Remarks

This dissertation introduced four different approaches for curating existing expert help and demonstrations and presenting them to users in context. It contributed algorithms for curating and ranking four different types of expert resource (tutorial videos, live stream videos, action macros, and written feedback), interfaces for presenting curated resources in context, evaluations demonstrating the efficacy of these algorithms and interfaces, and formative knowledge about peoples' use of creative software and resources. These contributions demonstrate the potential of bringing knowledge from user communities to everyone's fingertips in a personalized, contextual way. As creative software continues to be more available and plentiful than ever, my hope is that by leveraging the wealth of knowledge that already exists online, future systems can make creative work less solitary and more collaborative, allowing people to learn from and inspire each other.

Appendix A

Understanding the Motivations of Creative Live Streamers

As part of our formative work exploring the domain of creative live streams (Chapter 5, section 5.3), we conducted interviews with 8 creative streamers. Our goal was to understand streamers' motivations, processes, and challenges. This section presents our interview findings and a discussion of open questions, in the hopes that they may inspire and inform future work on creative live streams.

A.1 Why & How do People Stream Creative Work?

What motivates people to live stream creative work? What challenges do they encounter in the process, and how do these compare with streamers in other domains? We interviewed 8 creative streamers and found that streamers were primarily motivated by sharing and engaging with their audience. However, they find it difficult to connect with their audience while focusing on their work. Additionally, for many, live streaming requires significant effort and behind-the-scenes preparation.

Table A.1: Self-reported background information about the eight creative streamers we interviewed. "Skill" refers to the streamer's skill at the type of creative work they stream. After interviewing the streamers, we determined the structure type of their most frequent streaming style.

	Role	Content	Skill	Frequency	Platform	Primary type	Moderators
P1	Freelance Digital Illustrator	Digital Illustration	Expert	3 times / week	Twitch	Making	Yes
P2	Video Editor & Educational Content Maker	Q&A, Analyzing Videos	Expert	Occasional	YouTube, Instagram	Teaching	Yes
P3	Artist / Musician	Music Improvisation	Expert	Occasional	Facebook, Instagram	Performing	No
P4	Drawing Hobbyist	Digital Drawing	Intermediate	Monthly	Twitch	Making	No
P5	Drawing Hobbyist	Digital Drawing	Intermediate	Monthly	Twitch, previously Picarto	Making	No
P6	Adobe XD Evangelist	UX Design	Expert	Daily - Weekly	YouTube, Facebook, previously Twitch	Teaching/Making	Yes
P7	Adobe XD Evangelist	UX & Graphic Design	Expert	3 times / week	YouTube, Periscope, Facebook	Teaching/Making	Yes
P8	Adobe Designer	UX & Graphic Design	Expert	Daily - Weekly	YouTube, previously Twitch	Teaching/Making	Yes

A.1.1 Interview methodology

We recruited 8 streamers (4 male, 4 female, ages 20-45) from personal and professional connections for one-hour semi-structured interviews. We interviewed people across creative disciplines and experience with streaming (Table A.1). We asked participants about their current position and background, process and motivation for streaming, challenges and successes they have experienced, and strategies for engaging with their audience. Three of the participants also host for *Adobe Live*; we asked them about their experience hosting as well as streaming. We took notes and recorded every interview, and analyzed them by comparing participants' answers and identifying common patterns. Interviews were conducted over video chat (4), audio chat (2), or in person (2). Each participant received a $15 gift card for their time.

A.1.2 About the streamers

P1 is a freelance artist who began streaming her work full-time on Twitch in 2016. For the first two years, she streamed for 20-25 hours a week and spent the rest of her time on stream-related preparation. At this commitment level, streaming was her primary source of income. Income on platforms like Twitch mainly derives from ad revenue, viewer subscriptions, and donations. Over time, this became exhausting and felt unsustainable. *P1* took a break, and now

streams casually 3 times a week but not as a primary source of income. Her live stream setup includes a camera view of her face, a screencast of her work, a Stream Deck (Figure A.1), and two monitors for her to see chat activity and other information.

P2 is a video editor and creator who has been making video tutorials on photo and video editing for about 7 years. He hosts a podcast where he interviews people about their creative approach and life stories. He has tried Periscope, and began streaming on YouTube when it enabled mobile streaming in 2017: casual streaming was on the rise. Occasionally he live streams on YouTube or Instagram, answering viewer questions, teaching a particular topic, analyzing a popular video, or critiquing viewers' work. His setup comprises a camera view of his face, a screencast of his work, and a large monitor for him to see chat activity and other information.

P3 is a musician who live streams on Facebook and Instagram (with three band members). Her streams are spontaneous and improvisational; the quartet does not play together regularly but they have a fan base that they stream to whenever they are together. These streams require little setup; they are broadcast from a single mobile phone either held by a friend or propped up. She also occasionally streams product reviews and behind-the-scenes views of her shows.

P4 and *P5* are hobby artists who stream digital drawing about once a month on Twitch. They both started streaming 2 or 3 years ago. *P5* used to stream on Picarto, and moved to Twitch

Figure A.1: The Stream Deck is a programmable control pad used by many streamers, including *P1*, for easy access to common shortcuts and actions. It integrates with Open Broadcaster Software (OBS), a program used by many streamers to host their live streams (`elgato.com/en/gaming/stream-deck`).

about a month ago because it was easier to use and tends to attract more viewers as a better-known platform. *P5* rarely talks out loud during her streams (only when nobody else is home) and *P4* never does. Instead, they engage with viewers by typing in the chat. Neither shows their face when streaming; their setups include only a screencast of their drawing window.

P6, *P7*, and *P8* stream as part of their jobs at Adobe by hosting artists, streaming their own work, and teaching Adobe products. *P6* has been making video tutorials on photo editing for over 10 years. He briefly tried streaming on Twitch but found that his audience did not transfer over to the new platform. He has been streaming with Adobe for approximately 6 months. *P7* taught courses and training programs on design and illustration software for many years. He has been working at Adobe for 9 years, and streaming with Adobe for about 4 years. *P8* is a designer and trained illustrator and has been streaming with Adobe for 4 years. Before joining Adobe, she used to occasionally stream her art process on Twitch. By virtue of live streaming professionally, all three participants have fairly sophisticated technology setups, including a camera view of their face in front of a green screen, a screencast of their computer, several displays for them to see chat activity and other information, and sometimes additional cameras (Figure A.2).

A.1.3 Findings

Audience engagement is a primary goal for streamers

Like with gaming [83,181] and culture-sharing [142] live streams, audience engagement is important to creative streamers. All participants said they engage with their audience during streams, despite their different personalities and streaming styles. When asked about their main motivation for streaming, participants mentioned creating a space for people to hang out together, building an audience, sharing their process with others, and engaging in meaningful conversations.

When asked for an example of a rewarding or enjoyable moment, every participant mentioned audience engagement in some way. Three participants mentioned feeling rewarded by

Figure A.2: The technical setup for *Adobe Live*. (a) The artist (left) and host (right) sit in front of a green screen, with both computers connected for screencasting. (b) Behind the scenes, at least one person helps with technical support, including setup, testing, and monitoring. (c) The artist and host see a display with the live chat feed, and (d) a display showing how they currently appear in the live stream.

gratitude from viewers for inspiration and community. This inspiration goes both ways: *P5* mentioned that she has received valuable feedback from a viewer that helped improve her own work. Two participants valued that *"there's something more authentic about [live streaming] ... it allows me to just be myself more authentically and people can pick up on that, they can understand what you're really about in a way that you just can't express via [other modalities]"* (*P2*). *"There are really no mistakes, there's just honesty"* (*P3*).

A key difference between gaming and creative live streams affecting engagement is the scale of the audience. The average live audience size for our participants ranged from about 5 to 1000, with most sitting at the lower end. Popular streamers of video games such as Fortnite or League of Legends often average audiences of between 2000 and 40,000. This means that creative streamers often feel a tighter personal connection with their viewers, while viewers of gaming streams tend to mostly interact with each other, as the chat goes too quickly for the streamer to keep up [95, 131].

P7 expressed a desire to offer the audience more diverse interactive experiences beyond

just text chat. Currently, gaming live streams sometimes host "audience participation games" [71]. *P1* often organizes games during live streams, such as contests with prizes, voting on what she should do next, or "prompt games" where viewers contribute ideas. *P4* did a "request stream", where he drew whatever viewers requested, and *P2* often runs Q&A-form streams, where he will open an application and just let the audience ask questions. As he put it, *"I want to do what they want to do." Adobe Live* often has giveaways for audience participation, and Adobe hosts a *Daily Creative Challenge*. This kind of engagement *"make[s] it a collaborative thing"* (*P6*), increasing audience investment.

One emerging practice is live streaming portfolio critiques. Like a call-in radio show or newspaper advice column, a few people get direct feedback, and many people benefit through over-the-shoulder learning. This form of learning can be extremely beneficial [141]; it is notable that there is a streaming audience that seeks it out. Similar to shows and columns, streamers have the challenge of selecting which submission(s) to critique. *P2* initially handled this with chat but was quickly overwhelmed by the number of messages. To address this, he found and installed a widget[1] to help him select submissions and allow users to pay a small amount to have their work critiqued. While valuable, it takes time and effort to manage such tools. This also exacerbates streaming's already "fragmented technology ecosystem" [142].

Aside from the three Adobe participants who stream as part of their jobs, none of the participants currently stream as a major income source. Though these participants were not *primarily* motivated by monetary gain, two mentioned that it was a significant secondary benefit, *e.g.*, *P4* said, *"it doesn't matter how good my work gets if I don't actually market myself."* Many streamers in other domains (especially video games) also aim to grow their audience and make money [181]. *P1*'s sought to eventually be a self-sustaining artist; she emphasized that her primary goal was building the audience and creating a positive community; *"I believe that the audience brings [financial benefits]." P3* wished it was easier for viewers to donate. Compensation is possible on some

[1] `streamlabs.com/widgets`

platforms (*e.g.*, Twitch) but requires configuration.

Moderators & hosts alleviate common challenges for artists

A big part of engaging with the audience is interacting via the chat window with viewers' questions, comments, and feedback. Most participants said they sometimes have trouble keeping up with the chat as it requires switching focus from their creative work. This split-attention challenge echoes previous findings for programming [52] and culture-sharing [142] streams. *P5* even said, *"I usually put a warning beforehand that I'm not the most talkative while I'm drawing but I try to check up on the chat as often as I can."* For *P3* who streams on a smartphone, it is even harder to pay attention to chat, as it requires stopping her performance and coming up close to the camera.

Moderators are one way to alleviate this challenge for viewers. In large gaming live streams, trolling is common; many streamers have dedicated moderators whose main role is to ban or time-out people posting inappropriate content and enforce a streamer's community guidelines [140, 202, 203, 241]. Trolls are seen less often in smaller live streams, yet still appear; 5/8 participants have dedicated moderators. As prior work has shown [140, 203, 241], employing successful moderators requires significant preparation; streamers must work with moderators to develop guidelines, and they must constantly work to make sure their judgments align. *P1* and *P2* echoed these challenges: *P1* has spent significant time making a document of guidelines for her moderators. *P2* has not, and as a result finds that their judgments do not always align: *"They might want to ban someone that I think is fine, or they might not ban someone that I think should be banned."*

While moderators can help enforce basic rules and keep the chat constructive, they usually do not support streamers' *engagement* with their audience. Viewers often have questions, feedback, and suggestions for the streamer; these are easily missed. Some moderators do engage in the chat [241] but they require training in order to answer questions on behalf of the streamer

(*e.g.*, *P1*'s moderators). In addition, some streamers find it difficult to talk out loud to their viewers: both *P4* and *P5* said they would like to have others to talk with, as they did not want to fill the silence alone; *"I'm mostly intimidated by the idea of me having to fill a lot of void space"* (*P4*). While *P4* used Discord and *P5* sometimes used join.me for voice chat, these require extra work on the part of the streamer, and sit outside of the main live stream platform.

A different facilitation role that Adobe streams employ to address these challenges are *hosts*. *Adobe Live* streams feature paid hosts who keep the artist and audience engaged, help artists feel confident, and help them focus on their work. The host watches chat messages come in, says hi and responds out loud to viewers' messages, and decides which of viewers' questions to ask to the artist. As *P7* put it, the host is the *"representative for the chat."* Hosts strive to keep viewers engaged by asking the chat questions and including viewers' names when they respond to them. Hosts also strive to keep the *artist* engaged and talking. As *P6* put it, *"the last thing you want is dead silence."* This can be difficult when the artist is shy or quiet, so hosts have picked up tricks such as asking the artist questions about themselves, choosing questions from the chat that are likely to start a conversation, and switching the feed briefly over to their computer to show a relevant tip or trick.

Different platforms bring different audiences

Some participants stream on multiple platforms. Some start streaming on one platform and then switch to another. This brought up interesting trade-offs between different types of live streaming platforms. Besides mobile platforms being simpler than desktop, different platforms also bring different audiences. *P6* and *P8* used to stream on Twitch before Adobe's live streaming community started. They explained that Twitch is a general-purpose platform dedicated to live streaming. It attracts people who generally enjoy live streams and may be interested in creative work but are less often professional. On Twitch it's harder to attract people who are less familiar with live streams, perhaps because of unique specific features such as "emotes"; as *P1* explained,

"if you are in the ecosystem you're really happy with it, and if you're not in the ecosystem it's bizarre."

Adobe Live is an example of a professionally-managed livestream aimed at a company's customers. As a result, it tends to attract aspiring designers and creatives who use the software being shown and want to learn how to produce better work. It also attracts more people who are not familiar with live streaming, as it is shown on platforms that also include other forms of media (Behance and YouTube). A challenge with platforms like this is that *"people might not really get ... why watch a live stream*" (*P2*), as it is not yet widely understood.

Finally, platforms like Picarto focus specifically on *creative* live streaming. These attract viewers dedicated to the topic, which can make conversations more focused. The challenge with specific platforms like these (at least in an era where the phenomenon is still growing) is that fewer people have heard of them, so it can be harder to attract viewers. For this reason, *P5* switched to Twitch. Indeed, Picarto generally has 100-200 streams live at any given time, which is considerably less than the Art section alone on Twitch (which has over 300).

The type of creative work being done also affects the audience. For participants who do visual art such as drawing or use creative software, their streams tend to be of the Making or Teaching type, and their audiences mainly comprise other artists or people interested in learning the skill. For *P3* who streams Performing content on Facebook, her audience mainly consists of friends and fans. These viewers enjoy watching the performance and being a part of live music, but are not necessarily looking to learn music themselves.

Amount of preparation depends on stream type & preferences

While gaming live streamers can simply turn on their screencast and begin playing a game, creative streaming often requires more preparation. 6/8 participants said they prepare before beginning a live stream. Four of these primarily run Teaching streams; the other two primarily run Making streams. For Teaching streams (*P2*, *P6*, *P7*, and *P8*), the streamers spend time before the stream going over what they will show. For Making streams, *P1* and *P5* spend

time on the early stages of their creative work. In addition to preparing content, live streaming (especially on desktop platforms) also requires technical setup. Most participants who stream on desktop platforms said this takes time: setting up cameras and microphones, organizing windows across multiple displays, and testing the output.

Most Teaching streams require some content preparation, much like how course instructors make lesson plans. Socializing streams likely require little-to-no preparation, as the content of these streams is mainly driven by conversation with viewers. For example, *P2* sometimes streams casual Q&A streams on Instagram, enjoying their spontaneous nature: *"you just go live."* For Making and Performing, preparation time depends on the streamer. Some streamers also announce beforehand when they will stream so that viewers can plan to tune in. For casual Performing streams like *P3*'s, all she has to do is turn on the camera and position it. But rehearsed performances require practice beforehand. *P4* said he typically only plans his Making streams 5 minutes before starting, and will start drawing from scratch on the stream. *P8*, who used to do more Making streams, also did not prepare: *"it's as if I am opening up my sketchbook and my friends are there."* Other Making streamers like *P1* and *P5* prefer to start their work before beginning a stream.

Several participants emphasized that some activities make for more engaging live streams than others. Both *P1* and *P5* said their streams are most successful when they do initial sketching beforehand, then spend the stream filling it in and coloring. This is because the early ideation stages involve more problem-solving and deep thinking: *"to be able to put that full energy ... to get through the failures and to find the successes – I can't multitask it." P5* also felt this early stage was less appealing for audiences: *"For a long while they're going to have to look at a blank sheet of white 'paper' so they don't really see the sketches right away ... I think that loses their attention." P2*, when asked why he doesn't live stream his video editing process, he said he tried it but it was too difficult to focus: *"when you're video editing you need to listen to music and focus ... when you're streaming you need to be engaging with the chat."* This echoes previous findings for knowledge-sharing [144] and

programming [52] streams; streamers often prepare beforehand to ensure that the content being streamed will be entertaining for viewers and will not require too much focus on the streamer's part.

Permanence of live stream archives affects performance

We found that the ability to archive live streams significantly affects how streamers perform. Several interviewees mentioned that attentiveness to viewers of a future recording influenced their choices in the moment. *P7* said he sometimes records learning-focused live streams that are meant to be useful as replays, and he interacts less with viewers during those streams. *P2* often deletes or hides his completed live streams because they look less polished than his regular tutorial videos. *P4* and *P5* don't archive their videos, as *"[live streaming is] more of a in-the-moment [thing]"* (*P5*).

A.2 Open Questions and Opportunities

This chapter's interviews as well as Chapter 5's surveys uncovered several challenges many streamers and viewers face, often due to a mismatch of goals between streamers, viewers, and existing streaming platforms. In this section, we highlight areas for future research by asking three open questions.

A.2.1 How might creative live streams better engage viewers?

In line with prior work, we found that creative streamers primarily interact with audiences through live text chat. Most interview participants mentioned difficulty keeping up with this chat, even though these streams are generally much smaller than video gaming streams. Sometimes, conflicting viewer goals can hinder the chat experience. Learners' questions can get lost in the many lines of text written by viewers who are there for social engagement. Streamers often

enhance chat interaction using chat bots (one popular example is Nightbot[2]) and install widgets to provide contests and other interactivity, but these take extra work to create, integrate, and manage.

Augment chat functionality

One approach for enhancing streamer-audience engagement might be to provide separate channels for different types of chat (as one survey participant suggested). For example, learners could post questions in one channel while social banter happens in another.

Another approach could be to design more ways for viewers to communicate beyond text. Novel streaming interfaces allow audiences to participate in video games alongside a streamer, by drawing directly on the streamer's screen to suggest moves and voting on the streamer's next move [131], or even participating directly in the game as a side character [71]. Creative live streams may especially benefit from similar interactions. For example, viewers could annotate a streamer's work directly to ask a question about a particular section or provide feedback. Streaming platforms could also make it easier for streamers to set up polls without needing to spend too much extra time preparing them (*e.g.,* detecting when the streamer poses a question and automatically creating a poll).

Democratize the role of a host

As our interviews demonstrated, having an extra person present on a stream as a host can be immensely helpful for artists. Having someone always watching the chat can alleviate this responsibility from the streamer when there are a large number of viewers, but even when viewership is small, having someone to ask questions and engage the artist in discussion can help keep a stream interesting. While moderators can address some of these challenges, by current conventions they typically do not, and not all streamers have the time or experience to find and

[2] `nightbot.tv`

train reliable helpers (*e.g., P2*). How might we democratize the experience of having supportive hosts or facilitators?

One solution could be to take advantage of the auditory modality to mitigate the limited attention and screen space that streamers have when focusing on creative work. *P4* suggested a text-to-voice service to read out chat messages so he doesn't have to look up from his work to answer questions, but noted that such a service would need to understand his own preferences so it could appropriately "triage" the chat, highlighting only important or relevant questions and comments. Such a system might also help streamers feel more like they are participating in a conversation rather than one-way communication.

Another challenge is to create systems to help streamers troubleshoot their technical setup in lieu of trusted moderators or hosts. While streaming from mobile platforms has become as easy as pressing "go live," many interview participants described spending a lot of time experimenting with technical settings to ensure that screencasts and camera views are clear and detailed, audio is on and good quality, and background music is at an appropriate volume. Three interview participants mentioned that a system that could automatically help with this setup (or give feedback on the quality of their setup) would save substantial time and effort.

Allow searching by goal

Future work might explore how to match audiences to the right streams in the first place, like Sjöblom *et al.* suggest for gaming [216]. Current platforms typically allow viewers to find streams based on textual metadata, like the category and title. Platforms could allow streamers to make their goals for a stream explicit and searchable, so that those seeking to learn new skills could easily find Teaching streams, and those seeking to hang out with others could directly visit Socializing streams. In addition, platforms could enable or disable different modes of audience interaction depending on a stream's goals. Future research could explore what kinds of audience interaction best benefit different types of streams.

A.2.2 How might we make creative work more "performable"?

Several interview participants mentioned that they were not comfortable streaming certain parts of their creative process, because they worried it would not engage the audience, or because it required their full focus. They would instead work on these parts offline to prepare for streams. Programming streamers face a similar tension between sharing their realistic process (including difficult and less-exciting parts like debugging) and keeping the audience entertained [52]. Some artists (like *P4*) are comfortable sharing their entire process from the beginning. For many viewers, it can be inspiring and educational to watch an artist go through the early ideation stages, but these parts of the process may need extra work to explain to audiences, as they feature a lot of internal reflection and messy iteration [198]. It is possible that more automated facilitation (as discussed previously) may allow artists to focus more on their work when it needs their full attention. How else might streaming platforms better support sharing *all* parts of the creative process? Are there ways to make the early stages of creative work more "performable" for audiences?

A.2.3 How might we support watching live stream archives?

LiveClips (Chapter 5) introduced one potential way for viewing live stream archives: extracting short clips and recommending them to users in the context of their creative software. While this might help people find in-the-moment inspiration, what about people who want to learn from a Teaching stream, or catch up on their community's activity in a Socializing stream? What are other ways we might improve the viewing experience of live stream archives?

On some platforms, such as Instagram, live streams disappear shortly after they finish. On others, like YouTube and Facebook, they are re-playable archives that show up in search results alongside other videos. In between, platforms like Twitch archive videos for a limited time (14 or 60 days depending on account type); these archives are rarely re-watched [101], as the affordances for finding them are limited. While popular live streams on YouTube continue

to accrue views after they are archived, several survey participants (section 5.3) mentioned the viewer experience is poor because the videos are long, have limited navigation, and include long periods of downtime and conversation with the then-live chat [143]. Twitch viewers can create "clips" and streamers can create "highlights" of interesting moments, but they must remember to do so, and such moments can seem out-of-context when viewed on their own.

In recent work [62], we introduced a method for automatically segmenting live stream archives into navigable sections based on the streamer's creative process. Such segmentations may help viewers more easily find specific sections in a live stream archive or glean a quick summary of what happened without needing to watch the entire video. As another example, StreamWiki [143] enables viewers to collaboratively summarize a Teaching stream as it occurs. Future work should continue to explore novel ways for browsing and navigating creative live stream archives as they become increasingly prevalent online.

A.3 Acknowledgements

This chapter, in part, includes portions of material as it appears in *Sharing the Studio: How Creative Livestreaming can Inspire, Educate, and Engage* by C. Ailie Fraser, Joy O. Kim, Alison Thornsberry, Scott Klemmer, and Mira Dontcheva in the Proceedings of the 2019 on Creativity and Cognition (C&C '19). The dissertation author was the primary investigator and author of this paper.

Bibliography

[1] 2013. Watch Before You Click: Smart Motion Preview. (2013). https://blogs.bing.com/search/2013/05/07/watch-before-you-click-smart-motion-preview

[2] 2017. *Cisco Visual Networking Index: Forecast and Methodology, 2016-2021*. Technical Report. Cisco. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[3] 2017. Turnitin Feedback Studio. (2017). http://turnitin.com/en

[4] Mark S. Ackerman and Thomas W. Malone. 1990. Answer Garden: a tool for growing organizational memory. In *Proceedings of the conference on Office information systems -*. ACM Press, New York, New York, USA, 31–39. DOI:http://dx.doi.org/10.1145/91474.91485

[5] Eytan Adar, Mira Dontcheva, and Gierad P. Laput. 2014. CommandSpace: Modeling the Relationships between Tasks, Descriptions and Features. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST'14*. ACM Press, New York, New York, USA, 167–176. DOI:http://dx.doi.org/10.1145/2642918.2647395

[6] Samiul Alam Anik. 2015. *Integrating Comments in Video Tutorials*. Ph.D. Dissertation. https://mspace.lib.umanitoba.ca/handle/1993/31046

[7] Fraser Anderson, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2013. 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*. ACM Press, New York, New York, USA, 311–320. DOI:http://dx.doi.org/10.1145/2501988.2502045

[8] Heidi Goodrich Andrade. 2005. Teaching With Rubrics: The Good, the Bad, and the Ugly. *College Teaching* 53, 1 (jan 2005), 27–31. DOI:http://dx.doi.org/10.3200/CTCH.53.1.27-31

[9] Stavros Antifakos, Florian Michahelles, and Bernt Schiele. 2002. Proactive Instructions for Furniture Assembly. In *UbiComp 2002: Ubiquitous Computing*. Springer Berlin Heidelberg, 351–360. DOI:http://dx.doi.org/10.1007/3-540-45809-3_27

[10] Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K Roy, and Kevin A Schneider. 2013. Answering Questions about Unanswered Questions of Stack Overflow. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR '13)*. IEEE Press, 97–100.

[11] Nikola Banovic, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2012. Waken: Reverse engineering usage information and interface structure from software videos. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*. ACM Press, New York, New York, USA, 83. DOI:http://dx.doi.org/10.1145/2380116.2380129

[12] Connelly Barnes, Dan B. Goldman, Eli Shechtman, and Adam Finkelstein. 2010. Video tapestries with continuous temporal zoom. *ACM Transactions on Graphics* 29, 4 (jul 2010), 1–9. DOI:http://dx.doi.org/10.1145/1778765.1778826

[13] David Bawden. 1986. Information systems and the stimulation of creativity. *Journal of Information Science* 12, 5 (aug 1986), 203–216. DOI:http://dx.doi.org/10.1177/016555158601200501

[14] Michel Beaudouin-Lafon and Wendy E. Mackay. 2018. Rethinking Interaction: From Instrumental Interaction to Human-Computer Partnerships. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1–5. DOI:http://dx.doi.org/10.1145/3170427.3170635

[15] A. Begel and S.L. Graham. 2006. An Assessment of a Speech-Based Programming Environment. In *Visual Languages and Human-Centric Computing (VL/HCC'06)*. IEEE, 116–120. DOI:http://dx.doi.org/10.1109/VLHCC.2006.9

[16] William Benjamin, Senthil Chandrasegaran, Devarajan Ramanujan, Niklas Elmqvist, SVN Vishwanathan, and Karthik Ramani. 2014. Juxtapoze: supporting serendipity and creative expression in clipart compositions. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 341–350. DOI:http://dx.doi.org/10.1145/2556288.2557327

[17] Floraine Berthouzoz, Wilmot Li, Mira Dontcheva, and Maneesh Agrawala. 2011. A Framework for Content-Adaptive Photo Manipulation Macros: Application to Face, Landscape, and Global Manipulations. *ACM Transactions on Graphics* 30, 5 (oct 2011), 1–14. DOI:http://dx.doi.org/10.1145/2019627.2019639

[18] Benjamin S. Bloom. 1984. The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring. *Educational Researcher* 13, 6 (jun 1984), 4–16. DOI:http://dx.doi.org/10.3102/0013189X013006004

[19] Richard A. Bolt. 1980. "Put-that-there": Voice and gesture at the graphics interface. *ACM SIGGRAPH Computer Graphics* 14, 3 (jul 1980), 262–270. DOI:http://dx.doi.org/10.1145/965105.807503

[20] Charles C. Bonwell and James A. Eison. 1991. *Active learning: Creating excitement in the classroom*. School of Education and Human Development, George Washington University. 104 pages. https://eric.ed.gov/?id=ED336049

[21] Horatiu Bota, Adam Fourney, Susan T. Dumais, Tomasz L. Religa, and Robert Rounthwaite. 2018. Characterizing Search Behavior in Productivity Software. In *Proceedings of the 2018 Conference on Human Information Interaction&Retrieval - CHIIR '18*. ACM Press, New York, New York, USA, 160–169. DOI:http://dx.doi.org/10.1145/3176349.3176395

[22] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. 2010. Example-centric programming: Integrating Web Search into the Development Environment. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, New York, New York, USA, 513. DOI:http://dx.doi.org/10.1145/1753326.1753402

[23] Frederick P. Brooks. 1987. No Silver Bullet– Essence and Accidents of Software Engineering. *IEEE Computer* 20, 4 (apr 1987), 10–19. DOI:http://dx.doi.org/10.1109/MC.1987.1663532

[24] Michael Brooks, Sumit Basu, Charles Jacobs, and Lucy Vanderwende. 2014. Divide and correct: using clusters to grade short answers at scale. In *Proceedings of the first ACM conference on Learning @ scale conference - L@S '14*. ACM Press, New York, New York, USA, 89–98. DOI:http://dx.doi.org/10.1145/2556325.2566243

[25] Jake Brutlag. 2009. *Speed Matters for Google Web Search*. Technical Report. http://services.google.com/fh/files/blogs/google

[26] Andrea Bunt, Patrick Dubois, Ben Lafreniere, Michael A. Terry, and David T. Cormack. 2014. TaggedComments: promoting and integrating user comments in online application tutorials. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 4037–4046. DOI:http://dx.doi.org/10.1145/2556288.2557118

[27] Margaret Burnett, Simone Stumpf, Jamie Macbeth, Stephann Makri, Laura Beckwith, Irwin Kwan, Anicia Peters, and William Jernigan. 2016. GenderMag: A Method for Evaluating Software's Gender Inclusiveness. *Interacting with Computers* 28, 6 (nov 2016), 760–787. DOI:http://dx.doi.org/10.1093/iwc/iwv046

[28] William Buxton. 1986. Chunking and Phrasing and the Design of Human-Computer Dialogues. In *Proceedings of the IFIP World Computer Congress*. Dublin, Ireland, 475–480.

[29] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Fredo Durand. 2011. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR 2011*. IEEE, 97–104. DOI:http://dx.doi.org/10.1109/CVPR.2011.5995413

[30] John M. Carroll and Caroline Carrithers. 1984. Training wheels in a user interface. *Commun. ACM* 27, 8 (aug 1984), 800–806. DOI:http://dx.doi.org/10.1145/358198.358218

[31] John M. Carroll and Mary Beth Rosson. 1987. Paradox of the active user. In *Interfacing thought: Cognitive aspects of human-computer interaction*, John M. Carroll (Ed.). MIT Press, Cambridge, 80–111.

[32] Joel Chan, Steven P Dow, and Christian D Schunn. 2015. Do the best design ideas (really) come from conceptually distant sources of inspiration? *Design Studies* 36 (2015), 31–58. DOI:http://dx.doi.org/10.1016/j.destud.2014.08.001

[33] Joel Chan, Katherine Fu, Christian Schunn, Jonathan Cagan, Kristin Wood, and Kenneth Kotovsky. 2011. On the Benefits and Pitfalls of Analogies for Innovative Design: Ideation Performance Based on Analogical Distance, Commonness, and Modality of Examples. *Journal of Mechanical Design* 133, 8 (aug 2011), 081004. DOI:http://dx.doi.org/10.1115/1.4004396

[34] Joel Chan, Pao Siangliulue, Denisa Qori McDonald, Ruixue Liu, Reza Moradinezhad, Safa Aman, Erin T. Solovey, Krzysztof Z. Gajos, and Steven P. Dow. 2017. Semantically Far Inspirations Considered Harmful?: Accounting for Cognitive States in Collaborative Ideation. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition - C&C '17*. ACM Press, New York, New York, USA, 93–105. DOI:http://dx.doi.org/10.1145/3059454.3059455

[35] Minsuk Chang, Anh Truong, Oliver Wang, Maneesh Agrawala, and Juho Kim. 2019. How to Design Voice Based Navigation for How-To Videos. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, New York, New York, USA, 1–11. DOI:http://dx.doi.org/10.1145/3290605.3300931

[36] Tsung-Hsiang Chang, Tom Yeh, and Rob Miller. 2011. Associating the visual representation of user interfaces with their internal structures and metadata. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, New York, New York, USA, 245. DOI:http://dx.doi.org/10.1145/2047196.2047228

[37] Siddhartha Chaudhuri and Vladlen Koltun. 2010. Data-driven suggestions for creativity support in 3D modeling. In *ACM SIGGRAPH Asia 2010 papers on - SIGGRAPH ASIA '10*, Vol. 29. ACM Press, New York, New York, USA, 1. DOI:http://dx.doi.org/10.1145/1866158.1866205

[38] Yan Chen, Sang Won Lee, Yin Xie, YiWei Yang, Walter S. Lasecki, and Steve Oney. 2017. Codeon: On-Demand Software Development Assistance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 6220–6231. DOI:http://dx.doi.org/10.1145/3025453.3025972

[39] Gifford Cheung and Jeff Huang. 2011. Starcraft from the stands: understanding the game spectator. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. ACM Press, New York, New York, USA, 763. DOI:http://dx.doi.org/10.1145/1978942.1979053

[40] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. 2012. MixT: Automatic generation of step-by-step mixed media tutorials. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*. ACM Press, New York, New York, USA, 93. DOI:http://dx.doi.org/10.1145/2380116.2380130

[41] Parmit K. Chilana, Amy J. Ko, and Jacob O. Wobbrock. 2012. LemonAid: Selection-based crowdsourced contextual help for web applications. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. ACM Press, New York, New York, USA, 1549. DOI:http://dx.doi.org/10.1145/2207676.2208620

[42] Kwangsu Cho, Christian D Schunn, and Davida Charney. 2006. Commenting on Writing: Typology and Perceived Helpfulness of Comments from Novice Peer Reviewers and Subject Matter Experts. *Written Communication* 23, 3 (2006), 260–294.

[43] Herbert H. Clark. 2006. Context and Common Ground. In *Encyclopedia of Language & Linguistics, Second Edition*. Elsevier, 105–108. DOI:http://dx.doi.org/10.1016/B0-08-044854-2/01088-9

[44] Philip R. Cohen, Mary Dalrymple, Douglas B. Moran, Fernando C. Pereira, and Joseph W. Sullivan. 1989. Synergistic use of direct manipulation and natural language. In *Proceedings of the SIGCHI conference on Human factors in computing systems Wings for the mind - CHI '89*. ACM Press, New York, New York, USA, 227–233. DOI:http://dx.doi.org/10.1145/67449.67494

[45] Morgan Dixon and James Fogarty. 2010. Prefab: Implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, New York, New York, USA, 1525. DOI:http://dx.doi.org/10.1145/1753326.1753554

[46] Morgan Dixon, Alexander Nied, and James Fogarty. 2014. Prefab layers and prefab annotations: extensible pixel-based interpretation of graphical interfaces. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. ACM Press, New York, New York, USA, 221–230. DOI:http://dx.doi.org/10.1145/2642918.2647412

[47] Mira Dontcheva, Robert R. Morris, Joel R. Brandt, and Elizabeth M. Gerber. 2014. Combining crowdsourcing and learning to improve engagement and performance. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 3379–3388. DOI:http://dx.doi.org/10.1145/2556288.2557217

[48] Steven P. Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L. Schwartz, and Scott R. Klemmer. 2010. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. *ACM Transactions on Computer-Human Interaction* 17, 4 (dec 2010), 1–24. DOI:http://dx.doi.org/10.1145/1879831.1879836

[49] Steven P. Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. 2012. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work - CSCW '12*. ACM Press, New York, New York, USA, 1013–1022. DOI:http://dx.doi.org/10.1145/2145204.2145355

[50] Michael Ekstrand, Wei Li, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2011. Searching for software learning resources using application context. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, New York, New York, USA, 195. DOI:http://dx.doi.org/10.1145/2047196.2047220

[51] Sandra Erdelez. 1999. Information Encountering: It's More Than Just Bumping into Information. *Bulletin of the American Society for Information Science and Technology* 25, 3 (feb 1999), 26–29. DOI:http://dx.doi.org/10.1002/bult.118

[52] Travis Faas, Lynn Dombrowski, Alyson Young, and Andrew D. Miller. 2018. Watch Me Code: Programming Mentorship Communities on Twitch.tv. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (nov 2018), 1–18. DOI:http://dx.doi.org/10.1145/3274319

[53] Ethan Fast, Colleen Lee, Alex Aiken, Michael S. Bernstein, Daphne Koller, and Eric Smith. 2013. Crowd-scale interactive formal reasoning and analytics. In *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST '13*. 363–372.

[54] Leah Findlater and Joanna McGrenere. 2010. Beyond performance: Feature awareness in personalized interfaces. *International Journal of Human-Computer Studies* 68, 3 (mar 2010), 121–137. DOI:http://dx.doi.org/10.1016/j.ijhcs.2009.10.002

[55] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2002. Placing search in context: the concept revisited. *ACM Transactions on Information Systems* 20, 1 (jan 2002), 116–131. DOI:http://dx.doi.org/10.1145/503104.503110

[56] Allen Foster and Nigel Ford. 2003. Serendipity and information seeking: an empirical study. *Journal of Documentation* 59, 3 (jun 2003), 321–340. DOI:http://dx.doi.org/10.1108/00220410310472518

[57] Adam Fourney and Susan T. Dumais. 2016. Automatic Identification and Contextual Reformulation of Implicit System-Related Queries. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16*. ACM Press, New York, New York, USA, 761–764. DOI:http://dx.doi.org/10.1145/2911451.2914701

[58] Adam Fourney, Ben Lafreniere, Parmit Chilana, and Michael Terry. 2014. InterTwine: creating interapplication information scent to support coordinated use of software. In *Proceedings of the 27th annual ACM symposium on User interface software and technology - UIST '14*. ACM Press, New York, New York, USA, 429–438. DOI:http://dx.doi.org/10.1145/2642918.2647420

[59] Adam Fourney, Ben Lafreniere, Richard Mann, and Michael Terry. 2012. "Then click ok!": extracting references to interface elements in online documentation. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems - CHI '12*. ACM Press, New York, New York, USA, 35. DOI:http://dx.doi.org/10.1145/2207676.2207682

[60] Adam Fourney, Richard Mann, and Michael Terry. 2011. Query-feature graphs: bridging user vocabulary and system functionality. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, New York, New York, USA, 207. DOI:http://dx.doi.org/10.1145/2047196.2047224

[61] Adam Fourney and Michael Terry. 2014. Mining Online Software Tutorials: Challenges and Open Problems. In *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems - CHI EA '14*. ACM Press, New York, New York, USA, 653–664. DOI:http://dx.doi.org/10.1145/2559206.2578862

[62] C. Ailie Fraser, Joy O. Kim, Hijung Valentina Shin, Joel Brandt, and Mira Dontcheva. 2020. Temporal Segmentation of Creative Live Streams. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems - CHI '20 (to appear)*. ACM, New York, New York, USA.

[63] Krzysztof Z. Gajos, Mary Czerwinski, Desney S. Tan, and Daniel S. Weld. 2006. Exploring the design space for adaptive graphical user interfaces. In *Proceedings of the working conference on Advanced visual interfaces - AVI '06*. ACM Press, New York, New York, USA, 201. DOI:http://dx.doi.org/10.1145/1133265.1133306

[64] Geri. Gay and Helene. Hembrooke. 2004. *Activity-centered design: An ecological approach to designing smart tools and usable systems*. MIT Press. 111 pages. https://dl.acm.org/citation.cfm?id=983085

[65] Graham Gibbs and Claire Simpson. 2004. Conditions Under Which Assessment Supports Students' Learning. *Learning and Teaching in Higher Education* 1, 1 (2004).

[66] Sarah Gielen, Elien Peeters, Filip Dochy, Patrick Onghena, and Katrien Struyven. 2010. Improving the effectiveness of peer feedback for learning. *Learning and Instruction* 20, 4 (aug 2010), 304–315. DOI:http://dx.doi.org/10.1016/J.LEARNINSTRUC.2009.08.007

[67] Andreas Girgensohn, John Adcock, Matthew Cooper, and Lynn Wilcox. 2005. A Synergistic Approach to Efficient Interactive Video Retrieval. In *Human-Computer Interaction -*

INTERACT 2005, Maria Francesca Costabile and Fabio Paternò (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 781–794.

[68] Elena L. Glassman, Aaron Lin, Carrie J. Cai, and Robert C. Miller. 2016. Learnersourcing Personalized Hints. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing - CSCW '16*. ACM Press, 1624–1634.

[69] Elena L. Glassman and Daniel M. Russell. 2016. DocMatrix: Self-teaching from multiple sources. *Proceedings of the Association for Information Science and Technology* 53, 1 (jan 2016), 1–10. DOI:http://dx.doi.org/10.1002/pra2.2016.14505301064

[70] Elena L. Glassman, Jeremy Scott, Rishabh Singh, Philip J. Guo, and Robert C. Miller. 2015. OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale. *ACM Transactions on Computer-Human Interaction* 22, 2 (mar 2015), 1–35.

[71] Seth Glickman, Nathan McKenzie, Joseph Seering, Rachel Moeller, and Jessica Hammer. 2018. Design Challenges for Livestreamed Audience Participation Games. In *The Annual Symposium on Computer-Human Interaction in Play Extended Abstracts - CHI PLAY '18*. ACM Press, New York, New York, USA, 187–199. DOI:http://dx.doi.org/10.1145/3242671.3242708

[72] Dan B. Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M. Seitz. 2008. Video object annotation, navigation, and composition. In *Proceedings of the 21st annual ACM symposium on User interface software and technology - UIST '08*. ACM Press, New York, New York, USA, 3. DOI:http://dx.doi.org/10.1145/1449715.1449719

[73] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. 2009. Generating photo manipulation tutorials by demonstration. In *ACM SIGGRAPH 2009 papers on - SIGGRAPH '09*, Vol. 28. ACM Press, New York, New York, USA, 1. DOI:http://dx.doi.org/10.1145/1576246.1531372

[74] Michael D. Greenberg, Matthew W. Easterday, and Elizabeth M. Gerber. 2015. Critiki: A Scaffolded Approach to Gathering Design Feedback from Paid Crowdworkers. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition - C&C '15*. ACM Press, New York, New York, USA, 235–244. DOI:http://dx.doi.org/10.1145/2757226.2757249

[75] Sharon L. Greene. 2002. Characteristics of applications that support creativity. *Commun. ACM* 45, 10 (oct 2002), 100–104. DOI:http://dx.doi.org/10.1145/570907.570941

[76] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. 2007. Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*. ACM Press, New York, New York, USA, 1591. DOI:http://dx.doi.org/10.1145/1240624.1240865

[77] Tovi Grossman and George Fitzmaurice. 2010. ToolClips: An Investigation of Contextual Video Assistance for Functionality Understanding. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, New York, New York, USA, 1515. DOI:http://dx.doi.org/10.1145/1753326.1753552

[78] Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2010. 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*. ACM Press, New York, New York, USA, 143. DOI:http://dx.doi.org/10.1145/1866029.1866054

[79] Anhong Guo, Xiang 'Anthony' Chen, Haoran Qi, Samuel White, Suman Ghosh, Chieko Asakawa, and Jeffrey P. Bigham. 2016. VizLens: A Robust and Interactive Screen Reader for Interfaces in the Real World. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. ACM Press, New York, New York, USA, 651–664. DOI:http://dx.doi.org/10.1145/2984511.2984518

[80] Anhong Guo, Junhan Kong, Michael Rivera, Frank F. Xu, and Jeffrey P. Bigham. 2019. StateLens: A Reverse Engineering Solution for Making Existing Dynamic Touchscreens Accessible. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 371–385. DOI:http://dx.doi.org/10.1145/3332165.3347873

[81] Ido Guy. 2016. Searching by Talking: Analysis of Voice Queries on Mobile Web Search. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16*. ACM Press, New York, New York, USA, 35–44. DOI:http://dx.doi.org/10.1145/2911451.2911525

[82] Lassi Haaranen and Lassi. 2017. Programming as a Performance: Live-streaming and Its Implications for Computer Science Education. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '17*. ACM Press, New York, New York, USA, 353–358. DOI:http://dx.doi.org/10.1145/3059009.3059035

[83] William A. Hamilton, Oliver Garretson, and Andruid Kerne. 2014. Streaming on twitch: fostering participatory communities of play within live mixed media. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 1315–1324. DOI:http://dx.doi.org/10.1145/2556288.2557048

[84] Susan M. Harrison. 1995. A comparison of still, animated, or nonillustrated on-line help with written or spoken instructions in a graphical user interface. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '95*. ACM Press, New York, New York, USA, 82–89. DOI:http://dx.doi.org/10.1145/223904.223915

[85] Björn Hartmann, Daniel MacDougall, Joel Brandt, and Scott R. Klemmer. 2010. What would other programmers do: suggesting solutions to error messages. In *Proceedings of*

the 28th international conference on Human factors in computing systems - CHI '10. ACM Press, New York, New York, USA, 1019. DOI:http://dx.doi.org/10.1145/1753326.1753478

[86] J. Hattie and H. Timperley. 2007. The Power of Feedback. *Review of Educational Research* 77, 1 (mar 2007), 81–112.

[87] Andrew Head, Elena Glassman, Gustavo Soares, Ryo Suzuki, Lucas Figueredo, and Loris D'Antoni. 2017. Writing Reusable Code Feedback at Scale with Mixed-Initiative Program Synthesis. In *Proceedings of Learning at Scale '17*. ACM, 89–98.

[88] Marti A. Hearst. 2006. Design recommendations for hierarchical faceted search interfaces. In *Proc. SIGIR 2006, Workshop on Faceted Search*. 26 – 30.

[89] Marti A. Hearst. 2009. *Search User Interfaces*. Cambridge University Press. 385 pages. https://www.cambridge.org/us/catalogue/catalogue.asp?isbn=9780521113793

[90] Steven J. Henderson and Steven K. Feiner. 2011. Augmented reality in the psychomotor phase of a procedural task. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, 191–200. DOI:http://dx.doi.org/10.1109/ISMAR.2011.6092386

[91] Alex Hern. 2019. Apple contractors 'regularly hear confidential details' on Siri recordings. (jul 2019). https://www.theguardian.com/technology/2019/jul/26/apple-contractors-regularly-hear-confidential-details-on-siri-recordings

[92] Scarlett R. Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P. Bailey. 2009. Getting inspired!: understanding how and why examples are used in creative design practice. In *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*. ACM Press, New York, New York, USA, 87. DOI:http://dx.doi.org/10.1145/1518701.1518717

[93] Catherine M. Hicks, Vineet Pandey, C. Ailie Fraser, and Scott Klemmer. 2016. Framing Feedback: Choosing Review Environment Features that Support High Quality Peer Assessment. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 458–469. DOI:http://dx.doi.org/10.1145/2858036.2858195

[94] Zorah Hilvert-Bruce, James T. Neill, Max Sjöblom, and Juho Hamari. 2018. Social motivations of live-streaming viewer engagement on Twitch. *Computers in Human Behavior* 84 (jul 2018), 58–67. DOI:http://dx.doi.org/10.1016/J.CHB.2018.02.013

[95] Mu Hu, Mingli Zhang, and Yu Wang. 2017. Why do audiences choose to keep watching on live video streaming platforms? An explanation of dual identification framework. *Computers in Human Behavior* 75 (oct 2017), 594–606. DOI:http://dx.doi.org/10.1016/J.CHB.2017.06.006

[96] Jim Hugunin and Victor Zue. 1997. On the design of effective speech-based interfaces for desktop applications. *EUROSPEECH* (1997). https://www.semanticscholar.org/paper/On-the-design-of-effective-speech-based-interfaces-Hugunin-Zue/f9f7099ec9a30afb6f0de494d233551bbe2421c9

[97] Amy Hurst, Scott E. Hudson, and Jennifer Mankoff. 2010. Automatically identifying targets users interact with during real world tasks. In *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '10*. ACM Press, New York, New York, USA, 11. DOI:http://dx.doi.org/10.1145/1719970.1719973

[98] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. 1985. Direct Manipulation Interfaces. *Human–Computer Interaction* 1, 4 (nov 1985), 311–338. DOI:http://dx.doi.org/10.1207/s15327051hci0104_2

[99] Michelle Ichinco, Wint Yee Hnin, and Caitlin L. Kelleher. 2017. Suggesting API Usage to Novice Programmers with the Example Guru. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 1105–1117. DOI:http://dx.doi.org/10.1145/3025453.3025827

[100] Robert J. K. Jacob. 1990. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems Empowering people - CHI '90*. ACM Press, New York, New York, USA, 11–18. DOI:http://dx.doi.org/10.1145/97243.97246

[101] Adele Lu Jia, Siqi Shen, Dick H. J. Epema, and Alexandru Iosup. 2016. When Game Becomes Life: The Creators and Spectators of Online Game Replays and Live Streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications* 12, 4 (aug 2016), 1–24. DOI:http://dx.doi.org/10.1145/2957750

[102] Jiepu Jiang, Wei Jeng, and Daqing He. 2013. How do users respond to voice input errors?: lexical and phonetic query reformulation in voice search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*. ACM Press, New York, New York, USA, 143. DOI:http://dx.doi.org/10.1145/2484028.2484092

[103] Slava Kalyuga, Paul Chandler, and John Sweller. 1999. Managing split-attention and redundancy in multimedia instruction. *Applied Cognitive Psychology* 13, 4 (aug 1999), 351–371. DOI:http://dx.doi.org/10.1002/(SICI)1099-0720(199908)13:4<351::AID-ACP589>3.0.CO;2-6

[104] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. ACM Press, New York, New York, USA, 3363. DOI:http://dx.doi.org/10.1145/1978942.1979444

[105] Lewis R. Karl, Michael Pettey, and Ben Shneiderman. 1993. Speech versus mouse commands for word processing: an empirical evaluation. *International Journal of Man-Machine Studies* 39, 4 (oct 1993), 667–687. DOI:http://dx.doi.org/10.1006/IMMS.1993.1078

[106] Caitlin Kelleher and Randy Pausch. 2005. Stencils-Based Tutorials: Design and Evaluation. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '05*. ACM Press, New York, New York, USA, 541. DOI:http://dx.doi.org/10.1145/1054972.1055047

[107] David Kelley and Tom Kelley. 2013. *Creative confidence : Unleashing the creative potential within us all*. Crown Publishing Group, New York. 288 pages.

[108] Juho Kim. 2013. Toolscape: enhancing the learning experience of how-to videos. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems on - CHI EA '13*. ACM Press, New York, New York, USA, 2707. DOI:http://dx.doi.org/10.1145/2468356.2479497

[109] Joy Kim, Maneesh Agrawala, and Michael S. Bernstein. 2017. Mosaic: Designing Online Creative Communities for Sharing Works-in-Progress. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing - CSCW '17*. ACM Press, New York, New York, USA, 246–258. DOI:http://dx.doi.org/10.1145/2998181.2998195

[110] Juho Kim, Philip J. Guo, Carrie J. Cai, Shang-Wen (Daniel) Li, Krzysztof Z. Gajos, and Robert C. Miller. 2014. 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*. ACM Press, New York, New York, USA, 563–572. DOI:http://dx.doi.org/10.1145/2642918.2647389

[111] Juho Kim, Benjamin Malley, Joel Brandt, Mira Dontcheva, Diana Joseph, Krzysztof Z. Gajos, and Robert C. Miller. 2012. Photoshop with friends: a synchronous learning community for graphic design. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion - CSCW '12*. ACM Press, New York, New York, USA, 271. DOI:http://dx.doi.org/10.1145/2141512.2141598

[112] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. 2014. 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*. ACM Press, New York, New York, USA, 4017–4026. DOI:http://dx.doi.org/10.1145/2556288.2556986

[113] Yea-Seul Kim, Mira Dontcheva, Eytan Adar, and Jessica Hullman. 2019. Vocal Shortcuts for Creative Experts. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, New York, New York, USA, 1–14. DOI:http://dx.doi.org/10.1145/3290605.3300562

[114] Lorenz Cuno Klopfenstein, Saverio Delpriori, Silvia Malatini, and Alessandro Bogliolo. 2017. The Rise of Bots: A Survey of Conversational Interfaces, Patterns, and Paradigms. In *Proceedings of the 2017 Conference on Designing Interactive Systems - DIS '17*. ACM Press, New York, New York, USA, 555–565. DOI:http://dx.doi.org/10.1145/3064663.3064672

[115] Jonathan Koren, Yi Zhang, and Xue Liu. 2008. Personalized interactive faceted search. In *Proceeding of the 17th international conference on World Wide Web - WWW '08*. ACM Press, New York, New York, USA, 477. DOI:http://dx.doi.org/10.1145/1367497.1367562

[116] Reiner Kraft, Farzin Maghoul, and Chi Chao Chang. 2005. Y!Q: contextual search at the point of inspiration. In *Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM '05*. ACM Press, New York, New York, USA, 816. DOI:http://dx.doi.org/10.1145/1099554.1099746

[117] Markus Krause, Tom Garncarz, JiaoJiao Song, Elizabeth M. Gerber, Brian P. Bailey, and Steven P. Dow. 2017. Critique Style Guide: Improving Crowdsourced Design Feedback with a Natural Language Model. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 4627–4639. DOI:http://dx.doi.org/10.1145/3025453.3025883

[118] Chinmay Kulkarni, Steven P Dow, and Scott R Klemmer. 2014. Early and Repeated Exposure to Examples Improves Creative Work. In *Design Thinking Research*, Hasso Plattner, Christoph Meinel, and Larry Leifer (Eds.). Springer International Publishing, 49–62.

[119] Chinmay Kulkarni, Koh Pang Wei, Huy Le, Daniel Chia, Kathryn Papadopoulos, Justin Cheng, Daphne Koller, and Scott R. Klemmer. 2013. Peer and self assessment in massive online classes. *ACM Transactions on Computer-Human Interaction* 20, 6 (dec 2013), 1–31. DOI:http://dx.doi.org/10.1145/2505057

[120] Chinmay E. Kulkarni, Michael S. Bernstein, and Scott R. Klemmer. 2015. PeerStudio: Rapid Peer Feedback Emphasizes Revision and Improves Performance. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale - L@S '15*. ACM Press, New York, New York, USA, 75–84. DOI:http://dx.doi.org/10.1145/2724660.2724670

[121] Benjamin Lafreniere, Andrea Bunt, Matthew Lount, Filip Krynicki, and Michael A. Terry. 2011. AdaptableGIMP: Designing a Socially-Adaptable Interface. In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology - UIST '11 Adjunct*. ACM Press, New York, New York, USA, 89. DOI:http://dx.doi.org/10.1145/2046396.2046437

[122] Benjamin Lafreniere, Andrea Bunt, Matthew Lount, and Michael Terry. 2013. Understanding the Roles and Uses of Web Tutorials. In *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*. Association for the Advancement of Artificial Intelligence.

[123] Benjamin Lafreniere, Andrea Bunt, and Michael Terry. 2014. Task-centric interfaces for feature-rich software. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures the Future of Design - OzCHI '14*. ACM Press, New York, New York, USA, 49–58. DOI:http://dx.doi.org/10.1145/2686612.2686620

[124] Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. 2013. Community enhanced tutorials: Improving tutorials with multiple demonstrations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, New York, New York, USA, 1779. DOI:http://dx.doi.org/10.1145/2470654.2466235

[125] Ben Lafreniere, Tovi Grossman, Justin Matejka, and George Fitzmaurice. 2014. Investigating the feasibility of extracting tool demonstrations from in-situ video content. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 4007–4016. DOI:http://dx.doi.org/10.1145/2556288.2557142

[126] Gierad P. Laput, Eytan Adar, Mira Dontcheva, and Wilmot Li. 2012. Tutorial-based interfaces for cloud-enabled applications. In *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*. ACM Press, New York, New York, USA, 113. DOI:http://dx.doi.org/10.1145/2380116.2380132

[127] Gierad P. Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. 2013. PixelTone: a multimodal interface for image editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, New York, New York, USA, 2185. DOI:http://dx.doi.org/10.1145/2470654.2481301

[128] Jill H. Larkin and Herbert A. Simon. 1987. Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science* 11, 1 (jan 1987), 65–100. DOI:http://dx.doi.org/10.1111/j.1551-6708.1987.tb00863.x

[129] Joseph J. LaViola Jr., Sarah Buchanan, and Corey Pittman. 2014. Multimodal Input for Perceptual User Interfaces. In *Interactive Displays*. John Wiley & Sons, Ltd, Chichester, UK, 285–312. DOI:http://dx.doi.org/10.1002/9781118706237.ch9

[130] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with interactive example galleries. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. ACM Press, New York, New York, USA, 2257–2266. DOI:http://dx.doi.org/10.1145/1753326.1753667

[131] Pascal Lessel, Alexander Vielhauer, and Antonio Krüger. 2017. Expanding Video Game Live-Streams with Enhanced Communication Channels: A Case Study. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 1571–1576. DOI:http://dx.doi.org/10.1145/3025453.3025708

[132] Rock Leung, Leah Findlater, Joanna McGrenere, Peter Graf, and Justine Yang. 2010. Multi-Layered Interfaces to Improve Older Adults' Initial Learnability of Mobile Applications. *ACM Transactions on Accessible Computing* 3, 1 (sep 2010), 1–30. DOI:http://dx.doi.org/10.1145/1838562.1838563

[133] Sheena Lewis, Mira Dontcheva, and Elizabeth Gerber. 2011. Affective computational priming and creativity. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. ACM Press, New York, New York, USA, 735. DOI:http://dx.doi.org/10.1145/1978942.1979048

[134] Lan Li, Xiongyi Liu, and Allen L. Steckelberg. 2010. Assessor or assessee: How student learning improves by giving and receiving peer feedback. *British Journal of Educational Technology* 41, 3 (may 2010), 525–536. DOI:http://dx.doi.org/10.1111/j.1467-8535.2009.00968.x

[135] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 6038–6049. DOI:http://dx.doi.org/10.1145/3025453.3025483

[136] Wei Li, Justin Matejka, Tovi Grossman, Joseph A. Konstan, and George Fitzmaurice. 2011. Design and evaluation of a command recommendation system for software applications. *ACM Transactions on Computer-Human Interaction* 18, 2 (jun 2011), 1–35. DOI:http://dx.doi.org/10.1145/1970378.1970380

[137] Yang Li and James A. Landay. 2008. Activity-based prototyping of ubicomp applications for long-lived, everyday human activities. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*. ACM Press, New York, New York, USA, 1303. DOI:http://dx.doi.org/10.1145/1357054.1357259

[138] Ching Liu, Juho Kim, and Hao-Chuan Wang. 2018. ConceptScape: Collaborative Concept Mapping for Video Learning. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1–12. DOI:http://dx.doi.org/10.1145/3173574.3173961

[139] Zipeng Liu, Zhicheng Liu, and Tamara Munzner. 2020. Data-driven Multi-level Segmentation of Image Editing Logs. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems - CHI '20 (to appear)*. ACM, New York, New York, USA.

[140] Claudia Lo. 2018. *When All You Have is a Banhammer: The Social and Communicative Work of Volunteer Moderators*. Ph.D. Dissertation. Massachusetts Institute of Technology. https://cmsw.mit.edu/banhammer-social-communicative-work-volunteer-moderators/

[141] Angélica López, Maricela Correa-Chávez, Barbara Rogoff, and Kris Gutiérrez. 2010. Attention to instruction directed to another by U.S. Mexican-heritage children of varying cultural backgrounds. *Developmental Psychology* 46, 3 (2010), 593–601. DOI:http://dx.doi.org/10.1037/a0018157

[142] Zhicong Lu, Michelle Annett, Mingming Fan, and Daniel Wigdor. 2019. "I feel it is my responsibility to stream": Streaming and Engaging with Intangible Cultural Heritage through Livestreaming. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, New York, New York, USA, 1–14. DOI:http://dx.doi.org/10.1145/3290605.3300459

[143] Zhicong Lu, Seongkook Heo, and Daniel J. Wigdor. 2018a. StreamWiki: Enabling Viewers of Knowledge Sharing Live Streams to Collaboratively Generate Archival Documentation for Effective In-Stream and Post Hoc Learning. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (nov 2018), 1–26. DOI:http://dx.doi.org/10.1145/3274381

[144] Zhicong Lu, Haijun Xia, Seongkook Heo, and Daniel Wigdor. 2018b. You Watch, You Give, and You Engage: A Study of Live Streaming Practices in China. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1–13. DOI:http://dx.doi.org/10.1145/3173574.3174040

[145] Paul Lukowicz, Jamie A Ward, Holger Junker, Mathias Stäger, Gerhard Tröster, Amin Atrash, and Thad Starner. 2004. Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers. Springer, Berlin, Heidelberg, 18–32. DOI:http://dx.doi.org/10.1007/978-3-540-24646-6_2

[146] Kurt Luther, Jari-Lee Tolentino, Wei Wu, Amy Pavel, Brian P. Bailey, Maneesh Agrawala, Björn Hartmann, and Steven P. Dow. 2015. Structuring, Aggregating, and Evaluating Crowdsourced Design Critique. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*. ACM Press, New York, New York, USA, 473–485. DOI:http://dx.doi.org/10.1145/2675133.2675283

[147] Ramesh Manuvinakurike, Trung Bui, Walter Chang, and Kallirroi Georgila. 2018. Conversational Image Editing: Incremental Intent Identification in a New Dialogue Task. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. Association for Computational Linguistics, Stroudsburg, PA, USA, 284–295. DOI:http://dx.doi.org/10.18653/v1/W18-5033

[148] Ramesh Manuvirakurike, Jacqueline Brixey, Trung Bui, Walter Chang, Ron Artstein, and Kallirroi Georgila. 2018. DialEdit: Annotations for Spoken Conversational Image Editing. In *Proceedings 14th Joint ACL - ISO Workshop on Interoperable Semantic Annotation*. Association for Computational Linguistics, Santa Fe, New Mexico, USA, 1–9. https://www.aclweb.org/anthology/W18-4701

[149] Richard L. Marsh, Joshua D. Landau, and Jason L. Hicks. 1996. How examples may (and may not) constrain creativity. *Memory & Cognition* 24, 5 (sep 1996), 669–680.

[150] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2011a. Ambient help. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*. ACM Press, New York, New York, USA, 2751. DOI:http://dx.doi.org/10.1145/1978942.1979349

[151] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2011b. IP-QAT: in-product questions, answers, & tips. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, New York, New York, USA, 175. DOI:http://dx.doi.org/10.1145/2047196.2047218

[152] Justin Matejka, Tovi Grossman, and George Fitzmaurice. 2013. Patina: dynamic heatmaps for visualizing application usage. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, New York, New York, USA, 3227. DOI:http://dx.doi.org/10.1145/2470654.2466442

[153] Justin Matejka, Wei Li, Tovi Grossman, and George Fitzmaurice. 2009. CommunityCommands: Command Recommendations for Software Applications. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology - UIST '09*. ACM Press, New York, New York, USA, 193. DOI:http://dx.doi.org/10.1145/1622176.1622214

[154] Lucas Maystre and Matthias Grossglauser. 2015. Fast and accurate inference of Plackett-Luce models. (2015). https://dl.acm.org/doi/10.5555/2969239.2969259

[155] Joanna McGrenere, Ronald M. Baecker, and Kellogg S. Booth. 2002. An evaluation of a multiple interface design solution for bloated software. In *Proceedings of the SIGCHI conference on Human factors in computing systems Changing our world, changing ourselves - CHI '02*. ACM Press, New York, New York, USA, 164. DOI:http://dx.doi.org/10.1145/503376.503406

[156] Joanna McGrenere and Gale Moore. 2000. Are We All in the Same "Bloat"?. In *Proceedings of the Graphics Interface 2000 Conference, May 15-17*. Montréal, Québec, Canada, 187–196. DOI:http://dx.doi.org/10.20380/GI2000.25

[157] Rishabh Mehrotra, Ahmed Hassan Awadallah, Ahmed El Kholy, and Imed Zitouni. 2016. Hey Cortana! Exploring the use cases of a Desktop based Digital Assistant. In *Proceedings of ACM, Tokyo, Japan, August 2017 (CAIR'17)*. 5.

[158] Matthew K. Miller, John C. Tang, Gina Venolia, Gerard Wilkinson, and Kori Inkpen. 2017. Conversational Chat Circles: Being All Here Without Having to Hear It All. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 2394–2404. DOI:http://dx.doi.org/10.1145/3025453.3025621

[159] Naomi Miyake and Donald A. Norman. 1979. To ask a question, one must know enough to know what is not known. *Journal of Verbal Learning and Verbal Behavior* 18, 3 (jun 1979), 357–364. DOI:http://dx.doi.org/10.1016/S0022-5371(79)90200-7

[160] Bill Moorier. 2015. Introducing Twitch Creative. (2015). https://blog.twitch.tv/introducing-twitch-creative-fbfe23b4a114

[161] Felix Müller-Wienbergen, Oliver Müller, Stefan Seidel, and Jörg Becker. 2011. Leaving the Beaten Tracks in Creative Work - A Design Theory for Systems that Support Convergent and Divergent Thinking. *Journal of the Association for Information Systems* 12, 11 (2011). http://aisel.aisnet.org/jais/vol12/iss11/2

[162] Chelsea Myers, Anushay Furqan, Jessica Nebolsky, Karina Caro, and Jichen Zhu. 2018. Patterns for How Users Overcome Obstacles in Voice User Interfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1–7. DOI:http://dx.doi.org/10.1145/3173574.3173580

[163] Amir Shareghi Najar, Antonija Mitrovic, and Bruce M. McLaren. 2014. Adaptive Support versus Alternating Worked Examples and Tutored Problems: Which Leads to Better Learning? In *Lecture Notes in Computer Science 8538*. Springer, Cham, 171–182. DOI:http://dx.doi.org/10.1007/978-3-319-08786-3_15

[164] Susanne Narciss and Katja Huth. 2006. Fostering achievement and motivation with bug-related tutoring feedback in a computer-based training for written subtraction. *Learning and Instruction* 16, 4 (aug 2006), 310–322. DOI:http://dx.doi.org/10.1016/J.LEARNINSTRUC.2006.07.003

[165] Cuong Nguyen and Feng Liu. 2015. Making Software Tutorial Video Responsive. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. ACM Press, New York, New York, USA, 1565–1568. DOI:http://dx.doi.org/10.1145/2702123.2702209

[166] Huy Nguyen, Wenting Xiong, and Diane Litman. 2016. Instant Feedback for Increasing the Presence of Solutions in Peer Reviews. *HLT-NAACL Demos* (2016), 6–10.

[167] David J. Nicol and Debra Macfarlane-Dick. 2006. Formative assessment and self-regulated learning: a model and seven principles of good feedback practice. *Studies in Higher Education* 31, 2 (apr 2006), 199–218. DOI:http://dx.doi.org/10.1080/03075070600572090

[168] Donald A. Norman. 2005. Human-centered design considered harmful. *interactions* 12, 4 (jul 2005), 14. DOI:http://dx.doi.org/10.1145/1070960.1070976

[169] Donald A. Norman. 2007. The next UI breakthrough: command lines. *interactions* 14, 3 (may 2007), 44. DOI:http://dx.doi.org/10.1145/1242421.1242449

[170] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems - CHI '15*. ACM Press, New York, New York, USA, 1221–1224. DOI:http://dx.doi.org/10.1145/2702123.2702149

[171] Sharon Oviatt. 1999. Ten myths of multimodal interaction. *Commun. ACM* 42, 11 (nov 1999), 74–81. DOI:http://dx.doi.org/10.1145/319382.319398

[172] Sharon Oviatt and Philip R. Cohen. 2015. The Paradigm Shift to Multimodality in Contemporary Computer Interfaces. *Synthesis Lectures on Human-Centered Informatics* 8, 3 (apr 2015), 1–243. DOI:http://dx.doi.org/10.2200/S00636ED1V01Y201503HCI030

[173] Tim Paek, Bo Thiesson, Yun-Cheng Ju, and Bongshin Lee. 2008. Search Vox: leveraging multimodal refinement and partial knowledge for mobile voice search. In *Proceedings of the 21st annual ACM symposium on User interface software and technology - UIST '08*. ACM Press, New York, New York, USA, 141. DOI:http://dx.doi.org/10.1145/1449715.1449738

[174] Rui Pan, Lyn Bartram, and Carman Neustaedter. 2016. TwitchViz: A Visualization Tool for Twitch Chatrooms. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '16*. ACM Press, New York, New York, USA, 1959–1965. DOI:http://dx.doi.org/10.1145/2851581.2892427

[175] Randy Pausch and James H. Leatherby. 1991. An Empirical Study: Adding Voice Input to a Graphical Editor. *Journal of the American Voice Input/Output Society* 9 (1991), 2–55.

[176] Amy Pavel, Floraine Berthouzoz, Björn Hartmann, and Maneesh Agrawala. 2013. *Browsing and Analyzing the Command-Level Structure of Large Collections of Image Manipulation Tutorials*. Technical Report. http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-167.html

[177] Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. SceneSkim: Searching and Browsing Movies Using Synchronized Captions, Scripts and Plot Summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15*. ACM Press, New York, New York, USA, 181–190. DOI:http://dx.doi.org/10.1145/2807442.2807502

[178] Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. 2014. 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*. ACM Press, New York, New York, USA, 573–582. DOI:http://dx.doi.org/10.1145/2642918.2647400

[179] Tim F. Paymans, Jasper Lindenberg, and Mark Neerincx. 2004. Usability Trade-offs for Adaptive User Interfaces: Ease of Use and Learnability. In *Proceedings of the 9th international conference on Intelligent user interface - IUI '04*. ACM Press, New York, New York, USA, 301. DOI:http://dx.doi.org/10.1145/964442.964512

[180] Michael J. Pazzani and Daniel Billsus. 2007. *The Adaptive Web: Methods and Strategies of Web Personalization*. Lecture Notes in Computer Science, Vol. 4321. Springer Berlin Heidelberg, Berlin, Heidelberg. 325–341 pages. DOI:http://dx.doi.org/10.1007/978-3-540-72079-9

[181] Anthony J. Pellicone and June Ahn. 2017. The Game of Performing Play: Understanding Streaming as Cultural Production. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 4863–4874. DOI:http://dx.doi.org/10.1145/3025453.3025854

[182] Sundar Pichai. 2016. Google I/O 2016 Keynote. (2016). https://events.google.com/io2016/

[183] Peter Pirolli. 2003. Chapter 7: Exploring and Finding Information. In *HCI Models, Theories, and Frameworks*, John M. Carroll (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, Chapter 7, 157–191.

[184] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F. Cohen. 2011. Pause-and-Play: Automatically Linking Screencast Video Tutorials with Applications. In *Proceedings of the 24th annual ACM symposium on User interface software and technology - UIST '11*. ACM Press, New York, New York, USA, 135. DOI:http://dx.doi.org/10.1145/2047196.2047213

[185] Michael Prince. 2004. Does Active Learning Work? A Review of the Research. *Journal of Engineering Education* 93, 3 (jul 2004), 223–231. DOI:http://dx.doi.org/10.1002/j.2168-9830.2004.tb00809.x

[186] Daniele Procida. 2017. What nobody tells you about documentation. (2017). https://www.divio.com/blog/documentation/

[187] Arkalgud Ramaprasad. 1983. On the definition of feedback. *Behavioral Science* 28, 1 (jan 1983), 4–13. DOI:http://dx.doi.org/10.1002/bs.3830280103

[188] Leah M. Reeves, Jean-Claude Martin, Michael McTear, TV Raman, Kay M. Stanney, Hui Su, Qian Ying Wang, Jennifer Lai, James A. Larson, Sharon Oviatt, T. S. Balaji, Stéphanie Buisine, Penny Collings, Phil Cohen, and Ben Kraal. 2004. Guidelines for multimodal user interface design. *Commun. ACM* 47, 1 (jan 2004), 57. DOI:http://dx.doi.org/10.1145/962081.962106

[189] Brian J. Reiser. 2004. Scaffolding Complex Learning: The Mechanisms of Structuring and Problematizing Student Work. *Journal of the Learning Sciences* 13, 3 (jul 2004), 273–304. DOI:http://dx.doi.org/10.1207/s15327809jls1303_2

[190] Bradley J. Rhodes and Thad Starner. 1996. Remembrance Agent: A continuously running automated information retrieval system. In *The Proceedings of The First International Conference on The Practical Application Of Intelligent Agents and Multi Agent Technology (PAAM '96)*. 487–495. http://alumni.media.mit.edu/%7Erhodes/Papers/remembrance.html

[191] John Rieman. 1996. A field study of exploratory learning strategies. *ACM Transactions on Computer-Human Interaction* 3, 3 (sep 1996), 189–218. DOI:http://dx.doi.org/10.1145/234526.234527

[192] Tom Robertson. 2018. Introducing Tags and New Categories: New Ways to Discover Streamers on Twitch. (2018). https://blog.twitch.tv/introducing-tags-and-new-categories-33744ef7b04f

[193] Rod D. Roscoe, Laura K. Allen, Jennifer L. Weston, Scott A. Crossley, and Danielle S. McNamara. 2014. The Writing Pal Intelligent Tutoring System: Usability Testing and Development. *Computers and Composition* 34 (2014), 39–59.

[194] David H. Rose and Anne Meyer. 2002. *Teaching every student in the Digital Age: Universal design for learning*. Association for Supervision and Curriculum Development. 216 pages.

[195] Daniel M. Russell. 2011. Making the Most of Online Searches. *APS Observer* 24, 4 (apr 2011). https://www.psychologicalscience.org/observer/making-the-most-of-online-searches

[196] D. Royce Sadler. 1989. Formative assessment and the design of instructional systems. *Instructional Science* 18, 2 (jun 1989), 119–144.

[197] M.W. Salisbury, J.H. Hendrickson, T.L. Lammers, C. Fu, and S.A. Moody. 1990. Talk and draw: bundling speech and graphics. *Computer* 23, 8 (aug 1990), 59–65. DOI:http://dx.doi.org/10.1109/2.56872

[198] Donald A. Schoen. 1983. *The reflective practitioner: How professionals think in action*. Basic Books. 374 pages.

[199] Donald A. Schoen. 1985. *The design studio: An exploration of its traditions and potentials*. RIBA Publications for RIBA Building Industry Trust. 99 pages.

[200] Eldon Schoop, Michelle Nguyen, Daniel Lim, Valkyrie Savage, Sean Follmer, and Björn Hartmann. 2016. 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*. ACM Press, New York, New York, USA, 1607–1614. DOI:http://dx.doi.org/10.1145/2851581.2892429

[201] Jana Sedivy and Hilary Johnson. 1999. Supporting creative work tasks: The potential of multimodal tools to support sketching. In *Proceedings of the third conference on Creativity & cognition - C&C '99*. ACM Press, New York, New York, USA, 42–49. DOI:http://dx.doi.org/10.1145/317561.317571

[202] Joseph Seering, Robert Kraut, and Laura Dabbish. 2017. Shaping Pro and Anti-Social Behavior on Twitch Through Moderation and Example-Setting. In *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing - CSCW '17*.

ACM Press, New York, New York, USA, 111–125. DOI:http://dx.doi.org/10.1145/2998181.2998277

[203] Joseph Seering, Tony Wang, Jina Yoon, and Geoff Kaufman. 2019. Moderator engagement and community development in the age of algorithms. *New Media & Society* (jan 2019), 146144481882131. DOI:http://dx.doi.org/10.1177/1461444818821316

[204] Vidya Setlur, Sarah E. Battersby, Melanie Tory, Rich Gossweiler, and Angel X. Chang. 2016. Eviza: A Natural Language Interface for Visual Analysis. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. ACM Press, New York, New York, USA, 365–377. DOI:http://dx.doi.org/10.1145/2984511.2984588

[205] Anirudh Sharma, Sriganesh Madhvanath, Ankit Shekhawat, and Mark Billinghurst. 2011. MozArt: a multimodal interface for conceptual 3D modeling. In *Proceedings of the 13th international conference on multimodal interfaces - ICMI '11*. ACM Press, New York, New York, USA, 307–310. DOI:http://dx.doi.org/10.1145/2070481.2070538

[206] Ben Shneiderman. 2000a. The limits of speech recognition. *Commun. ACM* 43, 9 (sep 2000), 63–65. DOI:http://dx.doi.org/10.1145/348941.348990

[207] Ben Shneiderman. 2000b. Universal usability. *Commun. ACM* 43, 5 (may 2000), 84–91. DOI:http://dx.doi.org/10.1145/332833.332843

[208] Ben Shneiderman. 2002a. Creativity support tools. *Commun. ACM* 45, 10 (oct 2002), 116–120. DOI:http://dx.doi.org/10.1145/570907.570945

[209] Ben Shneiderman. 2002b. Promoting universal usability with multi-layer interface design. *ACM SIGCAPH Computers and the Physically Handicapped* 73-74 (jun 2002), 1–8. DOI:http://dx.doi.org/10.1145/960201.957206

[210] Ben Shneiderman. 2007. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM* 50, 12 (dec 2007), 20–32. DOI:http://dx.doi.org/10.1145/1323688.1323689

[211] Milad Shokouhi, Rosie Jones, Umut Ozertem, Karthik Raghunathan, and Fernando Diaz. 2014. Mobile query reformulations. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval - SIGIR '14*. ACM Press, New York, New York, USA, 1011–1014. DOI:http://dx.doi.org/10.1145/2600428.2609497

[212] Pao Siangliulue, Kenneth C. Arnold, Krzysztof Z. Gajos, and Steven P. Dow. 2015a. Toward Collaborative Ideation at Scale: Leveraging Ideas from Others to Generate More Creative and Diverse Ideas. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*. ACM Press, New York, New York, USA, 937–945. DOI:http://dx.doi.org/10.1145/2675133.2675239

[213] Pao Siangliulue, Joel Chan, Krzysztof Z. Gajos, and Steven P. Dow. 2015b. Providing Timely Examples Improves the Quantity and Quality of Generated Ideas. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition - C&C '15*. ACM Press, New York, New York, USA, 83–92. DOI:http://dx.doi.org/10.1145/2757226.2757230

[214] Arjun Singh, Sergey Karayev, Kevin Gutowski, and Pieter Abbeel. 2017. Gradescope: A Fast, Flexible, and Fair System for Scalable Assessment of Handwritten Work. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale - L@S '17*. ACM Press, New York, New York, USA, 81–88. DOI:http://dx.doi.org/10.1145/3051457.3051466

[215] Max Sjöblom and Juho Hamari. 2017. Why do people watch others play video games? An empirical study on the motivations of Twitch users. *Computers in Human Behavior* 75 (oct 2017), 985–996. DOI:http://dx.doi.org/10.1016/J.CHB.2016.10.019

[216] Max Sjöblom, Maria Törhönen, Juho Hamari, and Joseph Macey. 2017. Content structure is king: An empirical study on gratifications, game genres and content type on Twitch. *Computers in Human Behavior* 73 (aug 2017), 161–171. DOI:http://dx.doi.org/10.1016/J.CHB.2017.03.036

[217] Nancy Sommers. 1982. Responding to Student Writing. *College Composition and Communication* 33, 2 (may 1982), 148.

[218] Kay Stanney, Shatha Samman, Leah Reeves, Kelly Hale, Wendi Buff, Clint Bowers, Brian Goldiez, Denise Nicholson, and Stephanie Lackey. 2004. A Paradigm Shift in Interactive Computing: Deriving Multimodal Design Principles from Behavioral and Neurological Foundations. *International Journal of Human-Computer Interaction* 17, 2 (jun 2004), 229–257. DOI:http://dx.doi.org/10.1207/s15327590ijhc1702_7

[219] Wolfgang Stuerzlinger, Olivier Chapuis, Dusty Phillips, and Nicolas Roussel. 2006. User interface facades: towards fully adaptable user interfaces. In *Proceedings of the 19th annual ACM symposium on User interface software and technology - UIST '06*. ACM Press, New York, New York, USA, 309. DOI:http://dx.doi.org/10.1145/1166253.1166301

[220] Danny Sullivan. 2018. A reintroduction to Google's featured snippets. (2018). https://www.blog.google/products/search/reintroduction-googles-featured-snippets/

[221] Tamara Sumner and Markus Stolze. 1997. Evolution, not revolution: participatory design in the toolbelt era. In *Computers and design in context*, Morten Kyng and Lars Mathiassen (Eds.). MIT Press, Chapter 1, 1–26. https://dl.acm.org/doi/10.5555/270318.270319

[222] John C. Tang, Gina Venolia, and Kori M. Inkpen. 2016. Meerkat and Periscope: I Stream, You Stream, Apps Stream for Live Streams. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI '16*. ACM Press, New York, New York, USA, 4770–4780. DOI:http://dx.doi.org/10.1145/2858036.2858374

[223] Michael Terry and Elizabeth D. Mynatt. 2002. Side Views: Persistent, On-Demand Previews for Open-Ended Tasks. In *Proceedings of the 15th annual ACM symposium on User interface software and technology - UIST '02*. ACM Press, New York, New York, USA, 71. DOI:http://dx.doi.org/10.1145/571985.571996

[224] Maryam Tohidi, William Buxton, Ronald Baecker, and Abigail Sellen. 2006. Getting the right design and the design right. In *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06*. ACM Press, New York, New York, USA, 1243. DOI:http://dx.doi.org/10.1145/1124772.1124960

[225] K. Topping. 1998. Peer Assessment Between Students in Colleges and Universities. *Review of Educational Research* 68, 3 (jan 1998), 249–276.

[226] Cristen Torrey, David W. McDonald, Bill N. Schilit, and Sara Bly. 2007. How-To pages: Informal systems of expertise sharing. In *ECSCW 2007*. Springer London, London, 391–410. DOI:http://dx.doi.org/10.1007/978-1-84800-031-5_21

[227] Ba Tu Truong and Svetha Venkatesh. 2007. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 3, 1 (2007), 3. DOI:http://dx.doi.org/10.1145/1198302.1198305

[228] Sheng-Chau Tseng and Chin-Chung Tsai. 2007. On-line peer assessment and the role of the peer feedback: A study of high school computer course. *Computers & Education* 49, 4 (2007), 1161–1174.

[229] Juhani E Tuovinen and John Sweller. 1999. *A Comparison of Cognitive Load Associated With Discovery Learning and Worked Examples*. Technical Report 2. 334–341 pages. http://idtoolbox.eseryel.com/uploads/9/0/7/5/9075695/1999-03660-014.pdf

[230] Sara Värlander. 2008. The role of students' emotions in formal feedback situations. *Teaching in Higher Education* 13, 2 (apr 2008), 145–156. DOI:http://dx.doi.org/10.1080/13562510801923195

[231] Laton Vermette, Parmit Chilana, Michael Terry, Adam Fourney, Ben Lafreniere, and Travis Kerr. 2015. CheatSheet: A Contextual Interactive Memory Aid for Web Applications. In *Proceedings of the 41st Graphics Interface Conference (GI '15)*. Canadian Information Processing Society, CAN, 241–248.

[232] Laton Vermette, Shruti Dembla, April Y. Wang, Joanna McGrenere, and Parmit K. Chilana. 2017. Social CheatSheet: An Interactive Community-Curated Information Overlay for Web Applications. *Proceedings of the ACM on Human-Computer Interaction* 1, CSCW (dec 2017), 1–19. DOI:http://dx.doi.org/10.1145/3134737

[233] Xu Wang, Benjamin Lafreniere, and Tovi Grossman. 2018. Leveraging Community-Generated Videos and Command Logs to Classify and Recommend Software Workflows. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*.

ACM Press, New York, New York, USA, 1–13. DOI:http://dx.doi.org/10.1145/3173574.3173859

[234] Sarah Weir, Juho Kim, Krzysztof Z. Gajos, and Robert C. Miller. 2015. Learnersourcing Subgoal Labels for How-to Videos. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing - CSCW '15*. ACM Press, New York, New York, USA, 405–416. DOI:http://dx.doi.org/10.1145/2675133.2675219

[235] Donghee Yvette Wohn, Guo Freeman, and Caitlin McLaughlin. 2018. Explaining Viewers' Emotional, Instrumental, and Financial Support Provision for Live Streamers. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. ACM Press, New York, New York, USA, 1–13. DOI:http://dx.doi.org/10.1145/3173574.3174048

[236] Wenting Xiong, Diane Litman, and Christian Schunn. 2012. Natural Language Processing techniques for researching and improving peer feedback. *Journal of Writing Research* 4, 2 (nov 2012), 155–176. DOI:http://dx.doi.org/10.17239/jowr-2012.04.02.3

[237] Miao Yang, Richard Badger, and Zhen Yu. 2006. A comparative study of peer and teacher feedback in a Chinese EFL writing class. *Journal of Second Language Writing* 15, 3 (dec 2006), 179–200. DOI:http://dx.doi.org/10.1016/j.jslw.2006.09.004

[238] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. 2003. Faceted metadata for image search and browsing. In *Proceedings of the conference on Human factors in computing systems - CHI '03*. ACM Press, New York, New York, USA, 401. DOI:http://dx.doi.org/10.1145/642611.642681

[239] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. 2009. Sikuli: using GUI screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology - UIST '09*. ACM Press, New York, New York, USA, 183. DOI:http://dx.doi.org/10.1145/1622176.1622213

[240] Alvin Yuan, Kurt Luther, Markus Krause, Sophie Isabel Vennix, Steven P Dow, and Björn Hartmann. 2016. Almost an Expert: The Effects of Rubrics and Expertise on Perceived Value of Crowdsourced Design Critiques. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing - CSCW '16*. ACM Press, New York, New York, USA, 1003–1015. DOI:http://dx.doi.org/10.1145/2818048.2819953

[241] Donghee Yvette Wohn. 2019. Volunteer Moderators in Twitch Micro Communities: How They Get Involved, the Roles They Play, and the Emotional Labor They Experience. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19*. ACM Press, New York. DOI:http://dx.doi.org/10.1145/3290605.3300390

[242] Yinglong Zhang and Robert Capra. 2019. Understanding How People use Search to Support their Everyday Creative Tasks. In *Proceedings of the 2019 Conference on Human*

Information Interaction and Retrieval - CHIIR '19. ACM Press, New York, New York, USA, 153–162. DOI:http://dx.doi.org/10.1145/3295750.3298936

[243] Yu Zhong, T. V. Raman, Casey Burkhardt, Fadi Biadsy, and Jeffrey P. Bigham. 2014. JustSpeak: enabling universal voice control on Android. In *Proceedings of the 11th Web for All Conference on - W4A '14*. ACM Press, New York, New York, USA, 1–4. DOI:http://dx.doi.org/10.1145/2596695.2596720