

UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Enhancing Visual Media Through Reflection and Recomposition

Permalink

<https://escholarship.org/uc/item/0sr8z328>

Author

Warner, Jeremy

Publication Date

2023

Peer reviewed|Thesis/dissertation

Enhancing Visual Media Through Reflection and Recomposition

by

Jeremy Benjamin Warner

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Björn Hartmann, Chair

Professor Alexei (Alyosha) Efros

Professor Marti Hearst

Doctor Valentina Shin

Summer 2023

Enhancing Visual Media Through Reflection and Recomposition

Copyright 2023

by

Jeremy Benjamin Warner

Abstract

Enhancing Visual Media Through Reflection and Recomposition

by

Jeremy Benjamin Warner

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Björn Hartmann, Chair

Visual media like vector graphics and presentations are essential to human communication. Artists, designers, and presenters use an iterative reflection and recombination process to progressively refine their work. Here, *reflection* describes gathering and absorbing feedback or information after some visual media is created, while *recomposition* describes adapting and transforming some existing visual media. The key to supporting these stages is providing users with rich, flexible data representations that adhere to their domain and task. Additionally, user interfaces for interacting with this data must balance raw transformative power and the author's creative control. This dissertation supports these media reflection and recombination processes, and presents techniques for leveraging automatic styling and feedback systems in visual media creation processes.

The goal is to broaden the focus on generative creation tools to the more extensive iterative cycle designers employ (i.e., reflection and recombination). The first aspect focuses on collecting and refining presentation feedback with SlideSpecs. SlideSpecs is a system for collecting, aggregating, and understanding presentation feedback to assist author refinements. Next, this dissertation will present VST and VLT, a pair of automation-powered design tools for transferring and mixing vector graphics styles and layouts. These tools enable mixing higher-level design properties and rules across vector graphics by generating and leveraging cross-design correspondences. These approaches acknowledge the richness of authors' flexibility and control when shaping their media. With this framing, I demonstrate the benefits and requirements for extending this richness throughout the iterative design cycle.

To Paula and Hollis

Contents

Contents	ii
List of Figures	iv
List of Tables	ix
1 Introduction	1
1.1 Contributions	5
1.2 Overview	6
1.3 Authorship and Prior Publication	6
2 Background	7
2.1 Augmenting Media with Cross-Domain Representations	7
2.2 Finding Correspondences Across Visual Media	10
2.3 UI for Augmentation and Correspondence	15
3 Related Works	17
3.1 Collecting and Organizing Feedback	17
3.2 Contextualizing Verbal Discussions	19
3.3 Inferring Structures in Visual Designs	19
3.4 Generating Design Layouts	21
3.5 Supporting Creative Processes with AI	21
3.6 Vector Graphics Design Tools	23
4 SlideSpecs	24
4.1 Introduction	25
4.2 Formative Study: Existing Practices	27
4.3 Design Implications	32
4.4 Collating Feedback with SlideSpecs	33
4.5 Implementation	36
4.6 Effectiveness Study: Using SlideSpecs	39
4.7 Discussion	44

5	Vector Style Transfer	46
5.1	Introduction	46
5.2	Vector Style Transfer	50
5.3	Implementation	54
5.4	Evaluations	55
5.5	Discussion	64
6	Vector Layout Transfer	69
6.1	Introduction	69
6.2	VLT Walkthrough	73
6.3	Optimizing Layouts	74
6.4	Design Results	75
6.5	Discussion	76
7	Conclusion	79
7.1	Restatement of Contributions	80
7.2	Future Work - Reflection	80
7.3	Future Work - Recomposition	82
7.4	Closing Remarks	83
	Bibliography	85

List of Figures

1.1	Early forms of visual media (L: cave paintings, R: stone etchings).	1
1.2	Visual media distribution (L: printing press, R: plasma display).	2
1.3	An overview of the iterative creative process for visual media.	3
1.4	Relationships between images, vector graphics, and presentations.	4
2.1	Data connected across several different domains in SlideSpecs.	9
2.2	Rules that can be inferred within a design. These displayed rules (and more) are explored in detail by Shin et al.'s vector graphics graph walk kernel project [136].	11
4.1	SlideSpecs supports presenters in three phases: Presentation, Discussion, and Review. During the Presentation, SlideSpecs collates audience text critiques referencing talk slides, feedback tags, or other critiques. During the Discussion, the interface records and transcribes the discussion audio. Participants can easily reference previous critiques, and a facilitator annotates discussion topics. During Review, SlideSpecs provides presenters with rich contextual data that links the feedback gathered from both previous phases.	25
4.2	An overview of the SlideSpecs <i>feedback-providing</i> interface. In the <i>Slides Pane</i> , a detailed slide image view is shown (A), and hovering over the smaller slide thumbnails (B) temporarily updates the large image, current slide (A), allowing users to scan over talk slides. Audience members can select these thumbnails to attach their comments to designated presentation slides (C). In the <i>Comments Pane</i> , the interface features different ways of sorting the comments (D), a list of tags contained by comments, and a text area entering feedback. Last, the interface displays a live-updating list of the audience's comments (E), which includes comment metadata (i.e., comment author, creation time, and slides referenced).	28
4.3	SlideSpecs supports three phases (presentation, discussion, and review) and three distinct participant roles (presenter, audience, and facilitator). The presenter delivers their practice presentation while audience members and the facilitator share comments using the <i>Presentation Interface</i> (A). During the discussion (B), the presenter and audience discuss feedback while the facilitator annotates the discussion. The presenter later reviews all feedback at their leisure in the <i>Review Interface</i> (C).	29

- 4.4 The *Discussion Participant* interface. (A) The slide pane detail view. (B) The listing of slide thumbnails, which also allows discussion participants to tag and filter comments by a specific slide. (C) Comments currently under discussion. These are synced with the comments shown in Fig. 4.5B. (D) Comments queued for discussion, initially populated by comments that the audience marked for discussion. (E) Comments already discussed. Once the facilitator marks these comments as discussed, they move into this section. 35
- 4.5 The *Discussion Facilitator* interface. (A) A microphone button to toggle audio capture and a visualizer for the audio input to verify that the discussion audio is active. The text entry below the audio controls can submit new discussion comments and works as a search tool to find relevant comments. (B) The “discussing” pane shows comments actively being discussed. These comments sync with the discussion participant view. (C) Comments queued up for discussion. (D) Previously discussed comments. 36
- 4.6 The *Review* interface. (A) The audio waveform from the verbal discussion is shown along with playback and speed controls. Users can scrub and jump directly on the waveform to navigate through the audio file. (B) The slides pane. Clicking on a slide allows filtering the visible comments to only those which refer to the selected slide. (C) The full transcript of the discussion (shown collapsed). (D) The listing of all comments and discussion topics. Users can hover over a referenced comment to highlight the time range in which the comment was discussed in the waveform and view the directly related transcript segment. This connection is learned from the facilitator marking specific comments as being discussed during the discussion phase. 37
- 4.7 During the presentation, audience members write critiques and respond to existing critiques. Optionally, audience members can include context with their text critiques (e.g., specific slides, tags for feedback type). The threaded text comments flagged for discussion appear in the Discussion interface. During the discussion, the facilitator marks which topics are being discussed, matching existing text critiques where possible. SlideSpecs records and transcribes the discussion audio, allowing the presenter to review the collated text and verbal feedback together. 38
- 4.8 Participants used SlideSpecs for a wide variety of practice talks (A-D). They received feedback concerning many aspects of their presentation, from slide design, delivery, and requests for clarification. We highlight select feedback from four different practice presentations. These comments showcase topics that the audience’s feedback addressed; eight comments (A1-D2) are shown in detail alongside a rough characterization of what part of the presentation the comment addresses. 42

4.9	Likert scale responses for 14 audience members (left) and eight presenters (right) in the group deployment. All except one presenter and two audience members agreed that they would use the interface in the future (85%). Likert scale responses reflect that audience members broadly find seeing the real-time of others to be very helpful (10/14) though somewhat distracting (6/14). Audience members noted seeing and agreeing/responding to feedback was helpful. All 8/8 presenters favored slide organization during the feedback review and found the feedback collation useful.	43
5.1	VST generates new output graphics by transferring visual styles from source graphics onto target graphics. The styling interface enables filtering which styles to transfer, filtering which elements to stylize, and previewing all or part of the new stylized output graphics. The output graphics retain a similar structure to the original target while bearing visual styles from the source graphics.	47
5.2	An overview of automated design correspondence. To relate design elements, we first construct a graph from each given design, where the <i>vertices</i> are primitive design elements (e.g., shapes, text, images) and <i>edges</i> are semantic relationships (e.g., same fill, containment, same font). Once the SOURCE and TARGET graphs are constructed, we then compute a correspondence between the two designs' elements using the technique previously detailed in [136]. This automatically generated correspondence is VST's basis for (a) how to find similar elements <i>within</i> a design (e.g., for easier selection/styling) and (b) identifying which elements are similar to each other <i>across</i> designs (e.g., determining which initial styles to transfer). Each TARGET element is linked to a single SOURCE element. Only a subset of links between these designs' elements are shown.	49
5.3	An overview of the VST interface, including (A) the SOURCE graphics (where the style is sourced from), (B) the TARGET graphics, (C) the OUTPUT canvas (the current style transfer result), and (D) customization controls for matching element styles across canvases and filtering which style attributes to copy or modify. Designers can filter this list of attributes (shown in D) based on the current selection to do more focused editing or instead modify shared style attributes across the entire design.	50
5.4	The black lines show an initial correspondence between the elements of the SOURCE and TARGET designs. The green lines show an alternative, more desired set of links. When users select their desired SOURCE and TARGET elements and press <i>Transfer Source Style</i> , VST will update these links, redirecting the flow of visual styles across designs.	52

5.5	The Customization UI shows the SOURCE and TARGET selections and similar TARGET elements. The similarity controls [-/set/+] can adjust the selection to the desired TARGET elements. Once satisfied with the SOURCE-TARGET mapping, pressing <i>Transfer Source Style</i> will transfer all styles from the SOURCE elements to the active TARGET selection. The Customization UI also provides fine-grained control over which styles to transfer. Element style attributes can be copied, reset, or customized for each set of similar values. This list can be filtered only to show styles for the current selection and only to show modified attributes. The UI also features the <i>Copy All</i> and <i>Copy None</i> buttons – <i>Copy All</i> blindly copies all styles for every matched element (e.g., the fully automatic output), and <i>Copy None</i> restores the OUTPUT graphics to the original TARGET state.	54
5.6	Task 1 (Style Transfer) – Basic Graphics Pairs.	57
5.7	Task 2 (Style Transfer) – Open-Ended Transfer.	58
5.8	Design Replication Task – A new set of expert designers (replication designers RD1-4) replicated six reference designs (RT1-6) from the previous style transfer evaluation tasks (Fig. 5.7) using two different starting points: <i>Basic</i> and <i>Auto</i> . The first approach involved using Illustrator to transform the <i>Basic</i> input design to the replication goal. The second approach again used Illustrator, but instead has the algorithmic output (<i>Auto</i>) as the starting point. We provided the RDs with source styles and target structures from the previous study in vector form and a reference image of the replication goal for both approaches.	60
5.9	Summary of Likert survey data from designers D1-6.	63
5.10	Plots of the duration, edits, and selections data from the design replication (RT1-6). Along each recorded measure (duration, edits, and selections), the authors using VST outperformed all four expert designers using Adobe Illustrator in replicating the stylized designs. Duration: seconds per task. Edits: Number of edit operations per task. Selections: Number of selection updates per task. The <i>Basic</i> and <i>Auto</i> plots also include ticks showing the <i>standard error</i> for each task computed over RD1-4. VST was only used once per task to obtain a baseline, so there are no comparable ticks to show.	66
5.11	Additional graphics generated by transferring styles with VST.	68

- 6.1 Our layout transformation pipeline: given two vector graphics designs (A, B), we distill design layout data into grouped semantic layout rules for each design (L_A , L_B). We also compute a correspondence between the elements of the two designs (M_{AB}). Using L_A , L_B , and M_{AB} , we generate T: a transformation of the graphic design elements of A. Applying this transformation T yields design A^* , which we then distill new layout rules from (L_{A^*}). Designers can view the applied transformation and leverage control over which rules are prioritized, yielding new transformation T^* , which in turn yields a new design. This last component is an interactive, iterative process that aims to let designers retain full control of their design’s layout while benefitting from automation. 70
- 6.2 The left side of this figure shows two designs with varying layouts, along with differing layout rules that were inferred for corresponding groups of elements. The boxes and links in these designs represent different rule types that we recognize. The right side shows a representation of the different types of layout relationships we can model between elements. Asymmetric rules (e.g., containment) are represented internally as ordered trees while symmetric rules (e.g., alignment) are represented as simple sets (see also Table 6.2). 71
- 6.3 The VLT interface showing the source layout (e.g., B), the output layout (e.g., A^*), and the layout rule customization panel. This output and the original target (A) can be toggled. The layout rules dynamically update as the output canvas is updated; here they show detected horizontal and right alignment rules. There are also global and element-specific layout transfer buttons, and a per-element property transfer based on that element’s matched element. This also works for multiple selected elements, grouping alike values. 72
- 6.4 An output gallery of layouts made with VLT. Each column shows (in order from columns 1-4): the source or inspiring layout (B), the target input design (A), the fully automatic result of globally applying layout transformation rules to the entire design, and the final output design iteratively made with VLT’s range of semantic editing designer controls. 77

List of Tables

4.1	An overview of talk feedback collated with SlideSpecs. Audience members contributed 86% of text comments during <i>Presentation</i> and 14% of comments during <i>Discussion</i> . Here, N refers to the active group size – all who contributed at least one comment (mean: 9.5, s.d.: 3.8). Time refers to the minutes spent on each phase (mean: 17.0, s.d.: 3.5).	40
5.1	Replication work data – usage statistics averaged over replication tasks RT1-6 (see Fig. 5.8). The <i>Basic</i> and <i>Auto</i> columns show aggregate data collected from the four expert replication designers (RD1-4), while the <i>VST</i> column shows data from the paper authors using VST to replicate designs.	65
6.1	Designer Controls for Layout Editing	74
6.2	Supported Layout Heuristic Rules (e.g., L_A)	75

Acknowledgments

To my advisor, Björn Hartmann. Your thoughtful feedback, guidance, and encouragement were invaluable. I am incredibly grateful for the care and support that you have provided during my time at Berkeley. You've helped refine my ability to define and tackle interesting research questions and how to best communicate that research to a broader audience. Thank you for all that you've provided me; I couldn't ask for a better Ph.D. advisor.

To my thesis committee: Alexei Efros, Marti Hearst, and Valentina Shin. Thank you for the thoughts and advice on how to best shape and package my research into a more coherent vision. This work and its representation were greatly aided by your comments. Thank you also to Aleta Martinez, Jean Nguyen, and Shirley Salanio for their administrative efforts in carving a path through heaps of paperwork and departmental requirements.

To my research internship mentors: Amy Pavel, Gabriel Reyes, Jeff Bigham (Apple), Valentina Shin, Wilmot Li, Holger Winnemoeller (Adobe), and Ben Lafreniere, Tovi Grossman (Autodesk). Thank you for inviting me into your research groups and sharing what research in industry can really be like. This experience working across these larger-scale research labs was enlightening and served to broaden my areas of research interest.

To my undergraduate and master's advisors: Wendi Heinzelman and Philip Guo. Your initial introduction to research as a place to learn more about the world and participate in technical problems sparked a fascination that has shaped my life. Thank you for being kind and supportive research mentors; you continue to inspire me even after leaving your labs.

To other Berkeley faculty who taught me: Paul Wright and Björn Hartmann for designing interactive devices, Eric Paulos for a whimsical journey through critical making, Anca Dragan for understanding how robots and humans interact, Jonathan Ragan-Kelley for compilers and domain-specific languages, Toni Mar for helping to balance mind and body, Kimiko Ryokai for tangible user interfaces, Abigail De Kosnik for learning about the history and theory of media, Jacob Gaboury for reflecting on the politics of code, Jitendra Malik for an introduction to computer vision, Alexei Efros and Angjoo Kanazawa for diving into cameras and computational photography, Ren Ng for 3D graphics and rendering, and to John Canny and Eric Paulos for participating as my preliminary examiners. Thank you for your quality instruction and encouragement in deeply engaging with and understanding new topics.

To my research collaborators: Li Chen, Ilker Demirkol, Pak Lam Yung, Dawei Zhou, Ufuk Muncuk, Kaushik Chowdhury, Stefano Basagni (**all with Wendi Heinzelman**); John Doorenbos, Bradley Miller, Joyce Zhu, Mitchell Gordon, Jeffery White, Renan Zanelatto (**all with Philip Guo**); Will McGrath, Daniel Drew, Majeed Kazemitabaar, Mitchell Karchemsky, David Mellis, Rundong Tian, Sarah Serman, Ethan Chiou, Eric Paulos, Andrew Head, Ben Lafreniere, George Fitzmaurice, Tovi Grossman, Hijung Valentina Shin, Celso Gomes,

Holger Winnemoeller, Wilmot Li, Amy Pavel, Tonya Nguyen, Maneesh Agrawala, Angela Zhang, Frederick Kyu Won Kim, Shuyao Zhou, Susan Lin, Sauhard Jain, Matthew Lee, JD Zamfirescu, Can Liu (**all with Björn Hartmann**). Thank you for the engaging discussions and mountain of shared efforts that drove our projects to success; I'm grateful for your help.

To other Berkeley (and beyond) grad school compatriots: Vitchyr Pong, Jingyi Li, Jasper Tran O'Leary, Elena Duran, Will McGrath, Andrew Head, Yifan Wu, Daniel Lim, Alex Reinking, Eldon Schoop, Sarah Stermann, Molly Nicholas, Corten Singer, Danielle Poreh, Phillipe Laban, Austin Bailey, Johnson Truong, Alexander Youngberg, Sam Simpson, Cameron Louie, Forrest Huang, Nate Weinman, James Smith, TK Bala, Jeff Mahler, Cesar Torres, Katherine Stasaki, Meghan Clark, Kevin Tian, Christine Dierk, Katherine Song, Savvas Petridis, JD Zamfirescu, Boyan Xu, Michael Lee, Chris Berthelet, Nic Brody, Steve Trettel, Ally Nagasawa-Hinck, Johanna Roth, Richard 'Ducky' Lin, Parker Ziegler, Eddie Kim, Eric Rawn, Peitong Duan, Shm Almeda, Tim Aveni, Hila Mor, Matthew Jörke, Vivian Liu, Ananya Nandy, Yakira Mirabito, Elisa Kwon, Nicole Goridkov, Caseysimone Ballestas, Kevin Ma, and so many others who I've had the pleasure of interacting with at Berkeley. Thank you all for so fully enriching my rose-colored campus life. To my housemates (Stefan, Noah, Andy, and Josh): thank you for being there for me through thick and thin; I feel blessed to have friends who created such a stable, warm, and creative environment together.

Finally, I sincerely thank my parents, Paula and Hollis, and my larger immediate family (Naomi, John F., Hannah, Alex, John, Ashley, Felicity, Evangelina, Joylyn, Eliana, Maria, Abigail, and Evelyn). You mean the world to me. Your love, support, and companionship often gave me strength to press on through the rougher patches of graduate school, and are a reminder to cherish life as the fleeting yet wonderful moment that it is.

Thank you.

Chapter 1

Introduction



Figure 1.1: Early forms of visual media (L: cave paintings, R: stone etchings).

Vision and imagery are crucial aspects of communicating information. Humans have a rich history of creating techniques for generating images to convey or express information. Stemming back to the earliest cave paintings¹ and hieroglyphics² to modern printed and electronic visual arts and forms, visual communication long has been a cornerstone of history; of encoding parts of the human experience and findings into a more lasting format.

Coupled with this long history of visual media is a set of creative tools and techniques that enable visual and semantic depictions. These tools ranged from the paint used for painting on those cave walls and the chisel used to etch markings into stone. Modern tools for unlocking different forms of representing and storing images include the printing press, electronic computer displays and more.

¹Cave painting image source: <https://bit.ly/3q94s5y>

²Hieroglyphics image source: <https://bit.ly/305Qn0R>

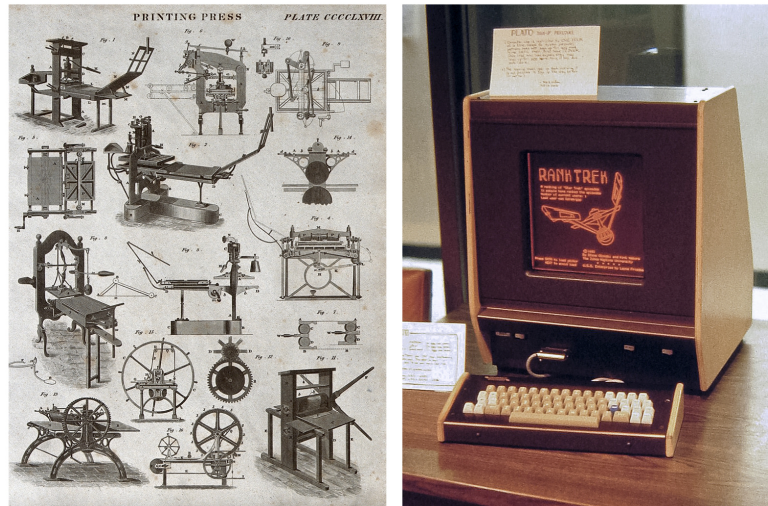


Figure 1.2: Visual media distribution (L: printing press, R: plasma display).

Entire industries have formed around the shared goal of creating, rendering, and distributing visual media. Research contributions from visualization, computer graphics, and user interface design fed into progress for visual media formats spanning print media³, film projectors, computer monitors⁴, and immersive virtual reality headsets. While media format and display technology for distributing images progressed, so did new means of shaping the visual content (e.g., graphic design tools, video editing tools). Ultimately, while these tools have enabled expression with new visual media forms, there are more than new tools to credit for this broad media transformation. There is also a nuanced iterative process that artists, designers, and authors leverage when generating visual media.

While this iterative process does start with *creation*, it also includes *reflection* and *recomposition*. Reflection serves as an opportunity to gather and evaluate feedback to see if the media feels right to the creator or is otherwise effective in its goal. Recomposition serves as means of remixing and restyling elements into a more refined creative spark as the work builds on itself. This iterative process also informs all proceeding future work. All art is at least partially derivative – creation feeds on the efforts and learnings of those before us.

This dissertation will focus on supporting this more extensive process of reflecting on and recomposing visual media, which involves collecting and integrating arranged feedback. This is an iterative refinement process that successful visual media often goes through many times. I will explicitly focus on more semantically meaningful representations of visual media (i.e., vector graphics, slide-based presentations) rather than images (grids of pixels).

³Printing press image source: <https://bit.ly/3DuvhUZ>

⁴Plasma screen image source: <https://bit.ly/3KgUSo5>

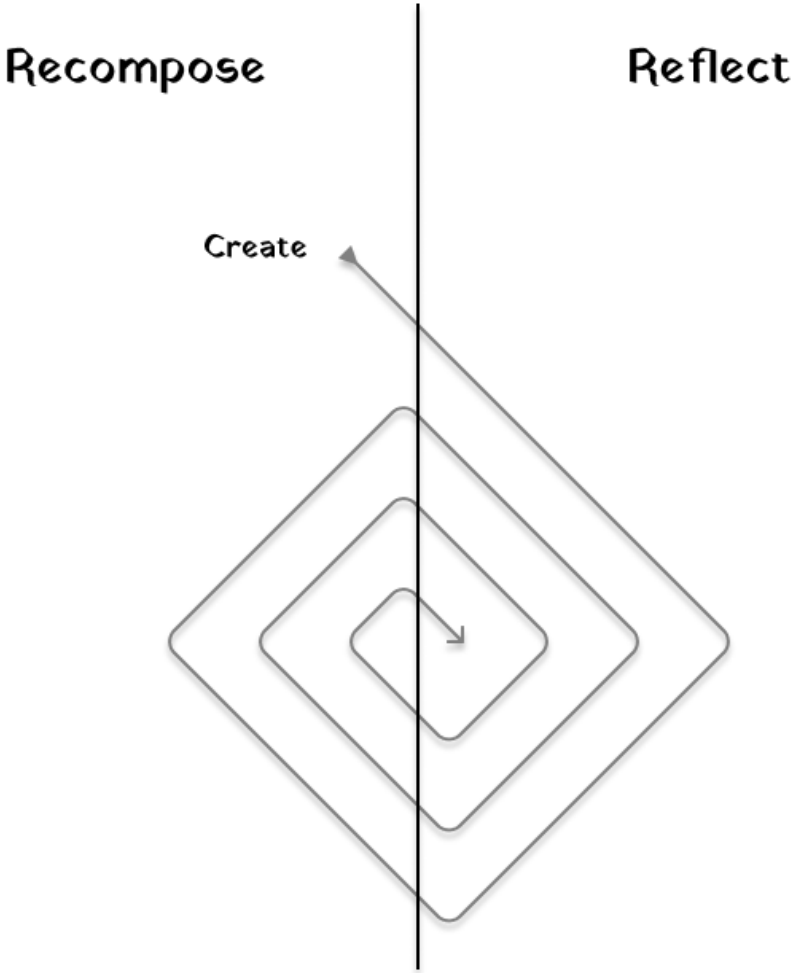


Figure 1.3: An overview of the iterative creative process for visual media.

While images are ubiquitous, these other integrated and richer forms of media are more tailored for more complex media. These formats are used broadly by designers to create different styled graphics, different layouts, and while presentations are common across a range of domains, including education, government, business, and more.

Vector graphic designs offer many benefits, which leads to their massive influence in the design sector. The first attribute is scalability, in that geometric vector graphic elements (e.g., lines, shapes, fonts) are infinitely scalable. This scalability means they can be rendered at any pixel resolution while retaining all shape information rather than becoming blurry. They can also contain embedded rasterized images, combining the best of both worlds.

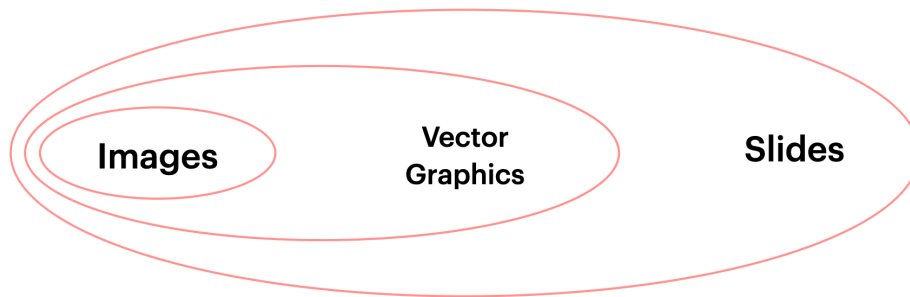


Figure 1.4: Relationships between images, vector graphics, and presentations.

Another attractive attribute is their editability – semantic information can be included in the documents directly rather than inferred afterward from a rendered image. Due to their explicit XML structure⁵, vector graphics can tersely represent complex images and shapes as primitives included in the SVG specification. While the specification for slide-based presentations is not as explicit in comparison, they generally consist of ordered vector graphics and dynamic media. Presentations arrange collections of shapes, text, images, and movies to align with a message or narrative that the author is crafting. Despite these formats' general popularity and flexibility, many ways still exist to further enhance visual media with reflection and recomposition.

⁵<https://www.w3.org/TR/SVG2/>

1.1 Contributions

This dissertation contributes research in two main areas:

- Reflection Support for Presentations (SlideSpecs):
 - SlideSpecs, a novel system for collating audience text and spoken presentation feedback which also presents a novel screenshot-based slide-detecting technique for automatic feedback contextualization.
 - Design implications for group slide-feedback interfaces derived from formative interviews and an evaluation applying SlideSpecs to eight talks across different research groups and topics and an in-depth analysis of our findings.
- Recomposition Support for Vector Graphics (VST, VLT):
 - VST, a design tool that introduces a novel user interface for interactive, user-guided, flexible style transfer for vector graphics. Its key interaction principles are: a) enabling users to edit computed correspondences at multiple levels, and b) enabling users to customize how attributes are transferred between designs across the correspondence.
 - Two VST user studies that demonstrate: a) that designers can successfully transfer styles between graphics with VST, and b) that designers without VST can spend more time and effort to produce equivalent design results.
 - A description of a pipeline for interactively optimizing visual layout transfer across designs and VLT, a novel tool that implements this pipeline, and a gallery of example vector graphics results generated with VLT.

1.2 Overview

This dissertation is organized into these remaining chapters:

Chapter 2 (Background) provides background on *enhancing visual media*, creating data representations across domains, finding correspondences in visual media, and key user interface components for viewing and interacting with this data.

Chapter 3 (Related Works) covers related works from creativity support tools, presentation software, and vector graphics tools.

Chapter 4 (SlideSpecs) details the findings from formative interviews around presentation preparation, technical implementation details for SlideSpecs, a system to support reflection via collecting and organizing presentation feedback, and findings from an evaluation with several research groups and classes using this project.

Chapter 5 (VST) describes VST, a system for transferring visual styles across vector graphics, a semantically-rich 2D visual format. It also showcases a system evaluation with ten designers, exploring VST’s potential to transform externally-created designs. VST shows a significant performance boost over traditional vector graphics tools.

Chapter 6 (VLT) describes VLT, a system for transferring document-level layouts across vector graphics. Design layout can be a complex constraint satisfaction problem, yet designers often match existing layouts or transform their design’s layout. VLT leverages design correspondences to facilitate this layout transfer process.

Chapter 7 (Conclusion) has a summative review of this dissertation, a restatement of the core thesis contributions, and a more in-depth reflection on steps for future work.

1.3 Authorship and Prior Publication

Multiple components of this dissertation were collaborative efforts that were previously published throughout my dissertation, including SlideSpecs at IUI 2023 [153], VST at UIST 2023 [152], and VLT at the ICML 2023 AI/HCI Workshop [154]. Though I led these projects, I greatly benefitted from my paper co-authors and their contributions: Amy Pavel, Tonya Nguyen, Maneesh Agrawala, Frederick (Kyu Won) Kim, Shuyao Zhou, and Björn Hartmann.

Chapter 2

Background

The core of this dissertation is on *enhancing visual media* through reflection and recombination. This background chapter will provide an overview of the themes and ideas that motivate this dissertation’s research to provide context for the remaining chapters. I will first reflect on the general process of using multiple forms of related data in rich cross-domain representations, provide background on relating multiple forms of media via correspondence, and lastly, cover general user interface attributes and themes that span these projects.

2.1 Augmenting Media with Cross-Domain Representations

A large branch of my graduate research has been dedicated to developing and generating cross-domain representations for complex forms of data. One of the first projects I worked on, called Bifrost [96], was centered around helping electronics developers understand the relationship between their system’s code and any physical state or peripherals interfacing with their device. I also later worked on a follow-up project, WiFrost [97], which incorporated even more breadth by including networked connections. In addition to hardware and software, we looked at instrumenting routers to observe and track packets being sent along with server architecture to collect application state. This theme extended to SlideSpecs and collecting feedback across a range of sources (i.e., different audience members) and feedback types (e.g., written, spoken).

The goal here generally was to represent a complex system or process with behavior/data in multiple domains in a more meaningful and ideally more understandable representation to the developer. This process of constructing this representation to account for information spread across domains also heavily influenced the work I will present in my dissertation, despite these two projects not being part of my dissertation. I will draw out some of the higher-level takeaways from the collection of this work on cross-domain representations here.

Grounding Domain Knowledge

First, achieving success in building customized, cross-domain data representations for specific domains requires some grounding knowledge of that domain. This knowledge can be achieved by personal experience with that domain or by collaborating closely with someone working in that domain. Alternatively, if you (as a researcher and tool builder) lack domain experience, immersing yourself and gathering as much experience as possible is often quite useful. This experience can guide you toward what practitioners find important, how to interpret formative studies, and inform system design decisions. Making a new algorithm is not automatically enough to build practical tools for a given domain.

In my case, I had ample personal electronics development experience from my undergraduate research and internships (which aided with Bifrost and WiFrost) and personal experience preparing for presentations and helping others do the same (which aided with SlideSpecs). We also recruited and interviewed others working in these domains to help ground and refine our intuitions, though without these experiences directing those projects would have been far more challenging. Our personal domain experiences gave us a rough sense of the issues and types of data that working in those domains was concerned with, and how we might better collect and leverage it. This personal experience served as a rough standing and guiding force when making design decisions about what to prioritize and what types of data to collect and connect. It also helped us characterize current issues and standard processes faced by people working in these spaces.

While these are my personal experiences, I have also seen this pattern with a former labmate of mine, Eldon Schoop, who worked in ML development tools [130] – his own experience building ML systems helped him shape tools to support people learning to work in that space, or Kevin Tian who did a similar process but in the scope of computer-aided fabrication and woodworking [145]. Those seeking to create useful cross-domain representations are greatly aided by having some personal domain experience. That does not necessarily mean you *need* to be a domain expert to build cross-domain representations; more that having some personal experience will attune you to the most pressing, real user concerns.

Data Collection and Augmentation

Knowing the process or issues that people in a current workflow face can be informative for finding the most relevant data types/domains. However, logistics and complexity can greatly complicate collecting different types of data. For example, the expansive breadth of data that those systems measured was an essential contribution of both Bifrost [96] and Wifrost [97]. The holistic scope of ‘system performance’ they analyzed together spanned from trace-level voltage readings, microprocessor internal state variables, network routing code, and application server infrastructure handling requests. There is a significant engineering effort just to collect and group that information in an organized manner.

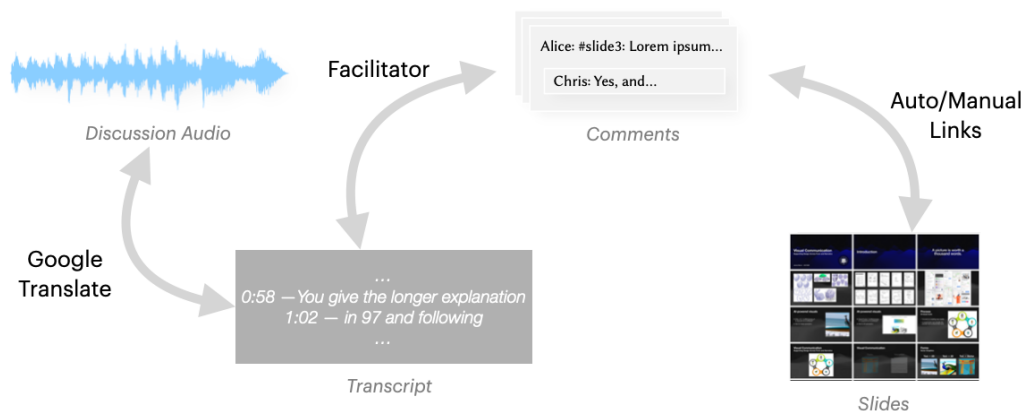


Figure 2.1: Data connected across several different domains in SlideSpecs.

For SlideSpecs (Chap. 4), we collected the presented slides, the audience’s written comments, and an audio file of the group discussion (Figure 2.1). We could have even collected videos of the presentation, biometric markers, or body language, though we decided not to. Tuning the cost-benefit analysis on what to collect and omit is essential to achieving success in this cross-domain media representation task.

Data augmentation, or refining and building up data collected in a single domain, is also very relevant for having high-quality information and dense representations to show users. An example of this could be building up a more flexible representation of vector graphics that is more amenable for document comparison than the standard SVG format. In addition to the original base representation, an additional augmented form of that data becomes available that might be useful in specific tasks (like style transfer (Chap. 5) or layout transfer (Chap. 6) for scalable vector graphics).

Connecting Data across Domains

Collecting several data types does not automatically create the maximum possible utility for cross-domain representations. A larger question here stems from how the data collected across domains is actually related. In the previous electronics debugging systems I brought up, *time* was the critical dimension that connected relevant data. Part of the challenge of WiFrost, which spans across sub-millisecond trace-level updates and potentially seconds-long network requests, was deciding how to (a) link time stamps across domains (e.g., like a movie clapperboard linking audio and video¹) and (b) represent data across different orders of timescale to enable users to make sense of what was recorded.

¹<https://en.wikipedia.org/wiki/Clapperboard>

Another concrete example of connecting data across domains is SlideSpecs, where we capture and relate data across a broad range of domains. An illustration of this is shown in Figure 2.1. We record and save the discussion audio file, generate a transcript to link the audio file per transcribed word, link written comments from the presentation with spoken discussion comments, and link written comments back to the presentation slides. Here, other domains besides time become relevant for organizing and linking data – like which discussion topic or written comment to link, or which presentation slide to link.

Different domains will have unique critical ways of linking data between them. One fundamental way SlideSpecs makes feedback more useful is by recording activity behavior across domains, then developing a shared context from which both the audiences and presenters benefit. These connections allow for building and studying new interactions, highlighting the more holistic and flexible representation, and the benefits it may provide users.

Connecting data across domains is enough to build a holistic representation of a singular complex process to support reflection (e.g., preparing for presentations). However, another relevant consideration is the comparison of multiple instances of these cross-domain representations. It’s worth considering the workflow of comparing two versions of these constructed representations. In the case of SlideSpecs, beyond connecting data across domains, it may be helpful to see how feedback changes over time between practice sessions, or even for different talks over time (e.g., are my presentation skills improving over time?).

This data connection is also relevant for my dissertation’s visual media *recomposition* focus, detailed in Chapters 5 and 6. In addition to constructing an augmented graph-based representation of the design, we construct and leverage connections across designs to support *recomposition*. The following background section delves into that specifically for the domain of vector graphics.

2.2 Finding Correspondences Across Visual Media

There is a critical difference between building up an augmented media representation and finding ways of relating instances of media to each other. While SlideSpecs combines and connects behavior across domains to support *reflection*, it isn’t a design tool with the means to update the presentation, nor can it compare feedback for different *versions* of talks. Supporting *recomposition* is far better exemplified by VST and VLT, which connect and transform related elements across vector graphic designs. This relation of elements across designs is called a *correspondence* – essentially a mapping from one set of design elements to another. There is a fair amount of background that VST and VLT directly use to function and other highly related work that motivates this research direction, which I will detail here. First, I will cover correspondences for rich 2D designs (i.e., vector graphics, web UIs, mobile UIs), provide more details on the computation of graph-based correspondence algorithms, and then touch on rasterized image correspondence techniques.

Rich 2D Designs

This dissertation focuses on complex media like vector graphics and presentations, so I'll first detail work that centers around 'rich' 2D designs; meaning visual media that supports editable object properties (e.g., fill, text, font). The most related work here is a project I contributed to during an internship at Adobe Research [136]. This project focused on (a) constructing a novel, graph-based augmented representation of vector graphics and (b) using this representation to match related elements across designs. This work features a graph construction method that takes a vector graphics design as an input, a graph-kernel method that computes a table of cross-design element pair similarities, and finally, an algorithm for building a correspondence from this similarity table. The base method for the graph construction represents each unique design object/element as a vertex in the graph. So, for example, each path or line of text would be considered a single vertex in this graph. Additionally, edges are added between vertices where a semantic relationship is present, like having the same color/font, having the same aligned bounding box, or semantic layout information (e.g., is left of, is above, is right of).

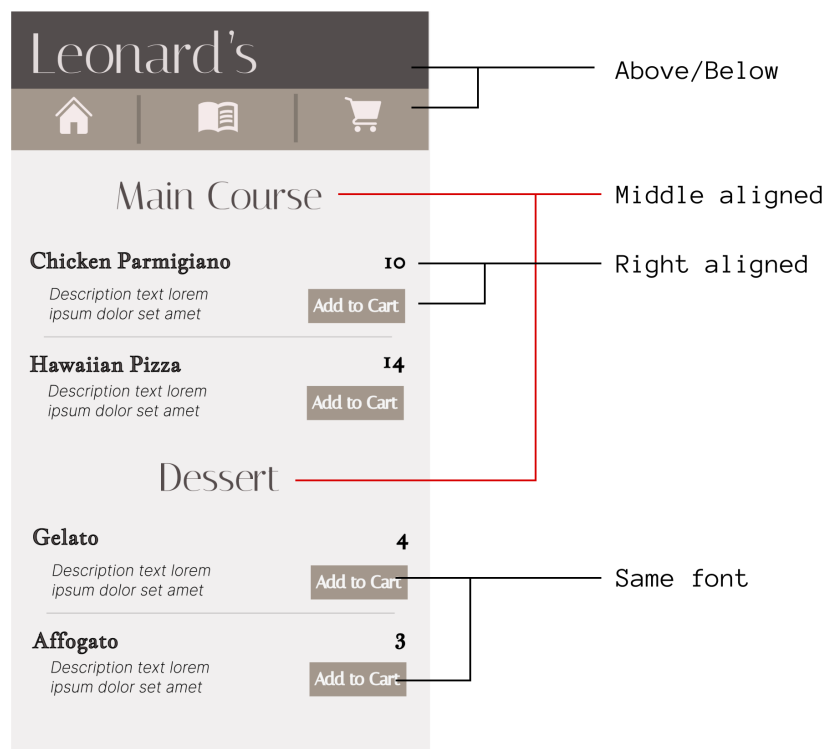


Figure 2.2: Rules that can be inferred within a design. These displayed rules (and more) are explored in detail by Shin et al.'s vector graphics graph walk kernel project [136].

Two main types of relationships are included in this graph construction technique. The first is element-element comparisons, where no additional context is needed to ascertain the relationship between the two elements. The other is a relational edge test, which may consist of directed or undirected edges corresponding to objects’ relative positioning. In some cases, these positional attributes are reciprocal, like being centered, and in others, they are asymmetrical (like contains vs. is contained by).

The vertex and edge kernels are also weighted based on relative frequency, with more common values/relationships having less discriminatory power. Then, a graph-walk kernel computes a numeric score based on the paths stemming from each pair of elements across designs. Finally, the element matching algorithm uses design-specific features to boost the matching results, including first iteratively matching the most uniquely similar elements.

With correspondence-based transfer tasks, even if a satisfactory match is computed between elements of two graphs, the specific sources of style that one might want to import to update their target design could vary drastically, even for the same two designs. For example, designers might import a color scheme from a certain design but not change the layout, or might want to update the font faces for a specific subheader but keep the color. The notion of a *correct* style transfer in this way is inherently limited, and we are left thinking about how we can do things that make style transfer tasks that would be otherwise performed manually much more effective and enable more rapid and new ideation of style concepts that otherwise might not be mixed.

While this graph-kernel-based approach is the most relevant related work, other rich media comparison tasks exist. Bricolage [76] also addresses a similar challenge but for the domain of website designs (HTML/CSS). They incorporate page segmentation to standardize document object model (DOM) tree formatting. They crowdsource ideal human mappings of elements across pages to train an edge-cost estimation network using a generalized perceptron algorithm for structured prediction [20]. Instead of graphs, websites are modeled as trees, and computing the correspondence between trees is formulated as a cost-minimization problem.

Beyond HTML websites, Rico [23] provides a dataset of mobile UIs that can be used to project designs into a constructed latent space. This latent space can then be used to retrieve other example datasets from the training corpus, generate layout options, interpolate between a pair of designs, and more. They also incorporate aspects of interactivity, which, while essential, can complicate the state required to model a user interface. With that said, vector graphics do not have the rich, structured, and labeled extensive set of available examples of website and Android apps that partially enabled these other projects’ successes.

Computing Element Connectivity

What sort of metrics or properties of these visual design elements do we care about? Ultimately, we want a general function that describes how similar a pair of two elements are. We also care about patterns: the functional grouping of multiple design elements. I'll discuss considerations for both design aspects and strategies for computing them here.

First, we want a comparison function C that takes in two design elements and yields a real numeric result that can be used as a metric for the similarity of those two elements:

$$C(d_1, d_2) \rightarrow \mathbb{R}$$

To do this, we will create a transformation T from the implicit space of design elements into a real higher dimension latent space:

$$T(d_i) \rightarrow \mathbb{R}^N$$

Once we compute this transformation T , a fast comparison of two elements can simply be some distant metric between those elements (e.g., the p-th order Minkowski distance):

$$C(d_1, d_2) = \left(\sum_{i=1}^N |T(d_1)_i - T(d_2)_i|^p \right)^{\frac{1}{p}}$$

Similar elements should remain relatively close to each other in the latent space (i.e., the transformation should be approximately linear). For example, if one were to adjust the size of a copied element, the more they were modified, the size the farther apart those elements should be in the latent space. Abstractly, we can say some minor element design transformation DT should retain similarity (ϵ):

$$|C(d_i, DT(d_i))| < \epsilon$$

To refine our transformation T , we should also include *contextual* information about how the element relates to other elements in the design. For example, is it the largest text element? Does it use a similar font to other elements? Is it contained by or aligned with other elements? This context can serve as a set of clues to help us refine our understanding of what 'role' the element under consideration plays. Given an unstructured document, how can one infer the relative relationship between different elements? We'd not only want to identify patterns within the document, but also to describe how similar a pair of patterns are. Conceptually, this is like learning a class hierarchy or set of templates that might otherwise be explicitly encoded into documents (e.g., HTML).

Comparing Graphs

The larger landscape of graph comparison techniques is worth considering as we model vector graphics designs as graphs. In addition to the graph-walk approach for vector graphics [136], earlier work also analyzed 3D virtual scenes by modeling them as a graph [34]. Here, vertices were again individual objects, though edges only represented some tangent surface touching (e.g., a lamp resting on a desk). They also used graph walks to compare the similarity of the two graphs. Outside of these works, graph machine learning focuses less on creating a correspondence between the graph’s vertices. Instead, they focus more on graph classification or similarity metrics, often where a desired class structure is known beforehand. In the case of design graph kernels, we construct edge types where the semantics of the design are not necessarily known ahead of time. Different graphs of the same visual design may encode different structural information based on their organization and layout.

We can also take inspiration from other domains that leverage graphs for computational comparison. These domains include protein analysis, social network analysis, and 3D geometric mesh analysis or defamation. In many of these techniques, the discrete Laplace operator² is used to measure multiple degrees of connectedness between vertices, which can identify typically characterize subpatterns within the graph (e.g., an amino acid group in a larger protein). Alternatively, one might look for a key connecting figure between multiple social groups or a functional unit like a couple or a parent-child relationship. Specifically, the multiscale Laplacian graph kernel [71] introduces an idea of a feature-based Laplace graph kernel that enables the Laplacian operator to be applied to graphs with different numbers of vertices. This enables operation on graph pairs of different sizes, essential for any transfer-type task. Feature Laplace graph kernels project the joint Laplacian matrices into a shared basis between the two graphs, enabling their direct comparison. More recent approaches leverage the embedded Laplacian distance [142] and embedded Wasserstein distance [70] to characterize graphs.

Comparing Images

Computer screens are 2D, meaning any visual media rendered onto them has a corresponding 2D (image) representation. While element hierarchy information is available in vector graphics and presentations, the techniques discussed so far have primarily omitted this while matching. We can explore image-based techniques for determining semantic connections between two pieces of visual media. Creating correspondences (connected points across images) is one of the most fundamental challenges in computer vision. Many approaches look at per-pixel image values or identified features and measure how those points vary between related images. Correspondences enable popular camera features to create panoramas with a perspective outside of a single camera’s viewbox. This can also be extended to movies with

²https://en.wikipedia.org/wiki/Discrete_Laplace_operator

retention across frames for consistent shots without camera cuts. It also enables semantic keypoint tracking³ like facial or body movement.

In computer vision, techniques like InstructPix2Pix [10] have made remarkable progress in pixel-based image transformation. Still, vector graphic renderings remain an untapped representation for the recomposition of media. This is partially true due to the lack of high-quality labeled example data, the time required to create SVGs or other vector graphics, and the relative difficulty of applying automatic techniques to a more complex image representation than a group of pixels. Some techniques for generating rich 2D media from images or text [61, 93] may change this as more example sets become available. This work seeks to leverage some automatic raster-based style transfer techniques that have revolutionized media creation and expression in the vector graphic design space while retaining and leveraging the contextual information that rich 2D media can include.

Previous research has integrated image-based comparisons for UI design tasks by introducing a screen correspondence approach that focused on mapping interchangeable elements between user interfaces [161]. This approach involves screen categorization and employs a UI Element Encoder to reason about the relative positions of UI elements. For UI layout, the Spotlight model [84] adopts a vision-language approach. This model takes a combination of inputs, including a screenshot, a region of interest, and a text description of the task. These inputs are processed using a multi-layer perceptron and a transformer model, resulting in a weighted attention output. The output of the Spotlight model can be utilized for UI design tasks like widget captioning and screen summarization.

2.3 UI for Augmentation and Correspondence

Relevant to both visual media reflection and recomposition is bridging the gulfs of evaluation and execution [105] from correspondences and their related media. Viewing, understanding, and modifying identified data augmentations and correspondences is critical. This goal is generally achieved with domain-specific user interfaces and visualizations. Domain-specific views and controls exist for both visual data augmentations (building up additional useful representations) and correspondences (relating two pieces of visual media together).

For this dissertation, **SlideSpecs** serves as a binding interface for the augmented data representation that spans across domains. In contrast, **VST** and **VLT** serve as examples of interfaces for interacting with and leveraging correspondences across instances of media. These systems and their related user interfaces are discussed in detail throughout Ch. 4-6. Here, I will outline the more general features that user interfaces for supporting data augmentation and correspondence should contain.

³https://en.wikipedia.org/wiki/Scale-invariant_feature_transform

Augmentation

For visual media augmentation, user interfaces often overlay generated or collected tags or points of interest onto the scene. User interfaces for matching standard templates (e.g., facial key points) may directly overlay relevant data on the matched image. Another automatic feature point detection algorithm, SIFT, can identify points of interest and often plot them directly over the image. Other times, object masking and object segmentation can be overlaid to inform the author of their generated data augmentation results. Note that these features should also ideally be composable, meaning their functionality should overlap and enhance each other’s power. Core interface features for supporting media augmentation include:

Bidirectional Links (Across Domains) Data points connected across domains should be able to toggle or highlight other related activity across domains in either direction.

Filtering Being able to focus on a subset of the cross-domain data representation for deeper inspection (e.g., view comments on a specific slide or from a time range).

Sorting Being able to sort aspects of the cross-domain representation by computed inherent (e.g., by scores, authors, timestamp) attributes of the data.

Correspondence

Given the comparative lack of 1-1 pixel-based connection that single images provide for presenting augmented data, there is a more varied set of approaches for visual media correspondence. For stitching images of a shared scene together (e.g., like a panorama), multiple common keypoints are set across the image to compute corrections across them. Core interface features for supporting media correspondences include:

Bidirectional Links (Across Media) Corresponding elements across visual media should be displayed and be able to highlight other related data points (e.g., related elements).

Similar Selections Given a distance metric for elements, interfaces should be able to query similar elements ‘by example’ – or based on a demonstrated user selection.

Composite Views Where possible, media should be overlaid to make visually distinguishing differences and relations between media spatially explicit (e.g., overlaying design graphics with related media, overlapping keypoints of shared images).

Results Visualization If the correspondence also powers a transformation of some type, the result should be previewed and editable (e.g., the output canvases in VST/VLT).

Chapter 3

Related Works

As this dissertation seeks to support the broad process of visual media refinement and re-composition, there is a wealth of related works. I will summarize the most related aspects of this prior research ranging from systems that support collecting and organizing feedback, generating context for verbal discussions, inferred structures in visual designs, generating novel design layouts, supporting creative processes with AI, and review a more general set of related tools for supporting vector graphics design.

3.1 Collecting and Organizing Feedback

One of the key elements of enabling reflection is to collect and organize feedback. There are several related projects which also provide examples of this task, which I'll detail here.

Past works have addressed providing synchronous, textual feedback written during in-class presentations [134] through a forum-style interface and critique during design reviews [107, 57, 109, 68], though they do not capture the post-presentation discussion, nor does they contextualize feedback in the presentation. While existing online slideware (e.g., Google Slides) may allow multiple users to comment on talk slides simultaneously, these platforms do not facilitate references between presentation feedback and post-presentation discussion. This research diverges from prior work by focusing on the domain of practice presentations to small groups, where long in-person discussions after the talks are common.

Prior work considered tools to improve the production and presentation of slides primarily by reducing the rigidity of linear slides, providing additional context for slide material, or slide generation tools [89, 4, 79, 138, 90, 41, 133]. Such slideware lies on a spectrum from traditional slide presentation tools (e.g., Google Slides, PowerPoint, Keynote), which provide a linear sequence of rectangular slides, to IdeaMaché [90], which allows freeform presentations in the form of a navigable collage. In between lie tools like Fly [89], NextSlidePlease [138], and CounterPoint [41], which each allow the user to arrange slides in a 2D

space and enable linear or branching paths through the space during presentations (to allow presentations to change dynamically based on time or audience feedback). Drucker et al. demonstrate an effective way to track presentation changes and refinements over time [29]. Other tools focus specifically on the presenter’s oration rather than general feedback [108, 3]. To increase feedback context, DynamicSlide [63] infers links across presentation videos and slides while others [114, 115] link slide text and images to presenter speech and promote accessibility. As SlideSpecs lets audiences provide feedback on uninterrupted rehearsal presentations, SlideSpecs supports any linear presentation. It also lets audiences peek forward or look back at previous slides to craft general presentation feedback.

Many systems exist for recording and reviewing critique in asynchronous and synchronous scenarios for different media (e.g., single page graphic designs [92], PDFs [170]) and audiences (e.g., crowds [92], peers [134, 74], teams [113, 107]). For instance, prior work has addressed capturing and organizing critiques [92, 159, 134, 107, 113] primarily in asynchronous settings where all communication is mediated through the critique interface. However, one unique aspect of practice talks is the synchronous setting in which all participants see the talk simultaneously and then discuss feedback post-presentation. Past work has studied in-person collaboration on creative tasks [155], the use of Google Docs for providing feedback in classes [134], and the use of Google Docs compared to in-person meetings for design collaboration [64].

Using SlideSpecs, audience members write critiques using real-time messaging on a shared chat channel, promoting specific critiques and threaded conversations. Researchers have also suggested techniques for improving peer and novice feedback, including introducing rubrics [92, 74, 146], computing reviewer accuracy [19], and providing live suggestions on the feedback [38]. While these prior works focus on supporting critique from peers and providing guidance on giving better critique, our research emphasizes supporting critique in a group setting where participants often have more experience and motivation. The work that focused on helping experts provide asynchronous feedback, both in small teams [170, 171, 113] and classrooms [40], hasn’t addressed the larger question of how to organize feedback from multiple critiques across media domains. We aim to address this open question: how can we effectively support synchronous critique from an experienced audience in a group setting across multiple domains of media?

One core issue when receiving feedback is the sheer amount of feedback. Automatic text summarization performance continues to improve, enabling new possible applications [137, 135]. Text summarization techniques have been applied across many domains (outside of meetings) to help the reader, including news [78, 166], sports [104], food reviews [21], auctions [51] and email [28]. Efficient text summarization can also be applied to real-world physical documents for accessibility needs [7]. SlideSpecs provides the audience with shared context to help reduce redundant comments.

3.2 Contextualizing Verbal Discussions

As SlideSpecs helps presenters record and organize spoken critiques produced during a discussion phase, we build on prior work in supporting meeting discussions. Producing a reusable record of meeting events remains a challenging task. Early work found that meeting notetakers faced difficulties including dropping crucial facts, using uninterpretable abbreviations, finding too little time to write notes, and missing participation opportunities while notetaking [66, 157]. We also find it challenging for presenters to write down critiques during presentation feedback sessions while listening and responding to the conversation. To improve meeting recall despite spotty notes, prior work recorded the meeting and then linked handwritten [157, 18] or typed meeting notes [17] taken during the meeting to their media timestamps. Other work indexed meetings by the speaker-segmented transcript [156], by domain-specific artifacts [99, 39, 24] referenced during the meeting (e.g., slides [99]), or by automatically extracted important moments [101, 103, 82].

We investigate how to let the presenter focus on organizing text-based critiques, using the meeting transcript primarily to provide more fidelity for written critiques. To enable this, we investigate how to help the facilitator efficiently establish links between discussion segments and relevant existing text critiques during the meeting. Unlike following an agenda with few items [39] or writing notes directly [157, 18, 17], locating specific text critiques requires the facilitator to search for particular comments and the audience to promote specific comments for discussion.

Our work relates to prior work on using group discussion chats during a presentation and organizing chats for later review. Prior work on digital backchannels [95] during presentations involves either forum-style promotion of questions and feedback [134, 121, 46] or real-time chat [125, 117]. In contrast to prior work that investigated general real-time chat during a presentation, we explore the use of real-time chat to collectively generate slide critiques where threading can be used for critique elaboration rather than general chat. To use real-time comments for later review, we allow lightweight markup of text chat similar to Tilda [172, 173]. Instead of focusing on general meetings, we explore how to design lightweight tagging of comments specific to critiques (e.g., location, critique type).

3.3 Inferring Structures in Visual Designs

Researchers have used several approaches to infer underlying or implicit structures in visual designs. Traditionally, this work primarily operates on some structured representation (like HTML or SVG). For website and user interfaces, large collections of example designs have been used to characterize and infer document structure [23, 75]. Linking styles via direct manipulation and element cloning provide a clear view and control of an element’s style properties [52]. There is also work to recognize higher-level design patterns through designs

with Bayesian grammar induction [141]. For the domain of D3 visualizations, Hoque et al. map data types onto shapes/axes to help search for relevant designs [54]. Harper et al. showcase tools for deconstructing and restyling a D3 visualization by extracting the data and modifying visual attributes of marks [45].

More recent work also focuses on inferring design structure from images directly. Computer vision techniques are improving on reverse engineering user interface models from screenshots [163, 132, 32]. Similar work using vision-based methods has leveraged attention towards answering questions and understanding mobile UIs [84, 140, 131]. Reddy et al. use differentiable compositing to identify pattern instances within a design [124]. Scene graphs have also characterized structural relationships within and between 3D environments [34].

For vector graphics, Shin et al. demonstrate a technique using graph kernels to find relationships between elements of designs [136]. We leverage this preexisting automatic technique to compute a correspondence between design elements (like those shown in Fig. 5.2). The contribution of this work centers on our novel design tool that goes beyond pure algorithmic automation by enabling flexible interactions between the capabilities of such an algorithm and the designer’s high-level styling goals. This approach enables efficient editing of vector graphics by computing element-to-element correspondences at different levels of granularity. Building upon these existing approaches, our work incorporates a graph-kernel based method for inferring objects and computing correspondences across canvases. We can leverage the structural information of the designs to establish correspondences and perform efficient inference across multiple graphic designs.

Recognizing design patterns plays a crucial role in a range of layout tasks. In recent years, deep learning models have been proposed to address different aspects of vector graphics, including inference, generation, and editing [44, 5, 88, 61, 93, 80]. For UI design tasks specifically, previous research introduced a screen correspondence approach that focused on mapping interchangeable elements between user interfaces [161]. This approach involves screen categorization and employs a UI Element Encoder to reason about the relative positions of UI elements. In the domain of UI layout understanding, the Spotlight model [84] adopts a vision-language approach. This model takes a combination of inputs, including a screenshot, a region of interest, and a text description of the task. These inputs are processed using a multi-layer perceptron and a transformer model, resulting in a weighted attention output. The output of the Spotlight model can be utilized for various UI design tasks (e.g., widget captioning, screen summarization).

3.4 Generating Design Layouts

Prior works have explored different approaches for layout generation and manipulation. Datasets such as Rico [23] and WebUI [162] can be used for training probabilistic generative models of UI layouts. Recent approaches explored transformer-based models in generating layouts [83, 2, 72]. With Im2Vec, researchers used differentiable rasterization to vectorize raster images and interpolate between them [123]. Others learned implicit hierarchical representations for vector graphics to aid generation tasks, though they have focused on simpler designs (e.g., fonts) [14, 91]. For layout transfer task, the Bricolage algorithm [76] employed a technique for generating mappings between Web pages by dividing them into significant regions and rearranging the elements to reflect parent-child relationships within a tree structure. However, it specifically focuses on HTML/CSS content and does not encompass visual layout transfer for vector graphics. Also, the wealth of example website designs that Bricolage could leverage for training is comparatively scarce for vector graphics.

DesignScape provides users with layout suggestions, improving the efficiency of brainstorming designs [106]. Li et al. used the idea of Generative Adversarial Networks and proposed a differentiable wireframe rendering layer, specifically improving alignment sensitivity and better visibility of overlapping elements [85]. Ye et al. [167] proposed Penrose that aimed to create mathematical diagrams using a layout engine that compiled code into layout configurations with the least overall energy while satisfying constraints and objectives. Cheng et al. [16] presented a latent diffusion model PLayer and conditioned on user input guidelines to generate UI layouts. Chai et al. [15] introduced the LayoutDM framework, which utilized a transformer-based diffusion model to generate layouts by representing each layout element using geometric and categorical attributes. This model employed a conditional denoising diffusion probabilistic model to gradually map samples from a prior distribution to the actual distribution of the layout data. Like Chai et al. [15], Naoto et al. [56] utilized diffusion models to generate layouts. Additionally, Dayama et al. [22] proposed an interactive layout transfer system that allowed users to transfer draft designs to predefined layout templates. Their approach relied on an interactive user interface and an existing library of layout templates. However, the system required that the component types be predefined and rigidly categorized as either headings, containers, or content. Our tool can transfer the user-input target layout onto the source design while retaining layout rules and consistency inferred from the designs, giving more flexibility for design tasks.

3.5 Supporting Creative Processes with AI

While automation is powerful, gracefully integrating it into existing creative practices demands care. Regarding working with AI as a design material, scholars have elaborated on the need for retaining control [127, 169, 139, 112, 147]. For GUI design, Dayama et al. present a method for interactive layout transfer, where the layout of a source design is transferred

automatically using a selected template layout while complying with relevant guidelines [22]. In photography, researchers have provided mechanisms for guiding photographers to optimize image aesthetics [77] and to find ideal portrait lighting conditions [31]. Goal-oriented transformations can also be applied to existing designs (e.g., improving accessibility) [174] or to produce alternative designs for different viewports [53].

Our rationale for using element relationships between designs as a primary mechanism for transfer is that this mirrors how designers tend to work already when manually transferring styles. Highly related to our line of work are feedforward and example-driven corrections. Feedforward work refers to showing the user the output or result of their action before it happens—a preview of applying different interface actions [149, 27, 65]. For example, OctoPocus provides dynamic guidance to bolster users’ ability to learn stroke-based gestures [6]. Example-driven corrections and interaction models like those in FlashMeta [118] or programming-by-demonstration disambiguation models [94] provide alternative techniques that address similar problems.

Feedforward and inherent feedback can promote UI element functionality understanding to users, though computing this information fast enough for live, interactive contexts can be challenging. With that said, cluing in authors on the impact of their actions is valuable. For example, the Lightspeed rendering pipeline enabled interactive prototyping of professional 3D graphics, enabling more design variation exploration [120]. One approach might leverage lower-fidelity previews of variations when interacting with automation, such as design galleries [81]. We avoid using design galleries as our early prototypes showed the varying complexity and breadth were visually overwhelming.

Example-based corrections generate a program that satisfies all demonstrated changes, iteratively growing more complex. Example-based style retargeting for websites provides a successful analog to vector graphic style transfer in HTML/CSS [76, 8]. Example galleries can effectively support open-ended design authoring, where styles come from potentially multiple sources [81]. While the document-object-model hierarchy is essential to styling web pages, such grouping structures and labels are entirely optional and often absent in vector graphics. Groups may be constructed arbitrarily (e.g., for editing convenience) rather than having any consistent semantic meaning. Designers can encode hierarchical information through groups but frequently opt to style elements directly [127]. Vector graphics present a unique challenge regarding general, interactive style transfer.

While automatic style transfer techniques can generate impressive image transformations, they are generally functional as *theme selections*. Due to the broad range of shape primitives, graphic designs do not immediately lend themselves to this document-level style transfer approach. The selective extraction and transfer of specific styles are too precise to be encoded in a one-dimensional slider [58, 62]. The variations of vector designs also make mapping onto an otherwise standard template difficult (e.g., facial key points) [144].

Additionally, text can be used to edit image content and style directly [10]. While layout is not our tool’s focus, prior work highlights optimization techniques that can be used to automatically format text documents [55]. ImagineNet restyles mobile apps with neural style transfer and updating assets in place [33]. To be stylized with image-based techniques, vector graphics must first be rasterized, losing future object-level awareness and scaling abilities.

The state of the art in automatic vector generation includes leveraging pixel-based diffusion models [61] by leveraging a differentiable vector graphics representation [88]. DeepSVG uses GANs to generate and interpolate between SVG icons and shares a large-scale SVG dataset [14]. Kotovenko et al. model a painting using discrete strokes to recreate style transfer better [73]. Within font, some work shows the possibility of even inferring and transferring style between font glyphs [25, 91]. These techniques often give users little to no control of *how* the style is transferred. Our work focuses on optimizing the potential value that these automatic approaches can provide by introducing meaningful high-leverage interactions to customize and control generated output while retaining the core vector graphics representation that designers are familiar with working with.

3.6 Vector Graphics Design Tools

Several techniques for authoring and transforming vector graphics design exist and inform this work. Object-Oriented Drawing introduces a new way to create and style elements directly on the canvas [164]. DataInk supports cloning and binding user-generated symbols to data, facilitating lightweight restyling [165]. Sketch-n-Sketch links drawing code and vector graphics, letting users directly edit the SVG in a canvas, modifying the code which generates it [49]. For mathematical diagramming, Penrose uses layout energy-minimization techniques coupled with a language for specifying explicit styles and content of what to render [168]. Falx uses user demonstrations and program synthesis to create new visualizations [151]. Existing tools can even convert web designs into a vector layout [26].

Para supports binding procedural art generation constraints with graphics, including cases where there are many-to-many constraints [60]. A follow-up project, Dynamic Brushes, combined procedural programming into brush behavior and design, enabling more custom expression [59]. Other design tools have looked at supporting design layout [67, 106], fashion [148] and design coloring [176, 48], or using AI as a source of variation when creating [30, 80]. To understand and improve the legibility of the brushes’ dynamic behavior, researchers created inspection tools for this authoring process [86]. However, this past research focuses more on procedural art creation rather than style transfer between existing graphics.

Chapter 4

SlideSpecs

Presenters often collect audience feedback through practice talks to refine their presentations. In formative interviews, we find that although text feedback and verbal discussions allow presenters to receive feedback, organizing that feedback into actionable presentation revisions remains challenging. Feedback may lack context, be redundant, and be spread across various emails, notes, and conversations. To collate and contextualize both text and verbal feedback, we present *SlideSpecs*.

SlideSpecs lets audience members provide text feedback (e.g., ‘font too small’) while attaching an automatically detected context, including relevant slides (e.g., ‘Slide 7’) or content tags (e.g., ‘slide design’). SlideSpecs also records and transcribes spoken group discussions that commonly occur after practice talks and facilitates linking text critiques to relevant discussion segments. Finally, presenters can use SlideSpecs to review all text and spoken feedback in a single contextually rich interface (e.g., relevant slides, topics, and follow-up discussions).

We demonstrate the effectiveness of SlideSpecs by deploying it in eight practice talks with a range of topics and purposes and reporting our findings. When surveyed, 85% of presenters and reporting audience members indicated they would like to use SlideSpecs again. Presenters reported that using SlideSpecs improved their feedback organization, provided valuable context, and reduced redundancy. A version of SlideSpecs for providing presentation feedback is available at: <https://slidespecs.berkeley.edu/>.

This work presented in this chapter was first published in Warner et al. as *SlideSpecs: Automatic and Interactive Presentation Feedback Collation* in the 2023 Proceedings of IUI, the ACM Conference on Intelligent User Interfaces [153].

4.1 Introduction

Presentations are a foundational way of sharing information with others across education, business, science, and government. Unfortunately, there are also abundant examples of ineffective or even misleading talks. Presenters obviously want to share the best version of their ideas as clearly as possible – so what goes wrong?

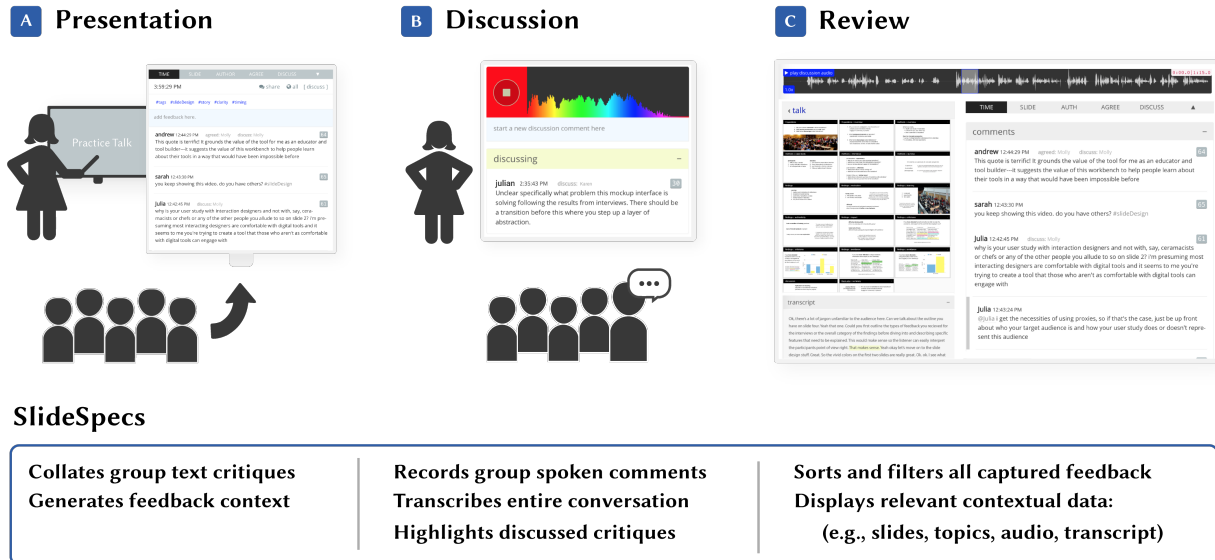


Figure 4.1: SlideSpecs supports presenters in three phases: Presentation, Discussion, and Review. During the Presentation, SlideSpecs collates audience text critiques referencing talk slides, feedback tags, or other critiques. During the Discussion, the interface records and transcribes the discussion audio. Participants can easily reference previous critiques, and a facilitator annotates discussion topics. During Review, SlideSpecs provides presenters with rich contextual data that links the feedback gathered from both previous phases.

Preparing an effective presentation takes time and work. The more a skilled presenter makes it look easy, the more work and talk revisions they've likely done beforehand. When you consider the potential broader impact of a talk, this work is justified. A 20-minute talk for a 60-person audience (e.g., a reasonable conference talk) consumes 20 hours of audience time. Talk videos can reach an even broader – a 600-view video of that same talk takes 200 hours of viewer time [9]. When framed this way, the time and work spent refining presentations are more clearly justified.

To improve the effectiveness of slide presentations, presenters often give practice talks to live audiences and receive presentation critiques. Presenters may receive feedback from audience members through discussion after the presentation and written text critiques (e.g., shared notebook, email).

To leverage audience feedback, presenters must record and distill critiques from an open-ended discussion, recall critiques and their corresponding contexts (e.g., relevant slides, audience-suggested solutions), and organize comments across multiple authors and mediums. Each task is challenging and potentially error-prone, especially given a limited amount of time for discussion and clarification. Presenters may lose track of relevant feedback, leaving potentially valuable critiques unaddressed.

To understand the presenters' challenges when refining talks, we conducted formative interviews with 14 participants. Received feedback ranged in *scope* (e.g., specific to a single slide vs. general presenting tips) and *subject* (e.g., slide design, narrative). Spoken feedback during and after the practice talk provided opportunities for discussion and clarification; however, bandwidth was limited (e.g., how many issues can be raised), individual discussions may not make the best use of the entire audience's time, and public discussion could bias or inhibit other audience feedback. Alternatively, audience members sometimes shared written text feedback with the presenter either privately (e.g., an email, index cards) or publicly (e.g., a shared online document). Text feedback provides a written record of critiques and offers more space to share concerns. Both verbal and written feedback benefit these presenters, but capturing critique context and organizing feedback across sources into a more valuable and accessible form remains challenging.

To support the automatic and interactive collation of audience feedback, we present SlideSpecs (Figure 5.1). SlideSpecs supports three common phases in the presentation revision process: (A) *Presentation*, (B) *Discussion*, and (C) *Review*.

In (A), the *Presentation*, the presenter uses their preferred slide software while the audience uses the SlideSpecs *feedback-providing interface* (Figure 4.2). This interface shows the talk slides, the speaker's current position within the slide deck, and the other audience critiques (optionally). Audience members can write critiques, include relevant *scope*, provide their critique's *subject*, reply to other comments, and flag other critiques for agreement or further discussion.

In (B), the post-talk *Discussion*, the presenter and audience can use the *discussion interface* to guide and capture conversations (Figure 4.4). SlideSpecs records and transcribes the conversation, including elaborating on existing comments and new critiques the audience may raise. A talk facilitator can dynamically associate each spoken audience comment with an existing text comment or designate the comment as a new point of critique (Figure 4.5).

In (C), the feedback *Review*, presenters access a contextualized record of all critiques from (A) and (B) in the *reviewing interface*. This interface shows transcribed discussion segments and displays these transcripts alongside relevant linked text critiques and discussion topics. Presenters can also sort, filter, and search all feedback with keywords, slide numbers, tags, and agree/discuss votes. The automatically generated context, transcribed discussion

segments, and audience-contributed tags help presenters better understand their critique and revise their talk accordingly.

SlideSpecs is the first system to collate and contextualize synchronously generated text and spoken presentation critiques in a unified interface. SlideSpecs collects and contextualizes text and verbal critiques across both practice presentations and group discussions. SlideSpecs uses a live group-chat style interface with lightweight tagging for feedback to support elaboration on other audience member critiques (Figure 4.2). SlideSpecs also provides a flexible set of filtering and sorting tools that presenters can apply to feedback across the entire process of practicing their talks.

To assess how using SlideSpecs affects presenters’ ability to leverage critiques for talk revision, we evaluate SlideSpecs across the presentation, discussion, and review phases (Figure 4.3). We demonstrate the effectiveness of SlideSpecs by deploying it in eight practice presentations (Table 4.1) and reporting our findings. We also reflect on several future benefits that further automation and text summarization may provide presenters (Chap. 7.2). Presenters reported that using SlideSpecs improved feedback organization, provided valuable context, and reduced redundancy. 85% of presenters and reporting audience members expressed interest in using SlideSpecs in the future.

Our contributions include:

- Design implications for slide-feedback interfaces derived from formative interviews
- SlideSpecs, a novel system for collating audience text and spoken presentation feedback
- A screenshot-based slide-detecting technique for automatic feedback contextualization
- An evaluation of SlideSpecs in eight talks across different research groups/topics

4.2 Formative Study: Existing Practices

Method

To learn more about existing practices for giving and receiving feedback on presentations, we analyzed existing guides for creating presentations and interviewed 14 presenters (N1-14). We found these presenters via university mailing lists that connect a wide range of domains. Our participant presenters represented a range of fields (e.g., AI, HCI, Optimization and Testing, Design, Student Government) and eventual talk venues (e.g., company presentations, conferences, workshops, job talks). All the presenters we talked to also provided feedback to others on practice talks in the past. Specifically, we asked presenters a

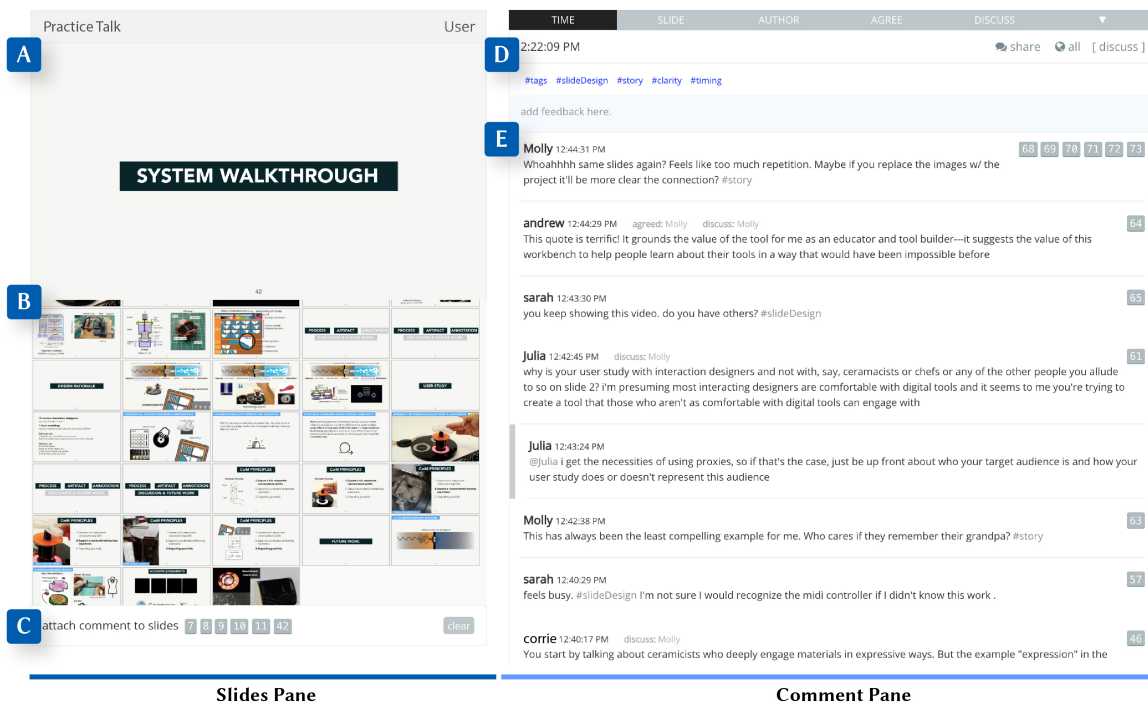


Figure 4.2: An overview of the SlideSpecs *feedback-providing* interface. In the *Slides Pane*, a detailed slide image view is shown (A), and hovering over the smaller slide thumbnails (B) temporarily updates the large image, current slide (A), allowing users to scan over talk slides. Audience members can select these thumbnails to attach their comments to designated presentation slides (C). In the *Comments Pane*, the interface features different ways of sorting the comments (D), a list of tags contained by comments, and a text area entering feedback. Last, the interface displays a live-updating list of the audience’s comments (E), which includes comment metadata (i.e., comment author, creation time, and slides referenced).

series of questions to find out: *What are the benefits and drawbacks of methods currently used to give and receive feedback on practice presentations?*

We recorded the audio of these formative interviews. We then categorized and reviewed participant answers into *role* (e.g., audience member, presenter), *phase* (e.g., feedback, discussion, revision), and *type* (text vs. spoken) before summarizing the categorized notes in our findings.

	Presentation (A)	Discussion (B)	Review (C)
Presenter	Deliver Talk	Verbal discussion addressing practice talk feedback	Review feedback in Review Interface
Audience	Share comments in Presentation Interface		
Facilitator		Manage discussion using Discussion Interface	

■ Done in SlideSpecs Interface

■ Existing in Practice

Figure 4.3: SlideSpecs supports three phases (presentation, discussion, and review) and three distinct participant roles (presenter, audience, and facilitator). The presenter delivers their practice presentation while audience members and the facilitator share comments using the *Presentation Interface* (A). During the discussion (B), the presenter and audience discuss feedback while the facilitator annotates the discussion. The presenter later reviews all feedback at their leisure in the *Review Interface* (C).

Findings

Presentation Feedback Sources

Participants reported using methods for giving and receiving feedback that fell into one or more of the following categories: (1) text feedback on the slide deck alone, (2) personal or shared text feedback during the practice talk (e.g., personal notes, email, Google Docs, shared whiteboard), and (3) spoken feedback at the end of the practice talk (e.g., in-person or via video conferencing). Two participants mentioned pre-talk slide deck feedback (slide design and story critique only); all but one mentioned in-session text feedback (6 using shared text feedback); all participants received spoken feedback. Most participants reported gathering feedback across all possible scopes, though one only sought feedback focused on the work itself (rather than any presentation-specific feedback).

Text Feedback

Participants mentioned that more audience members contributed to a larger quantity and diversity of feedback when using a shared text medium (e.g., Google Docs or shared whiteboard) or personal text notes (corroborating Jung et al. [64]). Reflecting on the quantity and content of text feedback, N3 noted that such feedback is more critical, can be a bit meaner, but that's important and even better since it's being given to help. Participants also mentioned preferring text feedback, particularly for specific lower-level critiques, as major feedback might require discussion: High level structural feedback wasn't useful in comments because I couldn't ask what they meant. (N2), the discussion gives presenters a better sense of what the audience is confused about (N8).

Participants reported that when preparing personal notes to deliver critiques, they take

rough notes during the presentation, refining the notes (e.g., rephrasing, ideating solutions, or removing comments brought up in discussion) before sharing them. For instance, N7 mentioned that her initial text notes are harsh and blunt and have to be reframed before sending them to the presenter for their review. On the other hand, when crafting shared notes, some participants focused all their time on the initial comment but did not revisit comments to revise before sharing them, reporting it feels like the responsibility of the presenter at that point (N14).

In addition, participants noted that in providing shared feedback, they still regulated their text feedback in the presence of other group members either to avoid the judgment of senior members (evaluation apprehension [155]) or to avoid duplicate work: I had the feeling that everyone was doing the same thing but better (N12), the advisor knows better (N2). Some participants found the shared feedback to be distracting or discouraging or spent time reading senior member feedback purposefully to learn how to provide feedback: First feedback session, didn't write a single thing, but I read a lot of the feedback because I don't know how to give feedback on this type of work...everyone is having all of these insights, but I'm not. (N11). Group dynamics influence the feedback process.

Verbal Feedback

Participants mentioned finding that verbal feedback provides the most opportunity for discussion: way more effective than Google Docs back and forth (N1). Still, it was very time-limited as only one speaker could voice ideas at a time (production blocking [155]). Participants emphasized that especially in time-limited scenarios, the most senior participants spoke most: the feedback process is almost like, oh the most senior person or the most opinionated person gives a high-level overview. Everyone else echoes and then adds on. (N5) With limited time, and senior participants speaking, participants cited that most spoken feedback consisted of high-level or delivery feedback: More high-level feedback with structure (N3), with the exception of lengthy in-person slide walk-through sessions or interrupting questions: in my research group, there's a big culture of asking questions mid-talk (N6). When in the role of giving feedback, participants mentioned that they considered several issues when considering whether to provide spoken feedback: how other participants may perceive their feedback: you want to be astute about it (N13), the importance of their feedback to improving the talk: Structure, flow, there has to be a story that's going on (N9), their familiarity with the subject, number of times speaking: I don't want to monopolize the conversation, once I say my two things (N13), and the ease of communicating a critique (e.g., whether the relevant slide appeared on screen).

Revision Process

When recording notes, both in the case of providing and receiving feedback, participants mentioned that they sometimes forget the relevant context of notes (e.g., what slide, what content, which audience member). But, presenters found such context important when

revising the presentation. When addressing feedback, participants suggested prioritizing based on several factors: (1) importance or trustworthiness of feedback author, (2) ease of fixing the problem (e.g., typos, font choice), (3) importance of the problem to overall argument, (4) Slide order (e.g., start with Slide 1), and (5) ease of remembering suggested problems (e.g., fix the problems that you remember first, and then go back to document to fill in the gaps). Most presenters formally organized their comments before revision: I make a to-do list after I synthesize all comments (N8), a to-do list and cross out items that are finished (N7), organized by a PM on google docs (N1). More experienced presenters expressed more selectivity in addressing comments: I just think about the notes I was given and try to catch the issues when I practice (N3). For those that organized their received feedback, the messiness (e.g., their own quickly handwritten or typed discussion notes or the group’s shared notes) and feedback source disparity (e.g., spread across emails) was a common complaint.

Classifying Critiques

In addition to our formative interviews, we also analyzed several existing guides for creating and giving effective presentations [143, 126, 102, 9]. We use each guideline from these guides as a possible source of critique. Two defining dimensions for critiques emerged from analyzing these guides: *scope* and *subject*.

Scope

Critiques often reference one of four types of locations in a slide presentation: a single slide (e.g., The results slide contains too much text), a contiguous set of slides (e.g., the motivation section could be shortened), a non-contiguous set of slides (e.g., switch the position of these two slides), or the entire talk (e.g., pause more often). We refer to critiques about any subset of slides as *local* and critiques about the whole presentation as *global*.

Subject

Presentation feedback falls into four main subject categories: *content*, *slide design*, *story*, and *delivery*. *Content* comments address the core of the presented work (e.g., consider interviewing the people that used your system). Such comments may express ideas or concerns about the work, but they also may reflect a misunderstanding or need for clarification within the presentation. *Slide design* addresses the visual layout of slide content, the amount of slide content, and consistency across slides. *Story* covers presentation structure, supporting examples, argument, and audience-presentation-fit. *Delivery* addresses aspects of giving the talk, including the script, the speed of narration, and gestures or body positioning. A comment in any of these categories may be positive (e.g., great use of color!) or critical (e.g., I don’t understand this example). These category labels are not mutually exclusive.

4.3 Design Implications

We discuss design implications determined from prior work and our formative work, then discuss our approach to addressing these implications with SlideSpecs. Grounded in our formative analysis, we present five key implications for designs in the space of recording and organizing presentation feedback:

Support existing workflows for group feedback.

Formative interviews reveal that groups provide feedback at different times (e.g., before, during, or after the presentation), using different media (e.g., text, spoken, digital, written), and with varying levels of privacy (e.g., public to group/private). Preserving a group’s social processes is advantageous in new technology adoption (Grudin’s third principle [43]), though this can drastically limit the space of acceptable designs. A balance should be struck between adapting to existing processes and effectively organizing feedback from diverse inputs.

Support a variety of critique contexts.

The context of feedback (e.g., authorship, critique type, slide) aids critique comprehension and prioritization during revision. Presenters and presentation guidelines reveal several important contextual features, and presenters will value these features differently.

Mitigate audience attention demands.

Audience members already manage multiple challenging tasks when providing feedback, including comprehending talk content, identifying critical issues, composing feedback, and managing social expectations. While audience members may be best suited to provide additional information about their critiques, a challenge exists in balancing any extra effort with present cognitive demands, especially as audience members are not always the primary beneficiary of the work [43].

Organize gathered feedback into action items.

Most presenters use a distilled list of action items to revise their presentation. However, they generally receive feedback via blocks of raw text and an open-ended, non-linear discussion of presentation issues. A design should support transforming these disparate inputs (e.g., text critiques and non-linear discussion) into usable, distinct action items.

Reduce note-taking demands.

Currently, presenters mainly capture and organize their critiques from discussions by taking notes. However, note-taking is demanding and error-prone: a well-documented challenge due to the high cognitive load of note-taking [110, 116, 158]. As the presenter frequently

participates in post-presentation discussions, they are suited particularly poorly to take high-quality, contextually vivid discussion notes.

4.4 Collating Feedback with SlideSpecs

With these design implications in mind, we developed **SlideSpecs**. SlideSpecs supports collating presentation feedback across three observed practice talk phases: the *presentation*, the post-presentation *discussion* and the feedback *review* (Figure 4.3). First, the SlideSpecs *presentation interface* records and contextualizes audience text critiques and facilitates audience collaboration with a lightweight tagging scheme and reply mechanism (Figure 4.2). The post-presentation *discussion interface* surfaces relevant text critiques to inform verbal discussions (Figure 4.4), and helps a facilitator to link discussion segments to related critiques (Figure 4.5). The *review interface* then lets presenters all collated feedback alongside the corresponding context (e.g., slides, discussion, topic) and a transcript of linked discussion segments (Figure 4.6).

Presentation Phase

SlideSpecs lets presenters upload a PDF of their presentation slides, and the system generates a thumbnail image for each slide in the presentation. To allow the audience to provide and organize text feedback (e.g., by author, by slide) during the presentation, SlideSpecs features a feedback-providing interface (Figure 4.2). The *Slides Pane* lets audience members view and select relevant slides for critique and the *Comments Pane* lets audience members provide their comments and view and interact with other audience comments.

Slides Pane

Audience members can view and attach feedback to presentation slides with the slide pane (Figure 4.2a, b, c). By hovering over a slide, audience members may view each slide in greater detail. Audience members can also select a set of slides to attach their comments to reference (Figure 4.2a, b, c). In this case, the audience member selects a range of slides (e.g., slides 7-11 & 42) for their feedback to reference (Figure 4.2b, c). If a comment references specific slides, the slide numbers will appear alongside the comment in the comment pane. By default, the slides pane displays the current slide, or the slide most recently matched to a presenter screenshot (Figure 4.2a).

Comment Pane

The comment pane lets the audience share their feedback within a comment field (Figure 4.2d), marked “add feedback here.” SlideSpecs detects the currently presented slide (details in Section 4.5), allowing the audience to contextualize their feedback automatically.

After sharing a comment, it appears in the comment list (Figure 4.2e) along with the author’s name (Molly), the time the comment was shared (12:44 pm), and any referenced slides (e.g. slides 68-73). To attribute explicit categories, the audience can include tags in the comment’s body (e.g., #slidedesign, #delivery, & #story). Audience members can interact with shared comments by writing a reply, agreeing with the comment, and flagging a comment for discussion. The comments pane also features a *focus* mode, which only shows self-authored comments.

Discussion Phase

After the presenter finishes their talk, the discussion interface supports records and links the group discussion through two views: the *participant view* and the *facilitator view*. The audience and the presenter use the *participant view*, which links to the *feedback providing interface*. The discussion view can be seen by the presenter and audience and shows all comments marked for discussion (Figure 4.4). A single audience member acts as the *facilitator*, who handles marking which comments as they are discussed. The facilitator view allows recording and transcribing discussion audio and marking new comments for discussion (Figure 4.5).

Discussion Participant View

To help audience members and the presenter select comments for discussion, SlideSpecs features a *participant view* that can be projected onto the presentation screen. The participant view contains the slides pane (as in Figure 4.2) to ease slide referencing and a list of comments marked for discussion by the audience. By default, comments are shown in a *To Discuss* list until the group addresses them. Audience members may mark comments to discuss by flagging the comment on their personal UI.

Discussion Facilitator View

SlideSpecs uses a human facilitator efficiently handle searching and marking discussion topics in the specialized and technical domains we studied. A purely algorithmic solution could potentially automatically cross-correlate verbal and written comments, though supporting jargon-dense technical research domains remains hard. The facilitator uses a unique view to record the discussion audio and to mark when specific audience comments are being discussed. The timing information captured about when comments were discussed can be leveraged later by the presenter in the review interface. Each comment status is linked to the participant view so that when a comment is marked as being “discussed” by the facilitator, each participant’s view also updates.

To quickly find relevant comments for discussion, the facilitator can leverage the list of comments flagged for discussion or create a new topic. The facilitator can also edit a topic’s

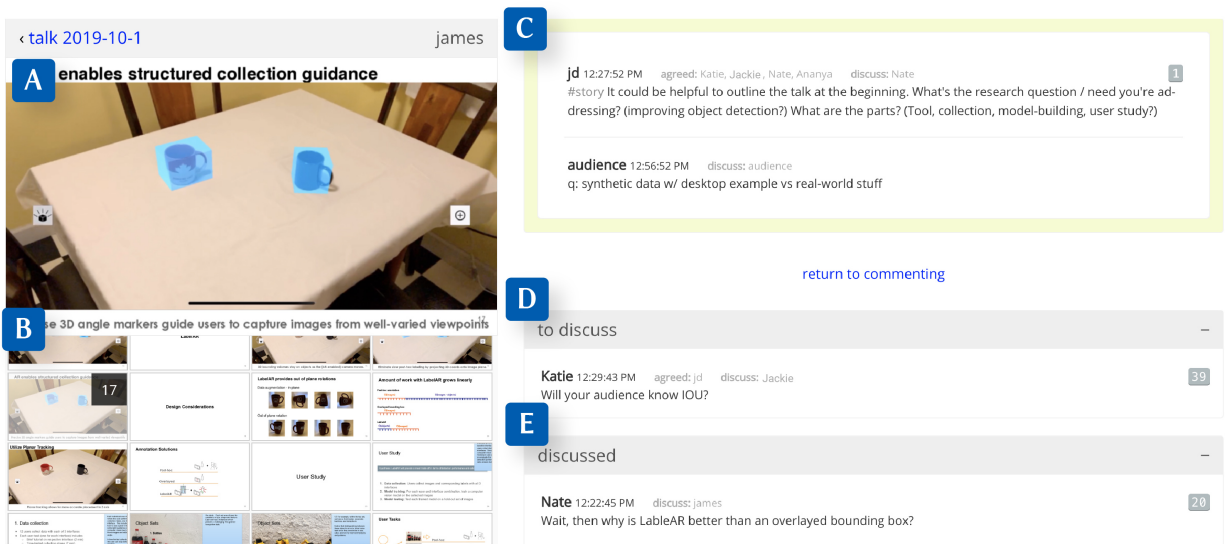


Figure 4.4: The *Discussion Participant* interface. (A) The slide pane detail view. (B) The listing of slide thumbnails, which also allows discussion participants to tag and filter comments by a specific slide. (C) Comments currently under discussion. These are synced with the comments shown in Fig. 4.5B. (D) Comments queued for discussion, initially populated by comments that the audience marked for discussion. (E) Comments already discussed. Once the facilitator marks these comments as discussed, they move into this section.

content to better match the discussion, given the facilitator will not often be sure what will be discussed ahead of time. The text input here doubles as a search box: when its text is updated, any comments with matching content or metadata are shown below the input as suggested topics for discussion. Multiple comments can be marked as being “discussed” at once, allowing for more leniency on the facilitator’s timing as discussions don’t always follow clear topic demarcations. These features serve to reduce the number of duplicate questions and topics, which can streamline the presenter’s review process. Existing AI-based NLP techniques could help alleviate the task burden placed on the human facilitator during this phase. This includes finding comments related to the ongoing spoken discussion and summarizing redundant comments. These exciting directions are all further discussed in our future work section (Sec. 7.2).

Review Phase

SlideSpecs includes a *reviewing interface* to let the presenter efficiently review feedback. The reviewing interface features (A) an audio player/waveform, (B) the slides pane, (C) the discussion transcript, and (D) a sorting/filtering pane coupled with the complete comments list and the list of each discussed comment. The presenter can use the transcript to index into part of the discussion: clicking on a word will start playing the audio file at that point.

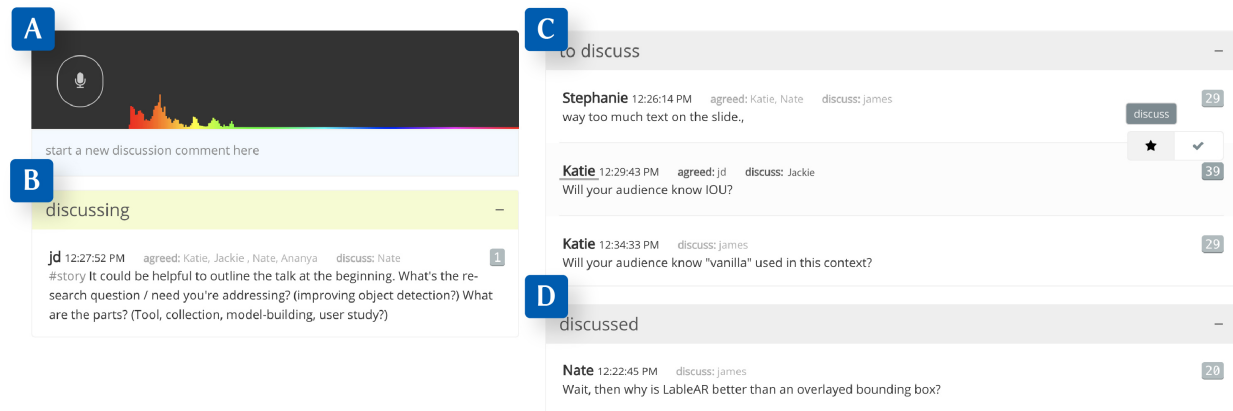


Figure 4.5: The *Discussion Facilitator* interface. (A) A microphone button to toggle audio capture and a visualizer for the audio input to verify that the discussion audio is active. The text entry below the audio controls can submit new discussion comments and works as a search tool to find relevant comments. (B) The “discussing” pane shows comments actively being discussed. These comments sync with the discussion participant view. (C) Comments queued up for discussion. (D) Previously discussed comments.

Additionally, every discussed comment can start playing the audio for the time range that it was discussed and features a trimmed version of the transcript for that time range. The discussion audio can be played back at 1x, 1.5x, or 2x speed. To let the presenter view the slides and tags that received the most comments, we overlay the number of comments on each slide in the slide pane. The interface displays the number of comments with each tag in the sorting and filtering pane (Figure 4.6).

4.5 Implementation

An overview of SlideSpecs’ architecture is shown in Figure 4.7, which highlights the SideSpecs data processing pipelines. We describe our slide-matching, web interface, facilitator search, and transcription implementations. The source code for this application is available online: <https://github.com/BerkeleyHCI/SlideSpecs>.

Slide Matching

To automatically provide relevant location hints for feedback, SlideSpecs predicts the currently active talk slide. We leverage color histogram information matching and optical character recognition for the presenter’s screen. While presentation software keeps track of the active slide, presenters used a wide range of presentation tools (observed in Section 4.2: PowerPoint, Keynote, PDF, HTML/JS). This variety makes authoring a uniform plugin to

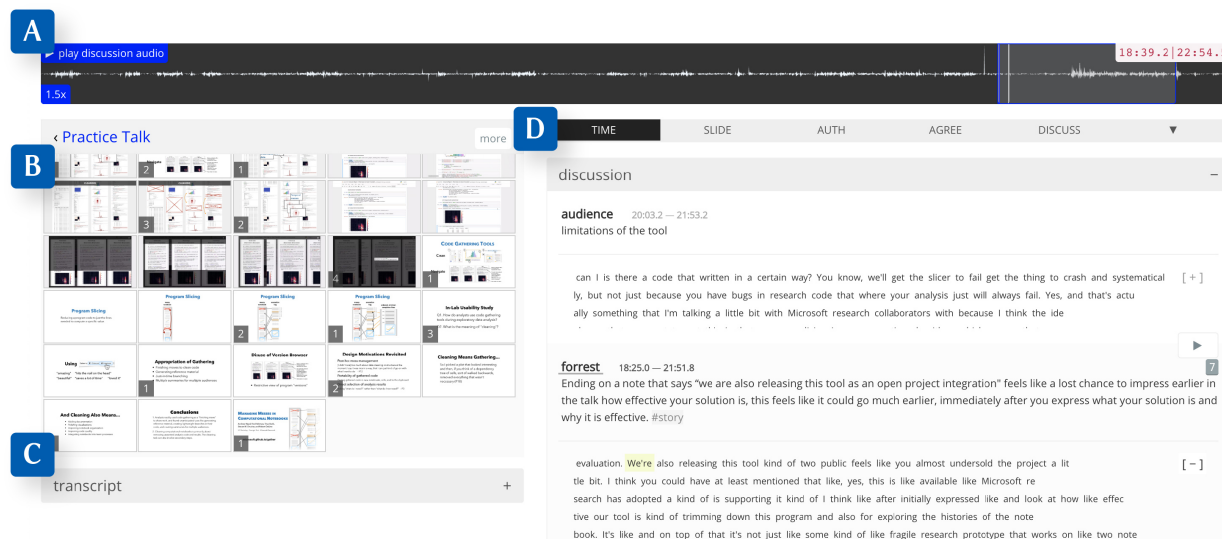


Figure 4.6: The *Review* interface. (A) The audio waveform from the verbal discussion is shown along with playback and speed controls. Users can scrub and jump directly on the waveform to navigate through the audio file. (B) The slides pane. Clicking on a slide allows filtering the visible comments to only those which refer to the selected slide. (C) The full transcript of the discussion (shown collapsed). (D) The listing of all comments and discussion topics. Users can hover over a referenced comment to highlight the time range in which the comment was discussed in the waveform and view the directly related transcript segment. This connection is learned from the facilitator marking specific comments as being discussed during the discussion phase.

monitor slide updates difficult. Instead, we use a custom technique for detecting the current slide based on presenter screenshots.

To enable slide-based references and automatic slide updates, the presenter uploads their presentation slides as a PDF to SlideSpecs. We automatically generate reference thumbnails for each slide from the presentation that serve as templates for matching. For each slide image, we precompute the included text using Tesseract’s OCR Engine¹. We also precompute a color histogram for each slide using OpenCV². This finds the *currently active* presentation slide.

Before their talk, the presenter downloads author-provided Python software that captures the projected screen image at 1-second intervals. For each screenshot, we first compute the color histogram of the image (how much of each color the image contains). Each presentation generally starts at the first slide and progresses linearly forward. Since the ordering of slides

¹<https://github.com/tesseract-ocr/tesseract>

²<https://opencv-tutorial.readthedocs.io/en/latest/histogram/histogram.html>

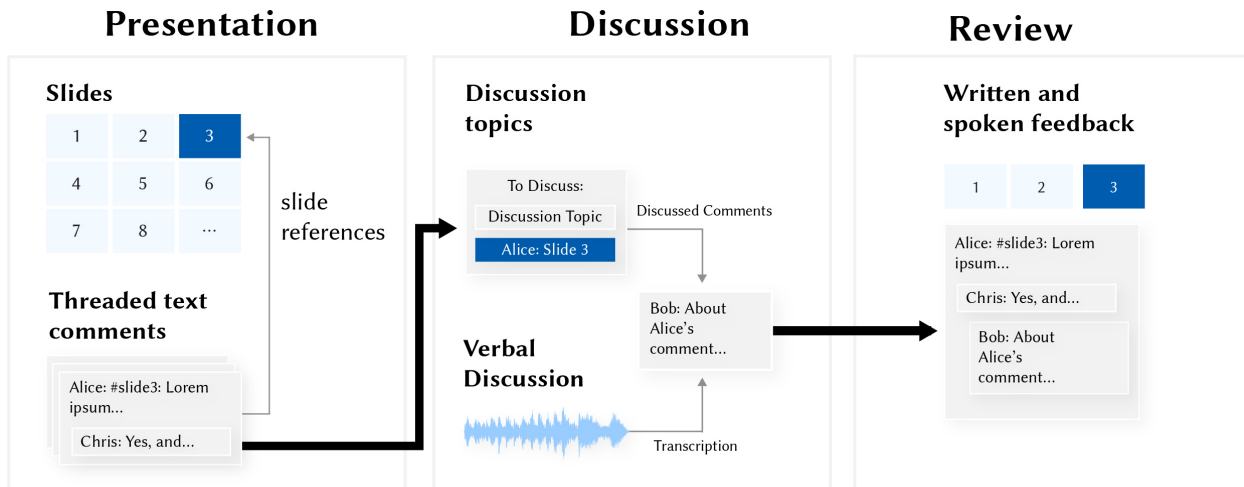


Figure 4.7: During the presentation, audience members write critiques and respond to existing critiques. Optionally, audience members can include context with their text critiques (e.g., specific slides, tags for feedback type). The threaded text comments flagged for discussion appear in the Discussion interface. During the discussion, the facilitator marks which topics are being discussed, matching existing text critiques where possible. SlideSpecs records and transcribes the discussion audio, allowing the presenter to review the collated text and verbal feedback together.

is known, we can constrain and refine our predictions. We compare the histograms using the spatial distance correlation along each color channel (RGB). If we have a confident histogram spatial match (e.g., less than a 0.01% difference) within a five-slide range, we immediately return that prediction. Still, some slides contain videos or have incremental builds/animations – visual complexity that will distort matches purely based on a color histogram.

If there is no confident histogram match in range, we recognize text within the screenshot (again with Tesseract³). We tokenize the extracted screenshot text with the spaCy NLP pipeline⁴. We compare common tokens for each slide in the expected five-slide range against their precomputed slide text tokens. We attempt to match the screenshot to one of these slides based on the highest shared tokens and return that slide as a prediction. If there is still no confident match, we search over the entire slide range (e.g., the presenter advanced quickly, rewound, or a previous prediction was incorrect). We then return the highest match over all slides, allowing the system to adapt to fast transitions or backtracking. This live prediction is streamed to each audience member’s interface, automatically updating the

³<https://github.com/tesseract-ocr/tesseract>

⁴https://spacy.io/models/en#en_core_web_md

inferred active slide. This prediction automatically labels audience-provided critiques with valuable context.

Web Interface

We used React.js to build each of the SlideSpecs interfaces. We deployed SlideSpecs with Meteor.js, enabling live updates across many clients in real-time. To access SlideSpecs, the presenter shares a unique SlideSpecs link with the audience. To guide the discussion, we enable users to flag comments they would like to discuss, which highlights comments later in the audience *Discussion* interface. The facilitator can also view and control these flagged comments.

Transcription

To unify written feedback entered during the talk and spoken feedback during the discussion phase, we employ speech recognition using Google’s Cloud⁵. SlideSpecs sends the recorded audio and retains individual speech tokens, timestamps, and confidence ratings. This transcript is shown in the final presenter *Review* interface.

Discussion Context

To further link from the *Presentation* and *Discussion* phases, our facilitator *Discussion* interface enables both recording discussion audio and marking comments as the audience discusses them. This context is recorded and appears later in the presenter *Review* interface. The facilitator comment box doubles as a search interface that can display written presentation comments matching author, content, and slide number. Each new keystroke instantly updates the list of related comments below the comment box, allowing the facilitator to tag relevant comments for context rapidly.

4.6 Effectiveness Study: Using SlideSpecs

To determine the effectiveness of SlideSpecs’ feedback collation, we deployed SlideSpecs in eight practice presentations. These presentations took place at two large research universities and were spread across four different research groups.

Method

We recruited presenters to give practice talks by sending emails via university mailing lists. For each talk, we first introduced SlideSpecs to the audience with a 5-minute tutorial. The list of presentation topics is shown in Table 4.1. During the talk, the audience used personal

⁵<https://cloud.google.com/speech-to-text>

Table 4.1: An overview of talk feedback collated with SlideSpecs. Audience members contributed 86% of text comments during *Presentation* and 14% of comments during *Discussion*. Here, N refers to the active group size – all who contributed at least one comment (mean: 9.5, s.d.: 3.8). Time refers to the minutes spent on each phase (mean: 17.0, s.d.: 3.5).

#	topic	N	<i>Presentation</i>		<i>Discussion</i>		<i>Comments with:</i>						
			slides	time	comments	time	comments	agree	discuss	slide ref.	reply	tag	
1	Haptic UIs	14	57	20	72	39	10	9	22	57	22	6	
2	Notebooks	9	33	18	32	23	10	10	13	30	13	5	
3	Tutorials	9	84	18	25	13	5	17	7	19	7	12	
4	Electronics	8	40	15	37	7	4	21	9	26	9	11	
5	Debugging	5	46	18	52	37	6	7	15	37	15	3	
6	Virtual Reality	16	83	22	115	35	11	25	36	95	36	8	
7	Computer Vision	10	41	14	38	15	12	11	16	33	16	6	
8	Sensemaking	5	32	11	18	19	14	6	14	14	3	10	
<i>average:</i>		9.5	52.0	17.0	48.6	23.5	9.0	<i>% total:</i>	23%	29%	68%	6%	13%

laptops or mobile devices to contribute written critiques with SlideSpecs. After the talk, the group verbally discussed feedback while a talk facilitator recorded the audio and marked comments for discussion. Facilitators were selected based on their relative seniority and familiarity with the presentation topic. Four unique facilitators (all senior Ph.D. Students) worked to both note the discussion topic and mark any relevant existing comments from the presentation during the discussion phase over these eight talks. We encouraged discussion participants (audience members) to discuss their feedback as they usually would. After the presentation and discussion, the presenter used SlideSpecs to review critiques. We logged all participant feedback and interaction with SlideSpecs.

After the talk, we sent an optional survey to each audience member with open-ended and Likert scale questions addressing likes, dislikes, and usefulness of system features. Each presenter and facilitator completed a similar survey on their experience with SlideSpecs. We also interviewed each presenter after they updated their presentations to learn if and how they used SlideSpecs to revise their talk. The audio of each presenter interview was recorded. We gathered more open-ended feedback on their experience and learned how SlideSpecs compared to previously used methods.

Findings

SlideSpecs effectively supported presenters by collating feedback into a single accessible and context-rich location. Presenters reported that using SlideSpecs improved feedback organization, provided valuable context, and reduced redundancy. When surveyed, 85% of presenters and responding audience members reported they’d use the tool again (Figure 4.9). In total, 52 unique audience members provided feedback over the eight presentations we deployed SlideSpecs. Of those, 14 audience members (A1-A14) reported on their experience in a voluntary survey (6/8 talks had at least one responding audience member). A1, A3, and

A7 mentioned using collaborative tools for feedback (Google Sheets/Docs) but preferred SlideSpecs. We report further on participants' experiences.

SlideSpecs gave presenters valuable organization. All presenters reported receiving useful feedback and that seeing the feedback organized by slides was useful (Figure 4.9). 5/8 presenters specifically rated organizing comments by authoring time that people provided critiques as useful for revision: it gave a **blow-by-blow** account of how people reacted during the presentation and discussion (P1). One presenter combined sorting techniques to prioritize feedback with the most consensus (e.g., agreement), then by the most discussed, and then by slide to form a *TODO* checklist.

SlideSpecs provided presenters valuable feedback context. Audience members used SlideSpecs to link slide references to 68% of critiques (313/461). They used tags (e.g., *#design*, *#story*, *#content*) to categorize their comments into subject types much less frequently (13%, 60/461). This difference could partially be because of the manual nature that comments had to be tagged. 8/14 reporting audience members explicitly mentioned liking being able to link slides to their critiques using slide images. Linking slides to images **saves typing**, lets audience members reference slides automatically when generating critiques, and frees audience members from recalling slide numbers. Presenters found the transcription and recording helpful when revisiting slides further from the original practice talk. 4/8 presenters reported reading or listening to the entire discussion section recording during their review. While most participants used either the transcript or recording to listen to segments of interest, P7 reported that in the future, they would use the transcript for efficiency but if there was ambiguity, even if just tone, I would play the audio to see what people actually meant. P3 described their process of supplementing listening with written comments: Took me 20 minutes to listen to it all. I went back-filled additional feedback onto the written comments. A lot of stuff came up in the discussion that normally I think I just would've forgotten.

Shared group awareness helps reduce redundant presentation feedback. Audience members collaboratively provided, on average, 48 written comments per talk, providing both local and global critiques. Audience members also interacted with others during critique: 23% of comments had at least one agreement (106 agreements over eight talks). 7/14 audience members reported liking seeing other audience members' critiques in real-time as it: **helps reduce redundant feedback (A2)** by providing an easy medium to share and agree (A10). Each agreement is potentially an instance of feedback that the presenter would have received multiple times.

Shared group awareness helps promote thematic feedback consensus. Several presenters reported noticing more thematic consistency in the verbal discussion that followed the practice talk after using SlideSpecs. P1 reported: During the verbal feedback session after my talk, people spoke not only about their individual comments but also about themes they saw emerging on SlideSpecs. So, I felt like I got more coherent verbal feedback that covered all the major issues in

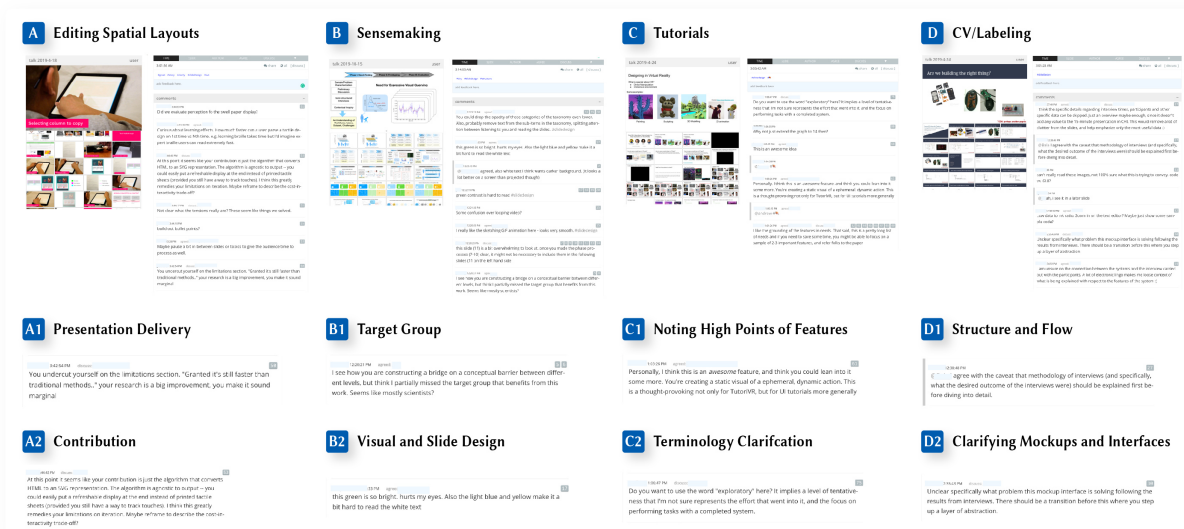


Figure 4.8: Participants used SlideSpecs for a wide variety of practice talks (A-D). They received feedback concerning many aspects of their presentation, from slide design, delivery, and requests for clarification. We highlight select feedback from four different practice presentations. These comments showcase topics that the audience’s feedback addressed; eight comments (A1-D2) are shown in detail alongside a rough characterization of what part of the presentation the comment addresses.

a pretty coherent way. A8 reported: During the discussion, another audience member said, ‘Here are the themes I saw from the comments’ – and this feedback was so useful. The unified thread of discussed comments helped the audience merge multiple related comments into higher-level themes during the discussion.

The Facilitator’s tasks are demanding. While the presenters benefited from the discussion organization, the facilitators generally found the role to be demanding. For example, F1 reported that it could be unclear which comments are being discussed: Since I could choose to make a new topic or mark existing ones for discussion, I sometimes stumbled between just marking a new topic quickly and trying to find the most relevant comment to bring up for discussion. F3 also reported on this challenge: Facilitating was challenging, as I was trying to both follow the conversation and think of what would be relevant for the presenter later on. Even though I used the comment search and author filtering, deciding the true “most relevant” comment was ambiguous, along with deciding when a comment was no longer being discussed. While the context that the facilitator provided to the discussion was valuable, the role’s tasks were complex and would benefit from automation support (e.g., text summarization, identifying relevant comments, and marking new topics).

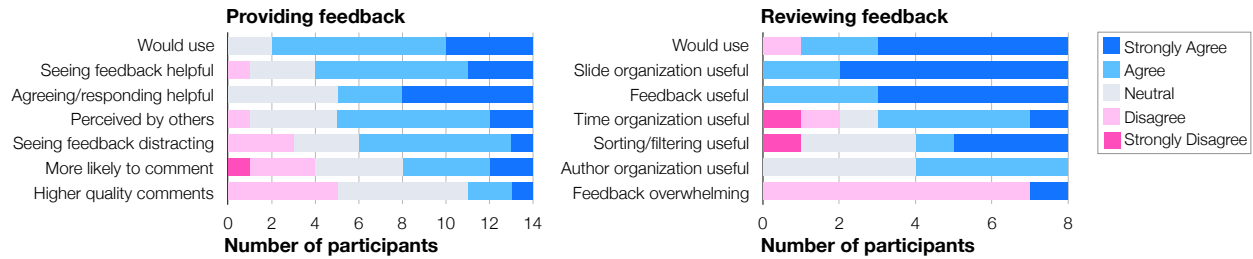


Figure 4.9: Likert scale responses for 14 audience members (left) and eight presenters (right) in the group deployment. All except one presenter and two audience members agreed that they would use the interface in the future (85%). Likert scale responses reflect that audience members broadly find seeing the real-time of others to be very helpful (10/14) though somewhat distracting (6/14). Audience members noted seeing and agreeing/responding to feedback was helpful. All 8/8 presenters favored slide organization during the feedback review and found the feedback collation useful.

Study Limitations

Participant Usage Period.

Most presenter participants used our tool only once; thus, our findings have limitations common to first-use studies. We have limited information about how tool use might change over time and how it may impact the nature and flow of group discussions and practices.

Domain and Talk Format.

Our sampling of presentations was limited to only including STEM fields. SlideSpecs likely doesn't support processes or talks that avoid slides or using computers (e.g., only verbal discussions).

In-Person Presentations Only.

We only studied in-person practice talks in this study. However, remote and hybrid talks are increasingly common, and SlideSpecs may serve presenters in that context differently.

Audience Survey Response Rate.

Despite having 52 unique audience participants in our practice talk sessions, only 14 participants sourced from 6/8 talks responded to our optional survey (27%). While we likely captured the strongest and most polarized perspectives, we missed out on reporting a broader set of user experiences.

4.7 Discussion

Through building and evaluating SlideSpecs, we gained a thorough understanding of what a presentation feedback tool should support. We reflect on our performance through several design implications for feedback collation and review.

Peer awareness reduces redundancy.

Our results demonstrated that incorporating peer awareness mechanisms (e.g., seeing or interacting with other audience feedback) reduced redundancy in the collected feedback. For instance, audience members interacted with others during the critique by expressing agreement (e.g., liking comments, commenting in a thread to elaborate on another member’s feedback), providing an easy mechanism for prioritizing feedback and reducing repeated statements.

Context helps presenters revise effectively.

SlideSpecs captures many forms of context automatically, which helps presenters revise their presentations more effectively. This included *scope*: what parts or slides the feedback is referring to, and *subject*: what category or type of feedback is being given. For instance, presenters may leverage the flexibility of SlideSpecs to sort the slides to achieve different revision goals: e.g., prioritize slides that require the most changes, fix comments with the most consensus, or revise the comments participants discussed first. This range of contextual data supports how presenters may weigh feedback. Given the value of feedback context, comments could also be automatically clustered around a tag or theme.

Feedback consists of more than context.

Another implicit assumption is that the audience can already provide valuable and relevant feedback. SlideSpecs makes providing feedback with enriching context simple, though it cannot make up for being unfamiliar with the talk domain or not knowing how to give effective feedback. While the shared context can inform more novice audience members about how to give feedback, it doesn’t inherently instruct the audience *what* feedback to provide the presenter. While our contributing audience members varied from undergrads to professors, we did not measure feedback quality against expertise. A follow-up study could compare these qualities; participants in our formative study also reported implicitly weighting feedback with the provider’s role (e.g., the lab PI vs. an undergraduate research assistant).

No single tool is always best.

A key design implication from our formative study was *supporting existing workflows for group feedback*, so we developed SlideSpecs to support a commonly observed (yet fairly linear) feedback process. SlideSpecs is designed to support existing practices that center around receiving in-person feedback. Still, no single tool can accommodate the entire range of feedback processes that different groups use. For example, SlideSpecs requires a deck of slides for the audience to comment on, and some talks forgo slides. Another assumption is that presenters will follow a linear pattern of Presentation, Discussion, and Review (Fig. 4.3). However, especially in longer talks, presenters may want to pause between each section to gather feedback incrementally. Another more volatile process might involve stopping anytime an audience member has a comment or question. To support other feedback processes, a different approach could be used where audio is constantly recorded/transcribed, regardless of the current feedback phase.

The eight presentations in Table 1 cover both a broad range of subject matter and a range of structures, including practicing for conference talks, grant meeting progress updates, and academic job talks. Despite this variation, many other types of talk structures exist that we did not evaluate (e.g., educational lessons, startup pitch decks). However, no evidence suggests that varying the internal structure of the talk makes SlideSpecs less relevant or valuable. Despite only brief training, the groups we worked with adopted SlideSpecs into many existing practice processes. On top of this, the presented talks in our study had a mix of intended purposes (conference, research update, job talk).

Hybrid in-person/online presentations also present unique challenges. Inviting online participants offers compelling benefits – allowing geographically remote participants, built-in recording features (e.g., Zoom), and more easily scaling the audience size. SlideSpecs could be deployed in these contexts as-is, though it would not be leveraging this new contextual data (e.g., multiple audio feeds for discussion, video/screen recordings for presentations). Still, the mixed social dynamic of hybrid meetings presents new challenges that SlideSpecs is not explicitly designed to support.

Acknowledgements

This research project required the presenters and research groups that were open to trying out SlideSpecs – thanks to all who participated and provided feedback with SlideSpecs. We also extend thanks to Pancham Yadav who assisted with the initial literature review. Thanks to James Smith for creating the SlideSpecs project video figure.

Chapter 5

Vector Style Transfer

Vector graphics are an industry-standard way to represent and share a broad range of designs. Designers frequently find inspiration and explore styles from existing designs. However, updating the overall visual styles of entire existing designs is tedious, limiting exploration opportunities. We present VST (Vector Style Transfer), a novel system for flexibly transferring visual styles between vector graphics by interacting with automatic graphics correspondence and styling algorithms. In VST, designers can tune generated correspondences and filter which attributes to transfer per correspondence. We report results from a user study in which designers used VST to control style transfer between several designs, including designs they created beforehand. VST shows that this correspondence tuning and customization can enable interactive, flexible style transfer. We also find that someone using our system can significantly reduce transfer time and work compared to an experienced designer using industry-standard tools. A version of VST for styling pre-matched design pair examples is available at: <https://berkeleyhci.github.io/vst/>.

5.1 Introduction

Vector graphics are an industry-standard way to represent and share a broad range of designs. As a design medium, vector graphics offer compelling advantages, including scalability and precision. Vector graphic designs store information about each graphical element that they contain. This information enables editing the design at a higher level of semantics when compared to pixels. Many vector graphics design tools have achieved success supporting designers working in this medium (e.g., Adobe Illustrator, Figma, Canva, Sketch).

This work presented in this chapter was first published in Warner et al. as *Interactive Flexible Style Transfer for Vector Graphics* in the 2023 Proceedings of UIST, the ACM Symposium on User Interface Software and Technology [152].

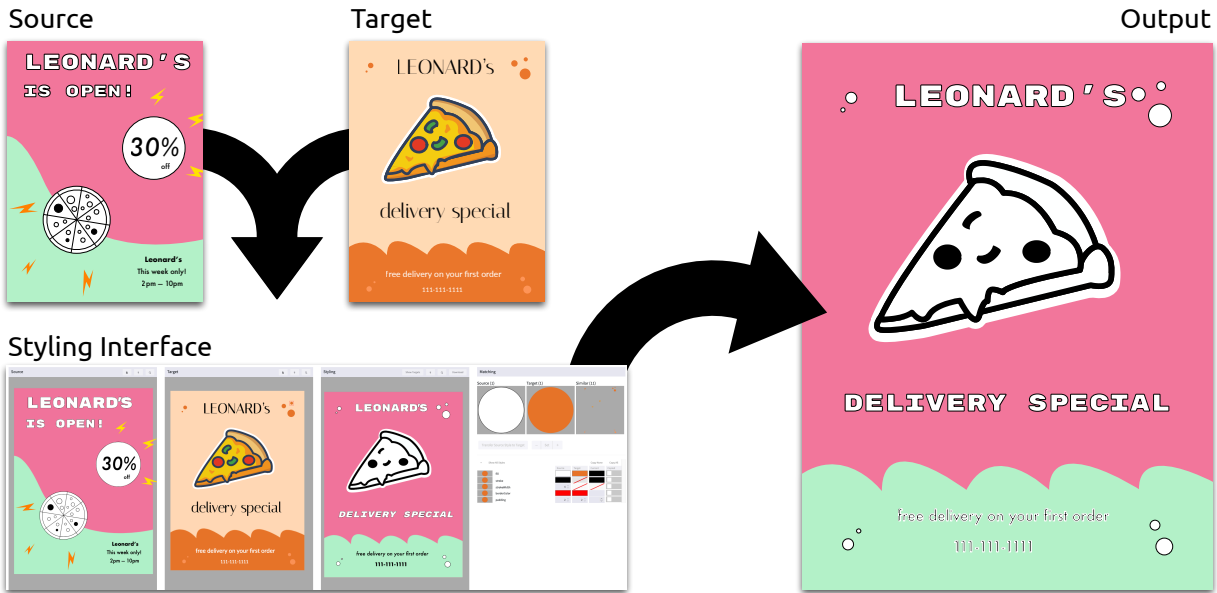


Figure 5.1: VST generates new output graphics by transferring visual styles from source graphics onto target graphics. The styling interface enables filtering which styles to transfer, filtering which elements to stylize, and previewing all or part of the new stylized output graphics. The output graphics retain a similar structure to the original target while bearing visual styles from the source graphics.

Designers often edit vector graphics' overall appearance or style while retaining their underlying content and structure. In this work, when we write *style*, we refer to the defining visual properties of a design's elements (e.g., color, shape, size, and font). Many alternative and valid definitions of this broad term exist. Style editing tasks arise in multiple situations, such as applying inspirations from a mood board, updating existing graphics to a new visual identity, or exploring multiple alternative style variations. For example, both a novice designer seeking to apply styles from a more polished design to their work and an experienced designer creating several variations of a similar design to present to a client for feedback face this task. This complex task requires many selection and editing operations for different groups of objects. Updating a design to conform to a new visual style can be exceptionally tedious and limits the exploration of different styles, even for experienced designers.

One potential solution is to use document-level themes or rules that consistently apply visual attributes to classes of objects. This approach is standard across many design and presentation software tools. For example, web pages use CSS (Cascading Style Sheets) to enable document-level styling, but these style-content links must be manually created and maintained. A notable downside of using document themes or stylesheets is their *rigidity*. Compelling themes require element class information and pre-planning, introducing *viscos-*

ity [42] into the authoring process. Despite CSS support in SVG [150] via the `<use>` tag [98], most vector graphics avoid it.

Another promising direction is to *automatically* transfer visual styles between graphics using information on how two given designs relate to each other. However, this approach often fails to transfer styles as each designer uniquely intends. This failure stems from two sources: 1) the accuracy limitations of the algorithm and 2) the inherent subjectivity around *good* style and varying tastes that designers may have. A fully automated approach may transfer styles in undesired or unpredictable ways. The lack of adequate designer controls is a clear barrier to leveraging automation [127].

A tool should enable rapid iterating on different possible style transfer results to address the shortcomings of a fully automatic style transfer approach. Our research aims to combine the benefits of automation with effective controls for customizing and exploring design variations. Our approach combines automatically generated design correspondences with interactive control of how and where to transfer styles. We leverage prior work [136] on generating an automatic correspondence between vector graphics. This method yields a between-design element correspondence (Fig. 5.2) and element-wise similarity along multiple dimensions.

We present a new design tool, VST, short for *Vector Style Transfer*. VST provides designers with an interface to visualize and customize how style flows across designs (Fig. 5.3). VST displays a dynamic list of element styles, allowing designers to easily copy, reset, and customize element style attributes (see Appendix 5.3 for all attributes). With VST, designers can map and remap example SOURCE element styles onto contextually similar elements. VST also features fast and flexible ways to identify, select, and style TARGET elements. The OUTPUT canvas re-renders the stylized TARGET graphics in real-time with any changes, providing immediate visual feedback.

Conceptually, VST expands the *eyedropper* or element-wise *style copy-paste* interactions to groups of elements. VST can infer many element relations directly, omitting the need for explicit element structure or class information. Our combined automation-powered interactive style transfer approach means that designers can get the best of both worlds – their style definitions can both be based on ad-hoc demonstrations and quick to apply flexibly across designs.

To evaluate VST’s style transfer capability, we recruited six designers to transfer styles between nine designs. Each designer participating in the study successfully used VST to interactively transfer styles to their satisfaction and make nine new OUTPUT designs. In a follow-up design replication study, we recruited four expert designers to each manually replicate six of these OUTPUT designs in their preferred design tool. The results from this

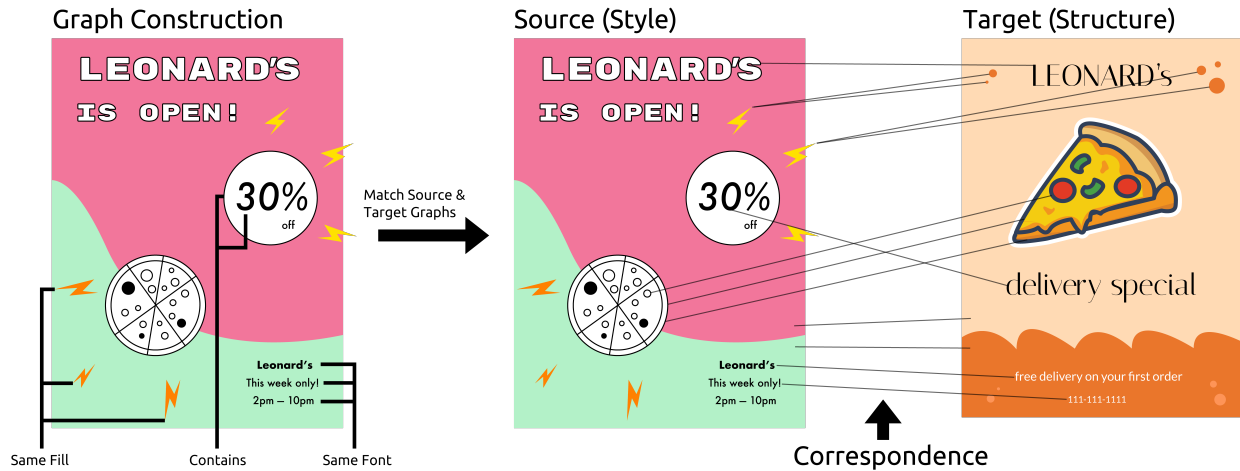


Figure 5.2: An overview of automated design correspondence. To relate design elements, we first construct a graph from each given design, where the *vertices* are primitive design elements (e.g., shapes, text, images) and *edges* are semantic relationships (e.g., same fill, containment, same font). Once the SOURCE and TARGET graphs are constructed, we then compute a correspondence between the two designs’ elements using the technique previously detailed in [136]. This automatically generated correspondence is VST’s basis for (a) how to find similar elements *within* a design (e.g., for easier selection/styling) and (b) identifying which elements are similar to each other *across* designs (e.g., determining which initial styles to transfer). Each TARGET element is linked to a single SOURCE element. Only a subset of links between these designs’ elements are shown.

preliminary study suggest that someone using VST may reduce the time and work for this style transfer task compared to experienced designers using industry-standard tools.

Our contributions include the following:

1. VST, a design tool that introduces a novel user interface for interactive, user-guided, flexible style transfer for vector graphics. Its key interaction principles are: a) enabling users to edit computed correspondences at multiple levels, and b) enabling users to customize how attributes are transferred between designs across the correspondence.
2. Two user studies that demonstrate: a) that designers can successfully transfer styles between graphics with VST, and b) that designers without VST can spend more time and effort to produce equivalent design results.

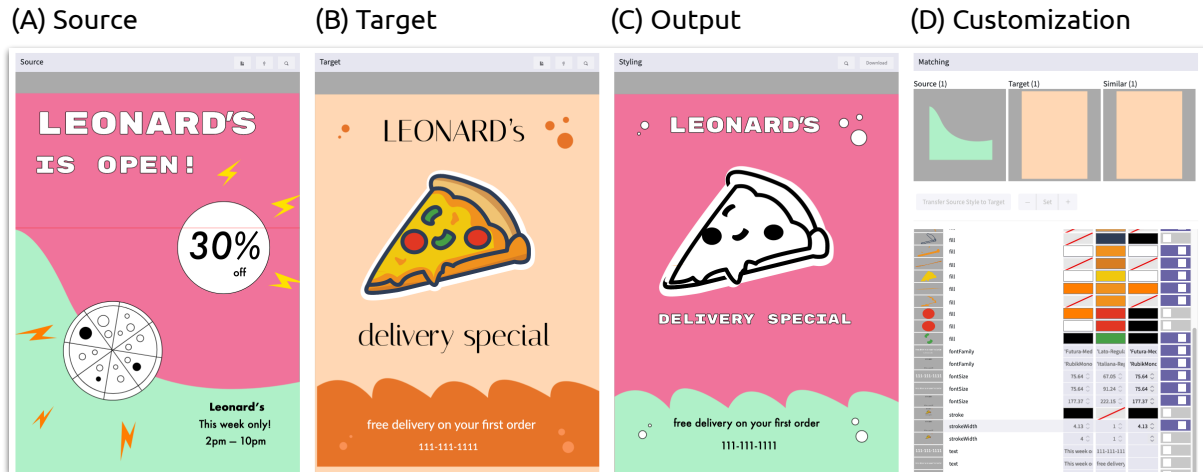


Figure 5.3: An overview of the VST interface, including (A) the SOURCE graphics (where the style is sourced from), (B) the TARGET graphics, (C) the OUTPUT canvas (the current style transfer result), and (D) customization controls for matching element styles across canvases and filtering which style attributes to copy or modify. Designers can filter this list of attributes (shown in D) based on the current selection to do more focused editing or instead modify shared style attributes across the entire design.

5.2 Vector Style Transfer

Task Characterization

When transferring styles between vector graphics, designers may identify an inspirational style they want to copy from a SOURCE design. Next, in a TARGET design, they may identify design elements they would like to stylize. Then, they will update the stylistic attributes of those relevant TARGET elements using the SOURCE style as a reference. Alternatively, they may first focus on the TARGET design they wish to change and pull stylistic influences in from a range of SOURCES, exploring possible variations. Generally, this styling is an iterative and flexible process that involves reasoning about (a) *which elements correspond to each other across designs* and (b) *which style attributes to transfer*. There is subjectivity regarding the most desired application of style, and higher-level considerations like the overall cohesion of the TARGET design after styles have transferred further complicate this task. The resulting OUTPUT design has the *style* of one design and the *content/structure* of another – though this distinction is still inherently subjective. Still, this task (using examples to update existing graphics with new visual styles) is expected in the graphic design process [81, 50, 69].

Design Goals

A high-quality element correspondence is one way to enable fast and effective style transfer for vector graphics designs. To provide designers with flexible control over style transfer is to provide them with tools to control the correspondence between designs. Moreover, to be worthwhile, the resulting designs should be of satisfying quality and faster to generate than existing tools, especially when considering the cost of learning to use a new tool. Grounded in our literature review and personal experience editing graphics, we created these design goals for Vector Style Transfer (VST):

DG1 Let designers powerfully tune design correspondences.

DG2 Enable flexible control over which styles are transferred.

DG3 Reduce the work and time needed for transferring styles.

Our vision for how the functionality of VST best fits into existing processes is as a plugin or new tool in existing vector graphics design software. Designers could select an object group and copy their style. Then, they could select any other group within their design document and apply that style – without manually selecting each element subset. Additionally, they could filter which styling attributes they would like to copy. This work could either be used as a starting point to render a design in several alternative styles or to make a set of designs adhere to a single style.

Exemplar Scenario

We will demonstrate VST’s functionality with an exemplar scenario involving vector style transfer. Consider Xavier, a designer hired by a local Italian restaurant, *Leonard’s*. After a recent renovation, the restaurant is set to have a grand re-opening. Xavier has created a new flyer to help them advertise, which the business manager approves. To unify the brand’s style, the business manager also asks him to create new versions of several existing graphics, including menus and a special delivery advertisement. These designs should look like they all refer to the same restaurant.

This style unification process Xavier faces involves many repeated manual edits and cross-references. Instead of manually ensuring exact visual consistency, he opens VST and loads in both graphics (SOURCE: the new flyer, TARGET: the previous advertisement). VST computes a correspondence between elements of these two designs and automatically copies styles between matches. This correspondence technique ensures a one-to-many mapping from the SOURCE elements to the TARGET elements. This ensures that every TARGET element will be matched, while some SOURCE elements may not be initially matched. Xavier then sees the OUTPUT canvas update with newly stylized graphics (Fig. 5.3).

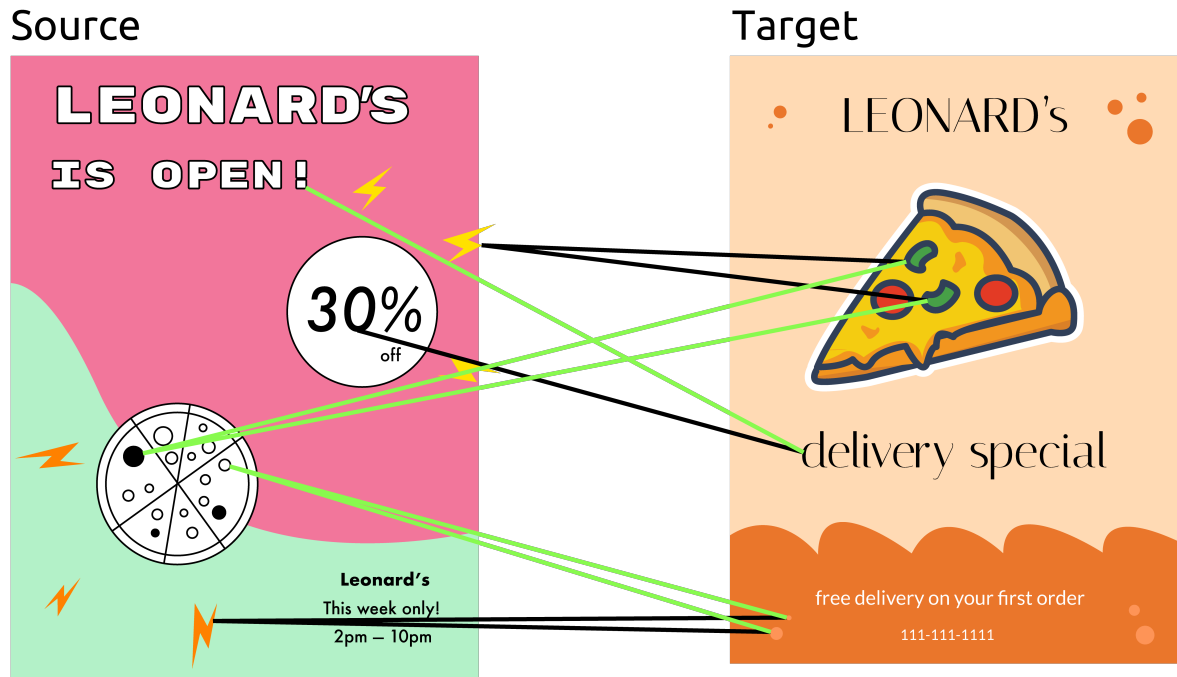


Figure 5.4: The black lines show an initial correspondence between the elements of the SOURCE and TARGET designs. The green lines show an alternative, more desired set of links. When users select their desired SOURCE and TARGET elements and press *Transfer Source Style*, VST will update these links, redirecting the flow of visual styles across designs.

For each TARGET element, styles are copied from the most similar SOURCE element as determined by the design correspondence algorithm [136]. In addition to seeing the updated target graphics, a list of changed style attributes is displayed on the right-hand side of the interface (Fig. 5.3D). The breadth of style attributes and the range of possible valid matches between elements makes using a fully automatic approach difficult. The inherent subjectivity of style also means this first attempt will not always be correct, especially for more complex and open-ended designs.

Xavier immediately detects outlier text elements that are visually misaligned with the SOURCE style directly on the OUTPUT canvas (Fig. 5.3). Designers are trained to use gestalt principles of perception to organize a design. Incorrect style transfer will lead to visual violations of these principles, which are often easy to detect [100]. This means that some elements likely have been ‘mismatched’ by the correspondence algorithm (Fig. 5.4). Using the SOURCE canvas (Fig. 5.3A), Xavier can then specify which SOURCE element the incorrectly styled text fields should visually match. When he presses the *Transfer Source Style to Target* button (Fig. 5.5), VST renders styles from the SOURCE element onto the TARGET selection in the rightmost OUTPUT canvas (Fig. 5.3C). Behind the scenes, VST

applies these fixes to a copy of the original correspondence, avoiding recomputing the entire correspondence after updates.

Still, manually selecting each target element to update is tedious. To enable faster transfer, designers can double-click on any TARGET element to select similar elements, as determined by the design correspondence. Repeatedly double-clicking an element iteratively grows the set of selected TARGET elements. This feature mirrors the multi-click selection in other media, like toggling between word-sentence-paragraph selections within a text document. Here, we use the underlying within-document element-wise similarity score to intelligently add elements most similar to the currently active selection. A similarity score is computed for each element relative to the currently selected elements, and the elements with the highest score is added to the active selection. An aggregate similarity score is computed for each unselected element relative to the currently selected elements, and the element with the highest score is added to the active selection. If multiple elements share near-equal scores, they will all be selected, making selecting instances of the same object or pattern efficient. In addition to double-clicking, the customization pane (Fig. 5.3D and Fig. 5.4) features precision similarity controls. This interface shows both the active TARGET and SOURCE selection and previews the sequence of next-larger TARGET selections (without having to actually adjust the active selection). Shift-clicking toggles the selection of any given element. Double-clicking on a SOURCE element conversely selects all TARGET elements currently matched to that element, which shows how style flows from the SOURCE to TARGET design. The customization panel shows a pane of similar elements, where Xavier can preview this selection (Fig. 5.5).

Despite Xavier updating the SOURCE-TARGET correspondence, the resulting OUTPUT design still has some problems. For example, while the font and color are corrected, the copied font size makes some elements not fit neatly in the new design (Fig. 5.5). Once matched, VST has controls for customizing which specific style attributes are transferred. To focus on the desired element, he clicks *Show Filtered Style* to only see the styling applied to the text element (Fig. 5.5). He toggles the `fontSize` attribute, resetting that element's font size and updating the OUTPUT canvas. Similar attribute values are grouped in this view to make selecting and editing easier. He continues this style transfer process until he is satisfied with the quality of the new design. Internally, these changes build up a list of attribute transformations to apply to the TARGET design. The customization pane can highlight just the modified attributes, summarizing stylistic changes at a glance. Finally, Xavier downloads the OUTPUT graphics from VST as an SVG file to save his work.

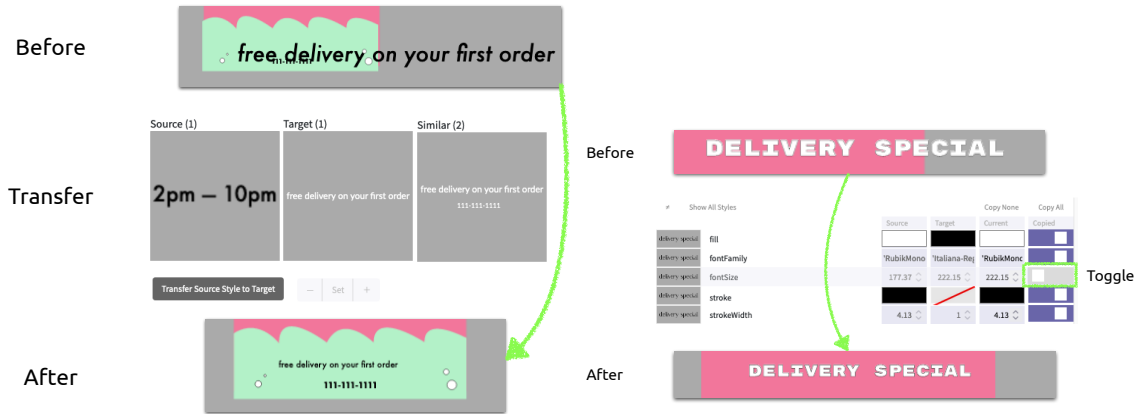


Figure 5.5: The Customization UI shows the SOURCE and TARGET selections and similar TARGET elements. The similarity controls [-/set/+] can adjust the selection to the desired TARGET elements. Once satisfied with the SOURCE-TARGET mapping, pressing *Transfer Source Style* will transfer all styles from the SOURCE elements to the active TARGET selection. The Customization UI also provides fine-grained control over which styles to transfer. Element style attributes can be copied, reset, or customized for each set of similar values. This list can be filtered only to show styles for the current selection and only to show modified attributes. The UI also features the *Copy All* and *Copy None* buttons – *Copy All* blindly copies all styles for every matched element (e.g., the fully automatic output), and *Copy None* restores the OUTPUT graphics to the original TARGET state.

5.3 Implementation

A styling-only version of VST is available at: <https://berkeleyhci.github.io/vst/>. We used ReactJS¹ to build the VST interface and deployed our prototype online. Vector graphics are rendered using FabricJS, a vector graphics library leveraging the HTML5 canvas backbone. SVG files, such as those exported from industry-standard design tools like Sketch, Figma, Canva, and Adobe Illustrator, can be directly imported.

Once VST has imported the input SOURCE and TARGET graphics, we compute a correspondence between the two designs using a comparison technique introduced by Shin et al. [136]. This technique represents each design as a multigraph (rather than a typical parent-child hierarchy tree) to support matching elements across a broader range of similar attributes. Vertices are primitive design elements (e.g., shapes, text, images), and edges represent semantic relationships between elements (e.g., alignment, containment, same fill). This correspondence contains per-element similarity scores across several dimensions (e.g., color, shape, size, and text).

¹<https://react.dev/>

In our implementation, correspondences between 20 or fewer elements are generally computed in real-time (< 1s). Though slower, our study’s larger design pairs are still tractable to match, with the largest pair (185 total elements) taking about 100s. Our example set’s average matching time per design pair (across Style Transfer Tasks 1 and 2) is 7.78s. Once obtained, match information can be exported and saved for later use. The source code for this app is also available at: <https://github.com/BerkeleyHCI/vst>.

Transferrable SVG Attributes

The SVG attributes that VST can transfer are: **(Color Based)** fill, stroke, strokeWidth, textBackgroundColor, **(Text Based)** lineHeight, textAlign, text (i.e., string content), **(Font Based)** fontSize, fontFamily, fontStyle, fontWeight, and **(General)** opacity, padding.

5.4 Evaluations

Style preferences are subjective, so making any absolute statements about a style transfer tool’s *performance* is difficult. Still, we sought to evaluate three key research questions:

RQ1 How would designers use VST for style transfer?

RQ2 Could VST stylize realistic, open-ended designs?

RQ3 Could VST reduce the time or work of styling?

Style Transfer Evaluation

Method

To answer RQ1 and RQ2, we ran an exploratory study with six experienced designers (D1-6). Before the study began, we asked designers to create a new design from a given prompt with their preferred design tool. The prompt requested a single menu page design for a local restaurant’s (*Leonard’s*) mobile phone application. The goal was to include designer-provided source graphics to create a more realistic style transfer scenario.

Four designers used Figma to generate their initial designs they brought into the study, while the other two designers used Adobe Illustrator. After designers responded to the prompt, we hosted an hour-long Zoom session with each designer. We instrumented the interface to log relevant events with timestamps (e.g., loading, saving, editing). We sought to gather rich commentary and reflection from designers as they engaged with the prototype. We invited designers to verbally share any thoughts on their experience and highlight any surprising interactions throughout the study. While we recorded usage times per example, designers were not told this nor instructed to be as efficient as possible.

The designers moved on only after indicating satisfaction with the relative appearance of their stylized OUTPUT graphics. Finally, designers answered a brief survey about their experience using VST, including Likert-scale (Fig. 5.9) and open-ended questions. Designers sent all styled designs and an interface usage log to the authors and received a \$30 Amazon gift card for their participation.

Participants

We recruited designers via design-oriented email lists at a large research university. Designers included undergraduates (4), Ph.D. students (1), and design professionals (1). Each participating student had completed multiple design internships, bolstering their relevant experience. Their preferred tools included Figma, Adobe Illustrator, and Canva. They had an average of 4.7 years of design experience (2–10 years).

Task 1: Basic Graphics Pairs

After an interface demo and the opportunity to ask questions, designers used VST to transfer styles between five pairs of example designs that the authors prepared. The design pairs we chose for designers to transfer from are shown in Fig. 5.6 (T1.1-5). We chose these graphics to capture a breadth of different graphic design domains (e.g., art, infographics, UI mockups). We instructed designers to apply styles from the SOURCE to the TARGET graphics to make the SOURCE and OUTPUT as stylistically similar as possible. Once satisfied, they would save the OUTPUT graphics and move on to the next pair.

Designers D1-6 used VST to transfer styles from the SOURCE to the TARGET graphics. Both SOURCE and TARGET designs were provided to the designers (see Fig. 5.6). In simpler cases, the design transfer result is uniform across designers (T1.1-3). Still, despite each designer starting from the same pair of designs, variations arose in more complex designs (T1.4-5).

Task 2: Open-Ended Transfer

To observe how VST handled styling more open-ended realistic designs (RQ2), designers transferred styles from their externally created designs onto three new related templates (T2.1-3). In these tasks, the SOURCE was a menu page created by each designer before the study with their preferred design tool. We matched their designs to three new template pages (a loading screen, a reviews page, and a checkout cart), all for *Leonard's* mobile app. The generated output design correspondences (Fig. 5.2) were not hand-tuned at all before the study. Again, designers were instructed to use VST to make the SOURCE and OUTPUT stylistically similar in each task. Once satisfied, they would save the OUTPUT graphics and continue.





























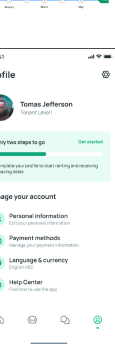


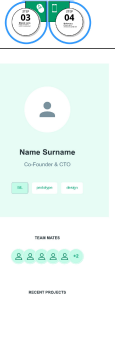



	Given	D1	D2	D3	D4	D5	D6
T1.1							
T1.2							
T1.3							
T1.4							
T1.5							
	Source (Style)	Target (Structure)	Designer Created (With VST)				Designer Created (With VST)

Figure 5.6: Task 1 (Style Transfer) – Basic Graphics Pairs.

Before the study, we gave designers (D1-6) a prompt for a menu design with specific elements without any style instructions. The column header in Fig. 5.7 shows designs that they brought into the study (SOURCES), and the row header shows design templates (TARGETS). The inner table shows new designs created by applying styles from their externally created SOURCE design onto previously unseen TARGET templates. Inspecting each column shows a unified visual style inherited from the SOURCE document, while rows show the TARGET structure.

Style Transfer Results

Our style transfer evaluation study found that designers could use VST to control style transfer across basic designs (RQ1), even generating variety in their OUTPUT designs from the same inputs. Those designers successfully used VST to flexibly transfer styles from more realistic, open-ended designs created with external tools (RQ2). We take this as an indication that VST enabled the style transfer it was designed to support. Each designer participating in the study (D1-6) used VST to generate eight new OUTPUT designs successfully.

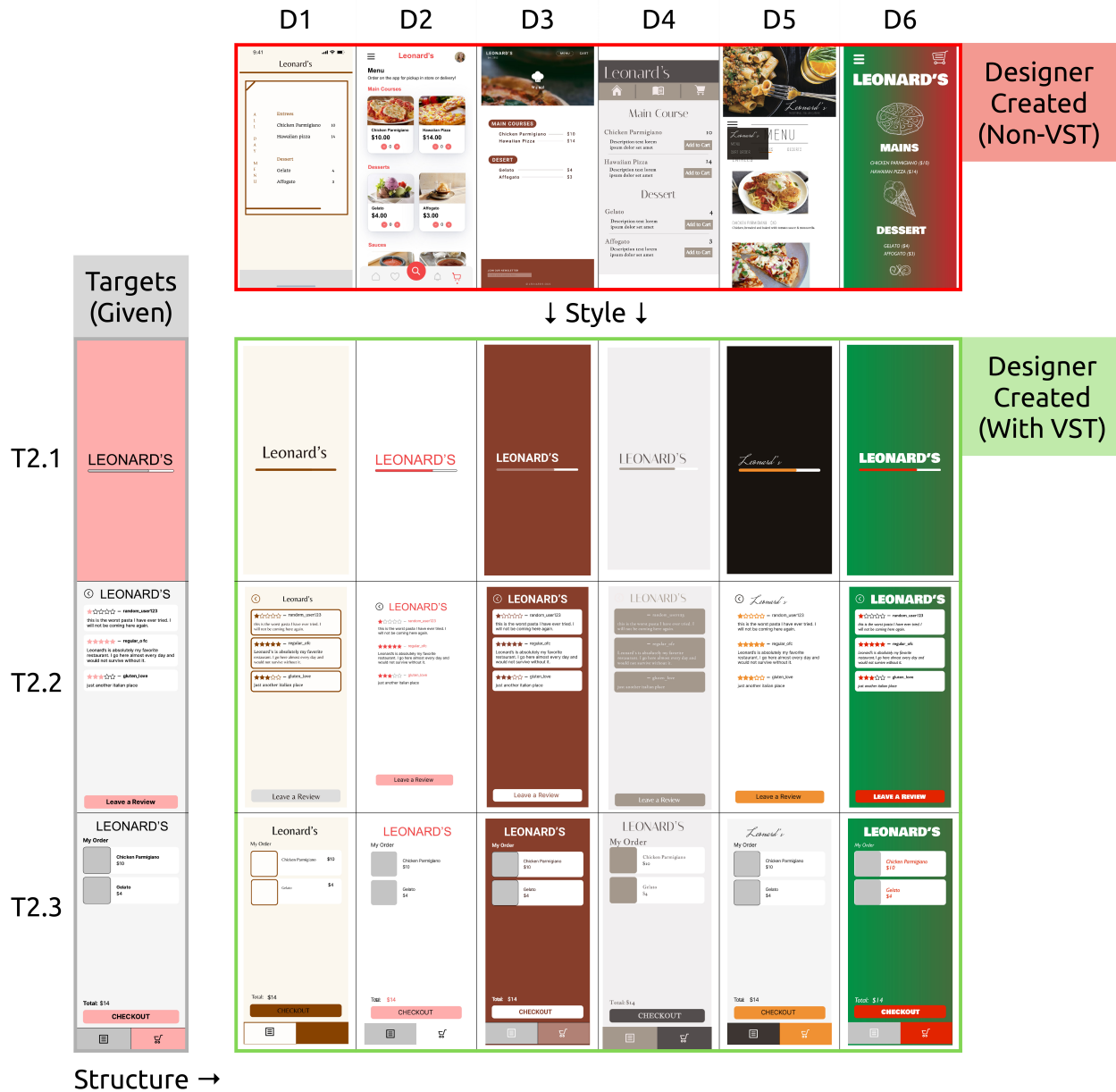


Figure 5.7: Task 2 (Style Transfer) – Open-Ended Transfer.

Designers also answered Likert-scale questions regarding their experience with VST (Fig. 5.9). Style transfer examples from the evaluation are shown in Figures 5.6 and 5.7.

Designers, despite never using a similar interface before, used VST’s features to both (a) modify design correspondences (DG1) and (b) filter and edit styles per correspondence (DG2). Software instrumentation revealed that almost all designers on almost all tasks used

VST to tune computed correspondence matches. On average, designers performed 6 such corrections per task. While making these corrections, designers used the functionality to *select similar* elements to the ones they manually selected. On average, designers performed 7.3 similarity selections and spent about 4.8 minutes per task. As a reminder, designers were only instructed to match the styles to the best of their ability – not to do so as quickly or efficiently as possible. We showcase additional, more complex VST graphics made outside of this study in the Appendix (Fig. 5.11) and in our paper’s accompanying project video.

VST let designers tune design correspondences (DG1). Overall, designers appreciated the style transfer control that VST provided them. The designers’ Likert-scale responses indicated they could produce designs they were satisfied with (Fig. 5.9). Most designers could see themselves using the tool again and found VST flexible enough to perform style transfer as they intended. Their verbal remarks are corroborated by the frequency with which they used the correspondence correction feature (Average: μ : 6.0, Standard Deviation: σ = 3.8) and attribute editing feature (μ : 24.0, σ = 17.3).

VST enabled flexible control of style transfer (DG2). The designers created a wide variety of designs, even when given the same input graphics (Fig. 5.6). For their own provided graphics, designers reproduced a consistent theme across a set of provided vector graphics templates (Fig. 5.7). Several designers remarked on the convenience of reusing visual styles directly. D4: Very fun! Appealing to a visual thinker who values efficiency and hates repeatedly doing the same things. Magical, “it read my mind!” kind of feeling. While most found it clear how to use the different parts of the prototype to achieve their desired style transfer, there was also feedback that the transfer results were sometimes surprising. This surprise likely stemmed from having multiple ways to style elements (e.g., tuning the correspondence vs. what styles the correspondence transfers).

Designers enjoyed applying broad changes. Designers valued the ability to apply broad style changes quickly. D3: I was impressed by how well the system generated its “best guess” when I selected the “Copy All.” I also thought it was easy to learn and intuitive. It had tools that worked similarly to design software I already used (like dragging values to change the font size). D5: I liked how efficient the transferring process was in closely replicating the desired style with just a button. Even if it wasn’t completely accurate, the toggle buttons under Copy All made fine-tuning specific aspects of design elements easy – I could definitely see how this interface could reduce the amount of time that a designer would need to update designs. The interface was organized and easy to navigate around. Designers also appreciated directly selecting similar elements easily, which helped broader styling. D4: Being able to select multiple elements precisely is very nice.

Correspondence-based transfer presents novel controls. No designer reported using a similar style transfer design tool before this study. D6: I have not used anything that performed this exact function before, but I’ve used a tool to try to analyze an image and find out what fonts were used. It was not as reliable as this tool. While most designers (4/6) indicated an

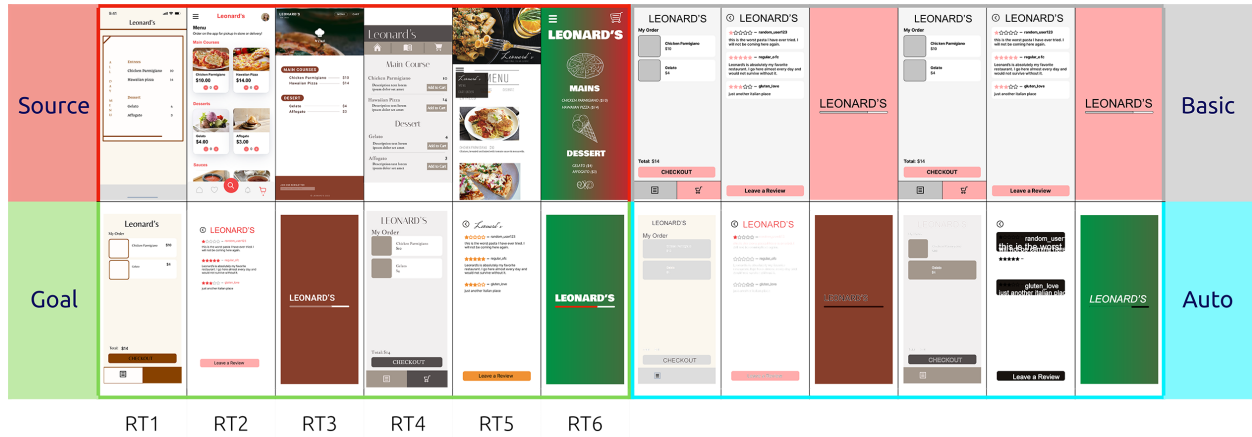


Figure 5.8: Design Replication Task – A new set of expert designers (replication designers RD1-4) replicated six reference designs (RT1-6) from the previous style transfer evaluation tasks (Fig. 5.7) using two different starting points: *Basic* and *Auto*. The first approach involved using Illustrator to transform the *Basic* input design to the replication goal. The second approach again used Illustrator, but instead has the algorithmic output (*Auto*) as the starting point. We provided the RDs with source styles and target structures from the previous study in vector form and a reference image of the replication goal for both approaches.

interest in using the tool again, others were hesitant, citing VST’s deviation from the types of tools they were familiar with. Some designers recognized the value of a style transfer tool: D4: I have manually copied styles and have had other humans manually copy my own. When successful, this tool manages to give you that feeling of empathy and creative connection (“Wow, the other designer understood my aesthetic and was able to replicate it! I feel they really understand my vision”). When it is not successful, it is easier and less stressful to correct than a human might be. When a human is unsuccessful, you might say, “This other designer tried to copy my font style, but they didn’t get it right; what a pain, I hope they don’t mind me changing it myself...” Plus, it is faster than asking another designer, fewer resources, less risk, and when it is successful, high reward!

Design Replication Evaluation

Method

To answer RQ3, we ran a follow-up study. Our goal with this study was to compare the time and work required for style transfer in VST with that of an expert using industry-standard design software. We recruited four new expert designers as replication designers (RD1-4). They were tasked with recreating a subset of the OUTPUT graphics from the previous study (T2.1-3) in their preferred design tool (Adobe Illustrator).

We conducted this study remotely over Zoom in a 3-hour session for each designer. Unlike the style transfer evaluation, the RDs were asked to work as swiftly and efficiently as possible. Once the RDs reported they were satisfied with the similarity between their replication graphics and the reference OUTPUT image, they would save their file and move on to the following example. While the RDs participated in our study, we recorded their screen, an audio log of the call, application edit history, and mouse activity. From this data, we recorded the number of selections (including selection adjustments like shift-clicking or clicking the background to clear the selection) and attribute edits (per selection—so, for example, modifying the fill of a group counts as one edit). We also recorded the time spent on each example task, measured from when all input files were opened to the last save of the output file. Finally, the RDs were shown, briefly used VST, and filled out a survey based on their experiences. The RDs received a variable Amazon gift card. The amount was prorated based on their required completion time (rated at \$30/hour).

Given that VST is a novel design tool there are no users with equivalent VST expertise comparable to the RDs' Illustrator skill. To approximate the performance of an expert VST user, the authors used VST to generate the same OUTPUT designs using the same input materials provided to the RDs. This data is labeled VST in Table 5.1. We report the comparison between these three design replication methods in our results.

Participants

We recruited from the same design community as before, now selecting only the most experienced designers. All RDs had professionally worked as designers. One was the instructor for a university course teaching students how to use Illustrator, and another held a residence in a design lab guiding student projects. These designers had, on average, 6.5 years of design experience and used Illustrator daily. None of these expert designers participated in the original study.

Task: Design Replication

We selected six OUTPUT design examples from Fig. 5.7 for this designer to replicate in Illustrator (*Goal* in Fig. 5.8). We selected designs to include both graphics from every task (T2.1-3) that we gave the original designers and to include one example per designer (D1-6). We provided the RDs with the SOURCE and TARGET vector graphics files and an image of the generated OUTPUT (created initially by D1-6). The RDs were then tasked with transforming the TARGET graphics to resemble the provided OUTPUT.

To measure what human adjustment is needed when working with the automatically stylized designs, we also asked the RDs to replicate the OUTPUT starting with the initial automatically stylized OUTPUT graphics from VST. These graphics (*Auto*) are created by copying all styles using the initial automatic SOURCE and TARGET correspondence. We asked the RDs to transform the now-partially stylized TARGET graphics to resemble the

OUTPUT image. Any difference between these two sets (*Basic* and *Auto*) would highlight the algorithm’s impact on the task time and work. To compare the potential of VST and existing tools, the authors also replicated the same OUTPUT designs from the previous study using VST (RT1-6). The same input materials were used as in the Illustrator replication: the SOURCE and TARGET vector graphics files and an OUTPUT image.

Design Replication Results

In our study, using VST to transfer styles was faster than expert replication designers (RD1-4) transferring styles within their preferred design tool (RQ3). The RDs also performed more edit and selection operations using Illustrator than the authors using VST. We report total work as a combination of selection and edit operations. On average, the RDs spent 534 seconds replicating from scratch (*Basic*) and 774 seconds replicating from the output of the correspondence algorithm (*Auto*). In comparison, the authors required, on average, 129 seconds to match styles using VST. A plot of the duration for each task is shown in Fig. 5.10. Stats averaged over all tasks (RT1-6) are shown in Table 5.1. Each replication designer also reported the style replication task as difficult and tedious.

Transferring styles with existing tools is tedious. After replicating the designs in Fig. 5.8 (RT1-6), the RDs reported on their experience by answering Likert-scale (ranging from 1-7) and open-ended survey questions. They reported that using Illustrator for this style matching task is tedious for both starting points, with *Auto* slightly more tedious than *Basic* (Average (μ): 6.8 \rightarrow 5.8, Standard Deviation: $\sigma_{Basic} = 1.3$, $\sigma_{Auto} = 0.5$). The associated scale labels were: 1-*Not tedious at all* and 7-*Extremely tedious*. They also reported starting from *Auto* was less fun than *Basic* (μ : 2.0 \rightarrow 3.8), with 1-*Not fun at all* and 7-*Extremely fun* ($\sigma_{Basic} = 1.0$, $\sigma_{Auto} = 0.8$).

Editing from Auto was not faster than Basic. Combining automated style transfer with existing design software tools may even hinder designer performance. The RDs reached roughly the same Likert-scale level of satisfaction with their final designs’ quality from both the *Basic* and *Auto* starting points ($\mu_{Basic} = 4.3$, $\mu_{Auto} = 4.5$), with 1-*Completely dissatisfied* and 7-*Completely satisfied* ($\sigma_{Basic} = 1.0$, $\sigma_{Auto} = 1.0$). However, they reported that generating the desired OUTPUT was harder with *Auto* than *Basic* (μ : 6.3 \rightarrow 5.0), with 1-*Not difficult at all* and 7-*Extremely difficult* ($\sigma_{Basic} = 1.0$, $\sigma_{Auto} = 0.8$). These stats match their written feedback: RD1: Editing the auto files is harder – there’s more variance in the output, and sometimes unnecessary properties were added from the automatic transfer. RD2: In the standard [Basic] file, editing elements is more straightforward, while for the modified [Auto] one, I spent some extra time cleaning. RD4: I largely had a similar approach to both design files, though the original [Basic] one tended to be easier.

Replication designers wanted transfer tools like VST. After briefly interacting with VST at the end of the study, all RDs were genuinely interested in trying out an Adobe

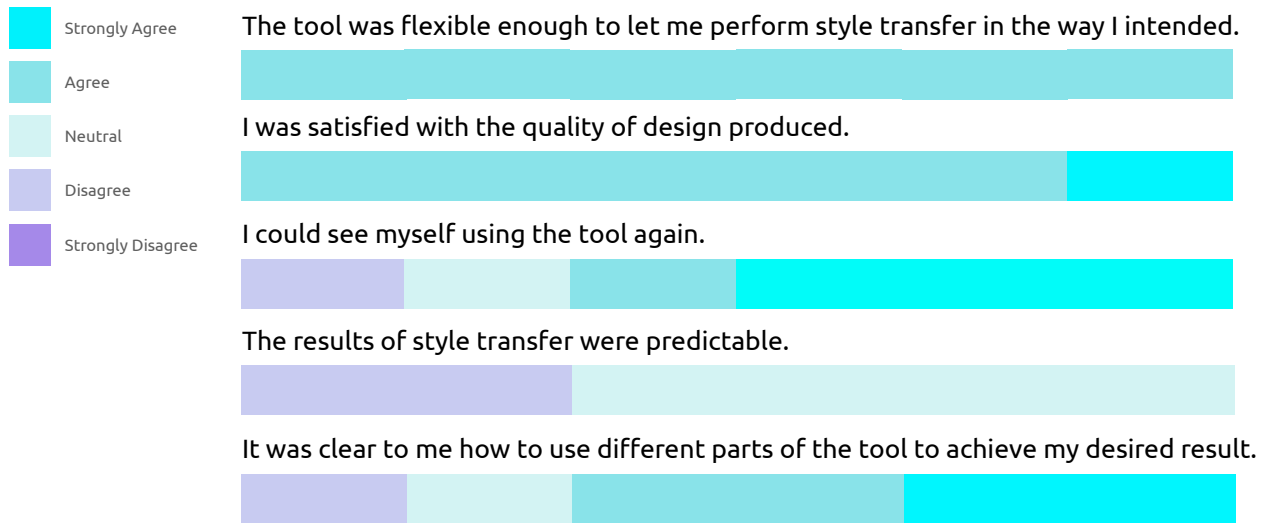


Figure 5.9: Summary of Likert survey data from designers D1-6.

Illustrator plugin with similar functionality ($\mu = 6.25, \sigma = 1.0$), with 1-*Not at all interested* and 7-*Extremely interested*. RD4: The prototype looks very interesting! RD1: I would definitely try it when I want to apply vector-based styles to my design. When asked about if and where they would find VST useful: RD1: I can see how this tool would be beneficial for tasks like redesigning an existing UI or early-stage exploration. When asked about other similar tools they have used: RD2: In Figma, we save the font/color as a library preset, then when we change the setting, it automatically updates the components. RD3: The style transfer prototype is more adaptive than design components because files that I need to change may not have a component system.

We also performed a one-way ANOVA test to compare the effect of the replication method on **work operations** (e.g., the number of selections and edits). This test revealed a statistically significant difference in task duration between at least two groups ($F = 7.7, p < 0.01$). A post-hoc Tukey’s HSD test for multiple comparisons found significant differences between the mean operations of *Basic* and *VST* ($p < 0.05$) and between the mean operations of *Auto* and *VST* ($p < 0.01$). Again, we found no statistically significant difference between the *Basic* and *Auto* design replication methods ($p = 0.33$). We observed similar significant differences when independently evaluating edits ($F = 5.7$) and selection ($F = 7.2$) operations.

We performed a one-way ANOVA test to compare the effect of the replication method on **task duration**. This test revealed a statistically significant difference in task duration between at least two groups ($F = 10.1, p < 0.01$). A post-hoc Tukey’s HSD test for multiple comparisons found significant differences between the mean times of *Basic* and *VST* ($p < 0.05$) and between the mean times of *Auto* and *VST* ($p < 0.01$). We found no statistically significant difference between the mean task duration values for the *Basic* and *Auto* design replication methods ($p = 0.17$).

A repeated measures ANOVA was also performed to compare the effect of replication method on **work operations**. There was a statistically significant difference in total work between at least two groups ($F = 13.72$, $p = 0.001$). We again performed three paired t-tests between our three evaluated replication methods. We found no significant difference between the *Basic* and *Auto* methods ($t(5) = 2.47$, $p = 0.056$, Cohen’s $d=0.7$). Between *Basic* and VST, we found a significant effect for techniques ($t(5) = 4.52$, $p = 0.006$, Cohen’s $d=0.8$) with VST outperforming *Basic*. Between *Auto* and VST, we found a significant effect for techniques ($t(5) = 3.90$, $p = 0.011$, Cohen’s $d=2.6$) with VST outperforming *Auto*.

5.5 Discussion

Design Recommendations

The success of VST demonstrates the value of two key design goals that are relevant as recommendations for other automation-powered correspondence-based transfer tools: include the ability to flexibly *tune generated design correspondences* (DG1) and include the ability to flexibly *customize what correspondences do* (DG2).

Tuning Generated Design Correspondences. Providing powerful and convenient ways to tune correspondences avoids requiring users to make each mapping manually (DG1). In VST, this functionality is represented by our *Selecting Similar* feature, the ability to view and select elements sharing any of the same values in the customization pane, and the *Similarity Threshold* feature (which lets users quickly preview selections).

Customizing Correspondence Functions. Customizing a correspondence retains the flexibility of a manual approach, ensuring that designers still have control (DG2). The domain will ultimately specify what is reasonable to transfer per correspondence. Generally, the designer should be able to control what happens when two objects are linked. In VST, we achieve this through our customization panel, where designers can copy, reset, and customize attribute values. We also provide flexible ways to filter this list (e.g., by active selection and showing modified/all attributes).

The Cost of Automation

One notable point in our results is that starting with the algorithm’s output (*Auto*) did not make replication easier. In fact, the RDs reported that starting with the automatically generated algorithm output was more difficult and less fun. Simply throwing automation into existing tools and processes may backfire. This is backed by our quantitative results: the *Auto* designs, on average, required more work than the corresponding *Basic* starting point (*Basic*: 265 operations, *Auto*: 383 operations). This is jarring, as applying the style transfer algorithm should have the opposite effect — otherwise, why apply it at all?

Table 5.1: Replication work data – usage statistics averaged over replication tasks RT1-6 (see Fig. 5.8). The *Basic* and *Auto* columns show aggregate data collected from the four expert replication designers (RD1-4), while the *VST* column shows data from the paper authors using VST to replicate designs.

		Basic	Auto	VST
Task Duration	Mean	532	774	129
	S.D.	341	347	80
Work Operations	Mean	265.7	383.5	30.3
	S.D.	167.8	159.2	18.9
Attribute Edits	Mean	80.0	113.1	13.0
	S.D.	59.9	77.8	8.7
Selection Updates	Mean	185.7	270.4	17.3
	S.D.	122.8	185.7	12.1

First, applying a semi-correct transformation reduces cohesion in the design. The lack of cohesion commonly found in *Auto* designs reduces the efficiency of applying gestalt principles. This makes selecting similar elements to style them together harder. Second, the vast scope of the copied attributes may introduce new work. Incorrectly changing an attribute does not create new work if it already needs to be changed. However, if part of a SOURCE style is not desired in the OUTPUT graphics, those attributes must be manually reset to their original TARGET value. Current design software fails to support this type of style transfer interaction.

In contrast, VST features convenient ways to quickly select and explore element styles (double-clicking an element/selection, precision selection controls, visually selecting via the same attribute value). Current correspondence algorithms do not seem to reduce the total work in style transfer otherwise. This is especially true for more complex examples where correspondence accuracy is often lower.

Object Groups As Stamps

We introduce the concept of *object groups as stamps*: using a group of graphical objects to dynamically apply styles to new content. One designer reported on the creative potential of this concept. D4: A [design] button where I just recorded the style [...], and then maybe I want to apply it, not just to other buttons but to a bunch of other things—it kind of feels like this is like my stamp, my magic stamp, and wherever I stamp it, it will transform that into that style. I’ve turned these elements into a tool, and when I think about it like that, I’m not just trying to copy this; I can use this to change the design creatively and maybe do some things that weren’t as expected. The initial expectation is I’m trying to copy this, but then another expectation is maybe I can make this design even better, and now I can make it more consistent.

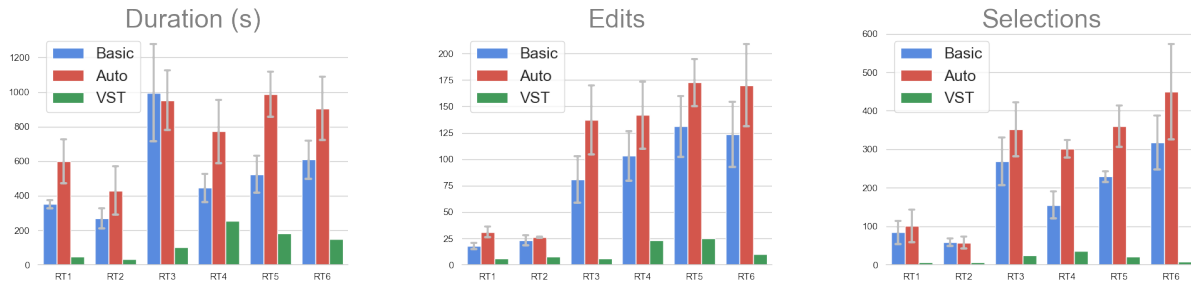


Figure 5.10: Plots of the duration, edits, and selections data from the design replication (RT1-6). Along each recorded measure (duration, edits, and selections), the authors using VST outperformed all four expert designers using Adobe Illustrator in replicating the stylized designs. **Duration:** seconds per task. **Edits:** Number of edit operations per task. **Selections:** Number of selection updates per task. The *Basic* and *Auto* plots also include ticks showing the *standard error* for each task computed over RD1-4. VST was only used once per task to obtain a baseline, so there are no comparable ticks to show.

Existing element style copy-paste mechanisms only use a single element source. Using the same metaphor, this *stamp* is limited to a single value for each attribute. Other even simpler tools (e.g., eyedropper color picker) simply copy a single attribute from a single source. These techniques are much more limited in the types of styles that can transfer. These limitations raise the required work and time from the designer compared to object groups as stamps. Implementing this *object groups as stamps* concept requires a design correspondence. Otherwise, which TARGET elements should receive which SOURCE styles is unclear. Further, multiple SOURCE elements may have different values for the same attribute. Without a mapping between the two sets of design elements, the transfer is reduced to element-style copy-paste. Using this required design correspondence introduces both new flexibility and potential errors. Being able to transfer a larger set of styles is obviously desirable (exploring more design variations, saving time). The strength of a tool like VST lies in the control that it provides designers when tuning these correspondences. This control is matched to the many ways a designer may want to transfer styles between two designs.

Limitations

VST is not a general-use vector graphics editing platform. The SVG standard is complex; even industry-standard platforms like Inkscape and Adobe Illustrator may render the same graphics differently. Still, some missing features limited how useful VST was for designers in its current state. Users wanted more advanced layering/z-reordering for sub-selections in complex design areas. Additionally, the current correspondence structure usage limits elements to inheriting styles from one SOURCE element unless manually mixed with other styles.

We also did not measure the impact of algorithm matching performance on this task. Informally, study participants D1-6 updated the correspondence an average of six times per task, though our study instrumentation did not record the number of adjusted elements per update. In Shin’s prior work [136], the average match accuracy was 95% (ranging from 78–100%). However, their evaluation [136] was performed with the SOURCE as an element group within a TARGET design, rather than a separate design. Explicitly varying the match quality and leveraging different matching techniques are opportunities for future work. Another limitation of this work is the smaller scale of the surveyed designer population (10 unique designers across both studies). For our design replication study, we worked with four expert designers. While this smaller study size allowed us to deepen the level of feedback and data we gathered, future studies could evaluate a larger expert population to get additional feedback. Future work could conduct a larger-scale study with more designers to potentially collect insights into a broader set of behaviors that designers exhibit. Also, when comparing VST to other tools, the authors have more awareness of the replication goal and task, which likely improves their relative performance. Another evaluation could train experienced designers with VST and have them replicate graphics from the original study.

The performance of our correspondence construction also inherently limits the scope and complexity of design that we can tractably operate on. Using a different correspondence structure that can parse more complex graphics in real-time could open exciting new possibilities, such as operating on an entire set of designs simultaneously rather than working per design pair.

Vertically stacked and overlapping objects present a challenge for the basic selection techniques that our prototype features. While we retain z-index information to correctly render parsed graphics, we don’t feature full-fledged layer support found in most other popular design tools. This makes selecting objects stacked below another object somewhat tedious — users can drag over the elements to select both, and then shift-click to deselect the top element.

We do provide other selection methods, like the thumbnails in the style attribute list (Fig. 5.5). Still, to handle more complex designs more advanced selection techniques are needed. Designers mentioned the lack of a unified global *Undo* function made exploring styles feel riskier and limited exploration. We avoided adding this feature in VST because of the multiple potential sources of change and the high memory cost of storing previous canvas states in the browser. One possible solution is to have edit histories for both the correspondences and for the values in the styling pane to further encourage the exploration of new styles. While we can render images embedded into an SVG, the prototype cannot currently transfer image fills between objects. With that said, none of the designers in our study explicitly mentioned this as a hindrance.

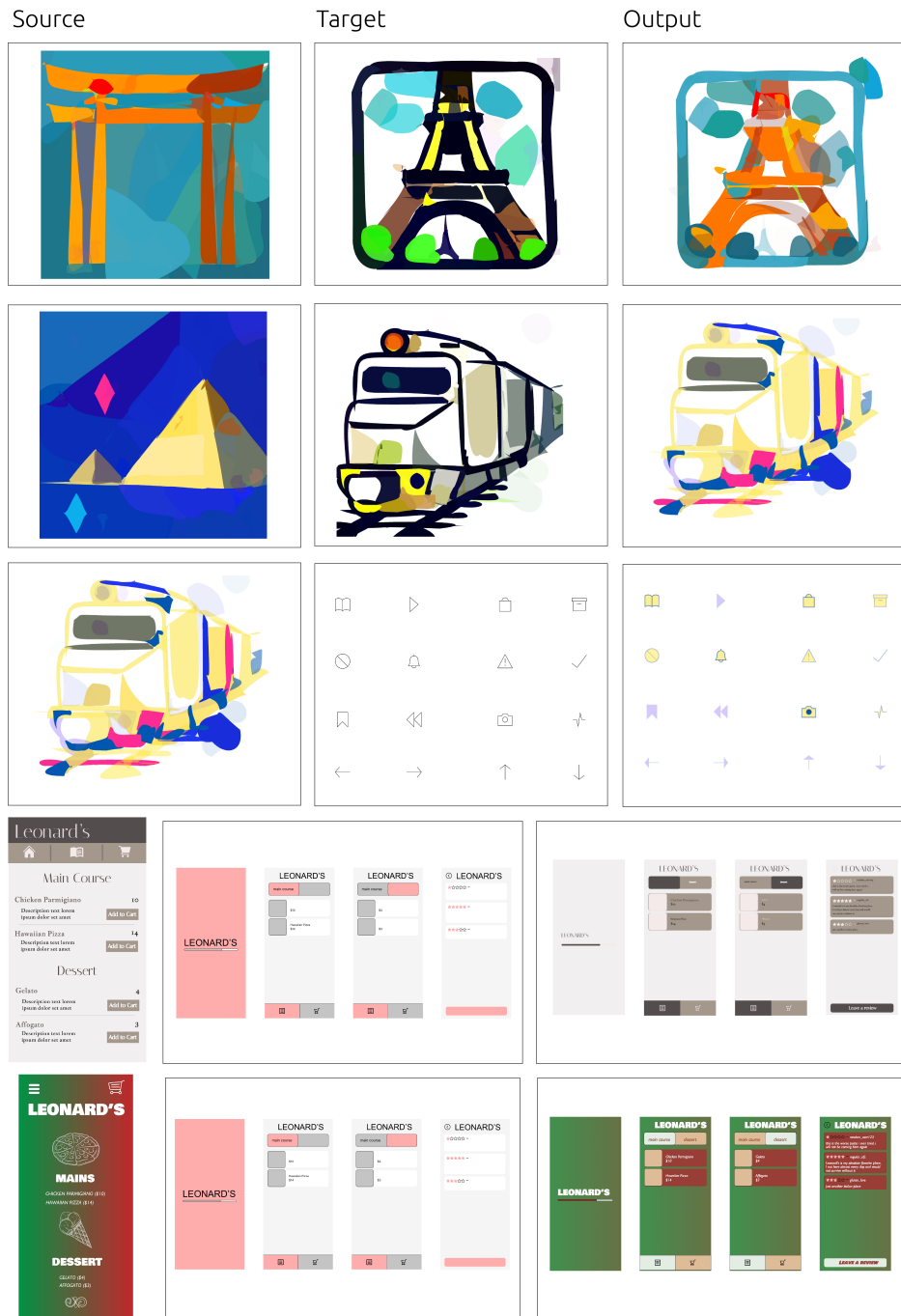


Figure 5.11: Additional graphics generated by transferring styles with VST.

Chapter 6

Vector Layout Transfer

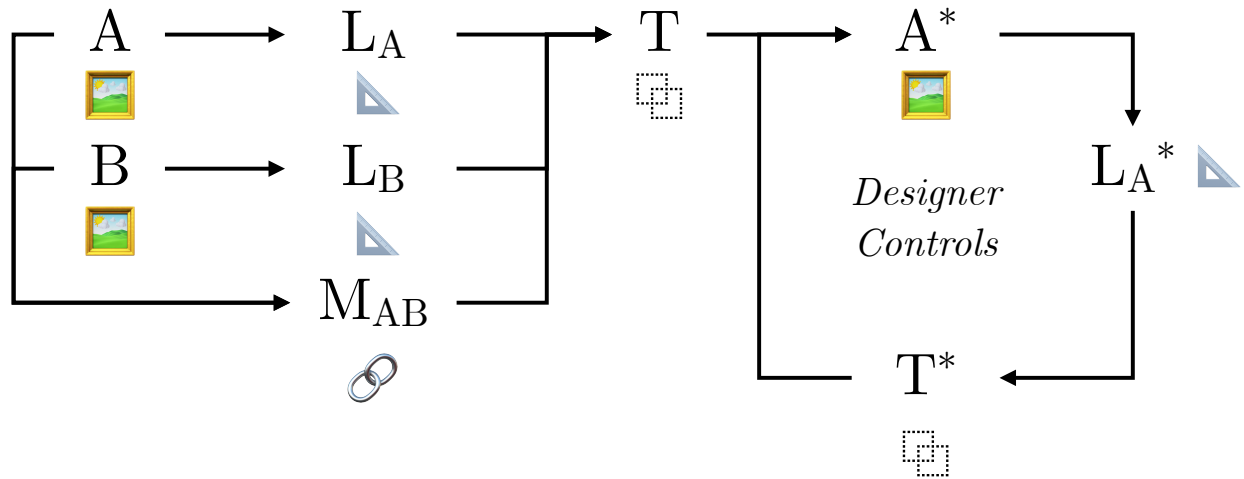
One key defining feature of a design besides its elements' styles is its layout - the spatial positioning and sizing of design elements relative to each other. Designers often explore layout alternatives and generate them by moving and resizing elements. The motivation for this can range from establishing a different visual flow, adapting a design to a different aspect ratio, standardizing spacing, or redirecting the design's visual emphasis. Existing designs can serve as a source of inspiration for layout modification across these goals. However, generating these layout alternatives still requires significant manual effort in rearranging large groups of elements. We present VLT, short for *Vector Layout Transfer*, a novel tool that provides new techniques (Table 6.1) for transforming designs which enables the flexible transfer of layouts between designs. It provides designers with multiple levels of semantic layout editing controls, powered by automatic graphics correspondence and layout optimization algorithms.

6.1 Introduction

Vector graphics designs have many benefits as a media format. Some artists primarily work with vector graphics over other representations because vectors best suit curvilinear geometry and give 'cleaner' aesthetics in their final result [87]. While this cleanliness and scalability are two reasons for vector graphics' success, another critical aspect is the flexibility of adapting layouts with discrete objects compared to editing rasterized images.

Humans have both natural biological inclinations and learned heuristics for inferring information from a design element's scale, position, and ordering. Perception of visual information is a well-established field, characterizing the different properties, aesthetics, and relations that objects can have to each other and what the effect is on the viewer [11, 119, 13].

This work presented in this chapter was first published in Warner et al. as *Interactively Optimizing Layout Transfer for Vector Graphics* in the 2023 International Conference on Machine Learning (ICML) AI/HCI Workshop [154].



Type Information





	<i>Design</i> : A, B, A^*		<i>Match Data</i> : M_{AB}
	<i>Layout Rules</i> : L_A, L_B, L_{A^*}		<i>Transformation</i> : T, T^*

Figure 6.1: Our layout transformation pipeline: given two vector graphics designs (A, B), we distill design layout data into grouped semantic layout rules for each design (L_A, L_B). We also compute a correspondence between the elements of the two designs (M_{AB}). Using L_A, L_B , and M_{AB} , we generate T : a transformation of the graphic design elements of A. Applying this transformation T yields design A^* , which we then distill new layout rules from (L_{A^*}). Designers can view the applied transformation and leverage control over which rules are prioritized, yielding new transformation T^* , which in turn yields a new design. This last component is an interactive, iterative process that aims to let designers retain full control of their design’s layout while benefitting from automation.

Larger elements tend to capture more attention from viewers, and the relative arrangement and position of individual elements also influence the design’s visual focus.

As a result, layouts are a core part of design in relation to attention and perception, ranging from map design [160], data visualizations [47], mobile user interfaces [111], and more generally across graphic design [175, 12, 37]. Skilled designers orchestrate these relational qualities, such as alignment, ordering, and sizing, to effectively allocate and streamline viewers’ attention toward the key information they aim to convey. This layout process is an iterative task involving resizing and moving many objects and possibly adding or removing

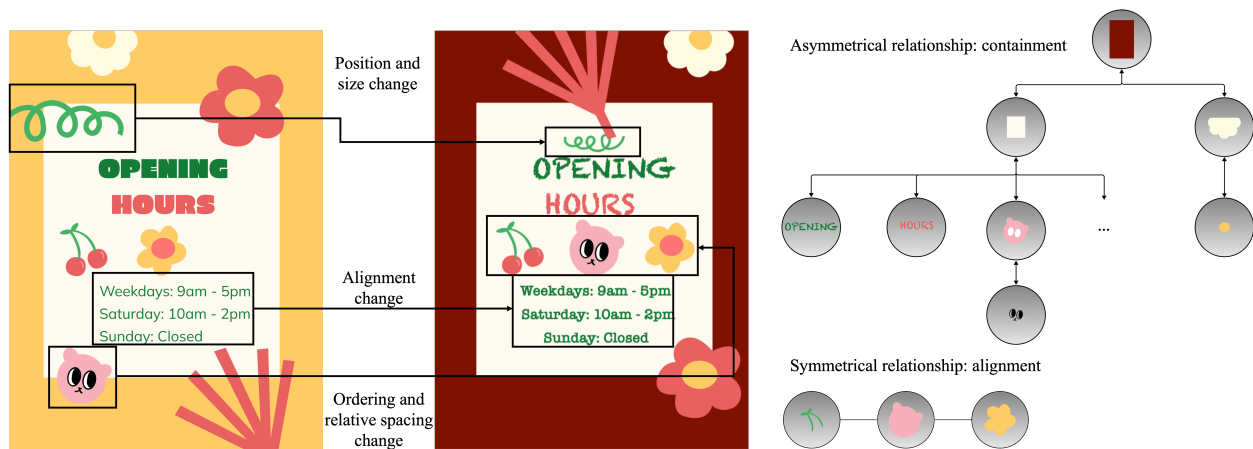


Figure 6.2: The left side of this figure shows two designs with varying layouts, along with differing layout rules that were inferred for corresponding groups of elements. The boxes and links in these designs represent different rule types that we recognize. The right side shows a representation of the different types of layout relationships we can model between elements. Asymmetric rules (e.g., containment) are represented internally as ordered trees while symmetric rules (e.g., alignment) are represented as simple sets (see also Table 6.2).

content altogether. Designers often explore the relational positions and layout of a vector graphics design to explore the effects of different variations [128].

Designers leverage many heuristics about what layout rules they should retain and which they should release to transform their designs. Editing relational features like ordering, relative offsets, and alignment for different groups of objects is a bottleneck task in this design process that diminishes the designers ability to explore new designs. While vector graphics are scalable, the relative dimensions (aspect ratio) and actual viewport size influence the preferred way to display information (e.g., mobile/desktop/poster/billboard), and reflowing an existing set of elements to a different size has been explored in related work [53].

However, often the source of inspiration for wanting to change the layout of a design is not simply resizing but matching another design’s layout; to *transfer* the layout from a source or given example design. Here, layouts are used to modify designs for greater purposes, including redirecting viewers’ attention across the design and redistributing visual emphasis within the same design elements. To facilitate this transfer of layouts across designs, we showcase a new tool (VLT) for vector graphic design layout transfer. Our approach to this layout design transfer problem is to (a) infer and parameterize layout rules present in a given design and (b) facilitate the interactive transfer and iterative refinement of those rules via multiple levels of semantic editing. We provide these varied levels of semantic

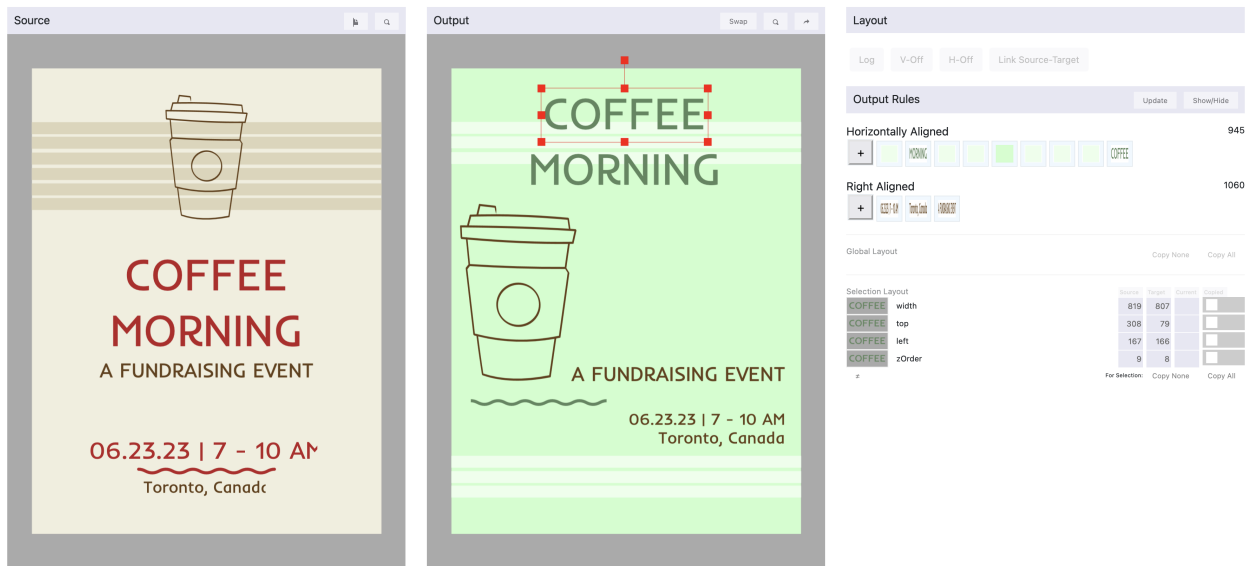


Figure 6.3: The VLT interface showing the source layout (e.g., B), the output layout (e.g., A*), and the layout rule customization panel. This output and the original target (A) can be toggled. The layout rules dynamically update as the output canvas is updated; here they show detected horizontal and right alignment rules. There are also global and element-specific layout transfer buttons, and a per-element property transfer based on that element’s matched element. This also works for multiple selected elements, grouping alike values.

editing and more powerful transformations with automatic graphics correspondence and layout optimization algorithms.

To enable *layout transfer*, we extract relational rules from a given source design and apply those layout rules to a given target design. This technique can reposition elements dynamically from a broad set of example designs. Enabling transfer involves (a) inferring which relationships to retain vs. those which to break, (b) creating a correspondence between the two designs’ elements to map adjustments across designs, and (c) computing and applying the minimal set of edits to integrate the source design’s layout.

Our approach also involves iteratively refining and specifying how the layout is transferred with a range of techniques (Table 6.1): (a) globally copying over layout rules for all elements, (b) copying all layout rules for a subset of elements, (c) specifying which rules design elements should adhere to, (d) specifying which properties to change per element, and finally (e) manually adjusting design elements with direct manipulation on the output canvas. The set of rules (e.g., L_A) for the output canvas updates in real time.

Our contributions include the following: **(1)** a description of a pipeline for interactively optimizing layout transfer across designs; **(2)** VLT, a novel tool that implements this pipeline; **(3)** an gallery of example results generated with our tool.

6.2 VLT Walkthrough

The broadest set of use cases for a tool like VLT is when designers would like to transform the layout of an existing design with a source reference design. Figure 6.1 shows an overview of how designers can use VLT to transfer layouts across designs, and Table 6.1 shows the core controls that VLT provides to designers for transforming the layout of their design using the source design as a source of inferred example rules. This walkthrough focuses on the iterative cycle designers can leverage to refine their output layout.

First, designers load two graphic designs A and B into VLT (A = target = existing design to transform, and B = source = reference design). Next, VLT will generate a correspondence matrix and match information (M_{AB}) between the two sets of design elements [136]. VLT also infers sets of semantic rules (listed in Table 6.2) for each layout.

Designers can then copy the layout of the previous source design globally by inferring the position and size from the matched elements across designs. The initial base transformation T uses the corresponding elements' base position and sizing, often giving subpar results (Figure 6.4). This naive case works on designs with a perfect one-to-one correspondence between design elements. However, many designs vary in the amount and type of elements they contain. Designs may also change in their canvas size or aspect ratio, which copying position and size alone cannot address.

In these cases, VLT can be used to retain and adjust layout rules present in the original target design. There is also an incremental rule-based optimization pipeline designers can leverage based on heuristic design rules (e.g., L_A). The dynamic set of layout rules that VLT infers can be viewed and modified in the right-most layout column of the interface (Figure 6.3), and a more detailed example with rule callouts is shown in Figure 6.2. The rule list updates according to the selected canvas elements. This brings the designers' attention to controls for leveraging these rules to modify their designs' layouts. Elements may be manually adjusted (i.e., direct manipulation) on the output canvas, and the set of detected layout rules updates in real time.

In addition to copying the layout of an entire design, designers may opt only to transfer (or reset) layout properties for specifically selected elements. Other elements can be added from layout rules here (clicking the + next to the rule member list) and conforming the marginal spacing across design versions. For example, selecting the H-Off or V-Off buttons will adjust the marginal spacing and offset for the currently selected elements to an inferred value based on their match. Designers may select elements from the source design (B),

Table 6.1: Designer Controls for Layout Editing

Granularity	Technique
Highest	Global Layout Copy
	Element Layout Copy
	Individual Rule Adherence
	Correspondence Update
	Element Property Copy
Lowest	Direct Manipulation

observe the rules they adhere to, and apply them (or a rule subset) to elements on the output canvas. Once satisfied, they can export the transformed design as an SVG.

6.3 Optimizing Layouts

To optimize the transferral of a layout across designs, we must first create a representation of that layout. We construct a transformation T that includes scale and translation amounts per graphic element to do this. Similarly, we first represent the layout of a specific visual design A as the *position* and *size* of each graphical element (e).

$$e \rightarrow [x, y, z, w, h] \tag{6.1}$$

Note that z here refers to the z-index or relative layering, while x and y refer to the uppermost, leftmost element canvas point for that element. Also, w and h refer to the element’s canvas width and height, respectively. So, a given transformation T to transform a graphic design A would consist of a set of changes to these element properties:

$$T \rightarrow \forall e \in A : [\delta x, \delta y, \delta z, \delta w, \delta h] \tag{6.2}$$

On top of this broad representation, we also build up sets of heuristic-based rules (e.g., L_A , L_B) that we can relate across multiple designs. These rules include containment, ordering, alignment, overlapping elements, relative margins, and size (Table 6.2), which may have either symmetric or asymmetric relations between elements. For example, alignment is symmetric in that all elements have the same relationship with each other (internally represented in VLT as a set), while containment has a structured ordering between related elements (internally represented as an ordered tree). Visual examples of the distinction between symmetric and asymmetric rules are shown in Figure 6.2.

Table 6.2: Supported Layout Heuristic Rules (e.g., L_A)

Type	Name
Asymmetric	Containment
	Relative Ordering
Symmetric	V/H Alignment
	Bounds Overlap
	Marginal Offset
	Same W/H

The optimal T choice for an exact one-to-one pairing of design elements is obvious – rescale and reposition the elements precisely to where they were in the corresponding design. However, there clearly are better ways to edit graphics than manually adjusting x and y coordinates. Recognizing and leveraging inferred design rules is a promising direction toward using automation while retaining designer control. We also want to handle complex one-to-many mappings between the sets of design elements.

First, layout rules from the source for corresponding elements are applied to the output graphics. This is initially done using the matched element’s position and size, which may cause multiple elements to overlap (Figure 6.4). To alleviate this, we also provide buttons to extend the marginal offset (Vertical-Offset/Horizontal-Offset) between matched elements onto the linked target elements. Individual rules can be specified to recompute a transformation that complies with the specified rule. This iterative optimization is an active project development area, and we detail ongoing work in our layout optimization in Sec. 7.3.

6.4 Design Results

To showcase the effectiveness of our method, we provide several example graphics that were transformed using the pipeline and tool detailed in this paper in Figure 6.4. The generation of these graphics was done by the authors using VLT. We aim to include more complex and varied examples, and have actual designers use VLT to transfer layouts across existing designs. For the graphics we generated, the amount of UI interactions to transform each design from Target to Final (per row) is 7/8/12/15, and the total number of transformed element properties is 111/76/291/128. The higher numbers for the property changes reflect that many properties can be changed with a single UI interaction in VLT. The procedure we followed to transfer layouts was to first match designs, transfer the global layout using the correspondence, leverage layout rules as needed, and finally tweak elements directly on the canvas. This follows granularity shown in Table 6.1; paint with the broadest strokes initially and iteratively handle smaller outlier classes.

6.5 Discussion

We discuss four main topics: (1) reflections from balancing designer control with boosting editing workflows with automation, and (2) different goals and methods for formulating the design layout transfer optimization task, and (3) ways for adding differentiability to this task to enable more a broader set of transfer techniques, and (4) limitations of working with layouts in this way and future steps we envision taking to address this.

Balancing Control & Automation

As automation-driven media creation and manipulation tools proliferate, there is a valid concern about displacing the designer from their current creative control. Our goal in this project is to retain the final control that designers have over their designs while reducing some of the tedium and manual labor that goes towards manifesting a specific vision for that given design. Our high-level approach towards this goal involves sharing a range of dynamic controls that the designer can adapt to the level of detail they wish to edit at, a sort of semantic range of design detail to operate over.

One of the ways we aim to provide this balance of control and automation includes providing several levels of detail and forms of editing and specifying transformation rules with VLT. This approach includes displaying inferred layout rules that can also modify existing designs, displaying editable global and element-specific layout data, and enabling live updates as the designer modifies their output (including via direct manipulation). Generally, the more deeply intertwined any automation becomes into existing creative practices necessitates deeper robustness and reliability to successfully operate ‘as expected’, which for many domains (image style transfer, text-to-image creation, vector layout transfer) remains a challenging and subjective task.

Layout Optimization

The current process for initially learning a layout transformation T is driven by correspondences, then refined by leveraging manually-crafted design heuristics. We want to leverage a more flexible approach to both initially craft and incorporate designer demonstrations and updates into design layout transformations. We envision using a combination of heuristic layout information currently gleaned from the SVG canvas and other vision-based UI understanding features to bolster the layout transformation and optimization process. Additionally, our current design transformation only consists of rescaling (height, width) and repositioning (x , y , z /layer) design elements. Other valid transformations exist, such as rotation and skew, but we have yet to implement them as we have found them less common. Enabling these transformations may yield additional desired variations that VLT cannot currently produce.

Source (B)	Target (A)	Auto (A*)	Final (A*)

Figure 6.4: An output gallery of layouts made with VLT. Each column shows (in order from columns 1-4): the source or inspiring layout (B), the target input design (A), the fully automatic result of globally applying layout transformation rules to the entire design, and the final output design iteratively made with VLT’s range of semantic editing designer controls.

We also take great inspiration from [76], which details a technique for learning the cost of connecting edges across a pair of web designs. In their work, a new semantic hierarchy is first inferred for both designs and then a minimal cost mapping across the vertices of the trees is computed. To do this, they train a standard multi-layer perceptron for training weights related to retaining tree ancestry, vertex siblings, and explicitly unmatched elements. This learning also considers the visual and semantic properties of each vertex that they

match. Their training is based on a set of human-provided mappings across visual design examples. Also, the optimization in their work focuses on producing a mapping between design elements, while we seek to optimize a transformation of one design’s layout based on that mapping, compared to the mapping itself.

Differentiable Layouts

Adherence to a discrete set of recognized layout rules is difficult to optimize because of the binary nature of rule groups – elements either adhere or not. To enable optimization of this discrete model, we are working to build a reward function R_T for transformation T based on the *relative* adherence and weight of inferred design heuristics and rules. We will apply Gaussian smoothing to the position and width/height constraints for symmetric relations like alignment, element overlap, offset, and sizing (Table 6.2). Here, r represents the layout rules that applying T yields, ω_i is the rule weight (which designers may adjust in a range of ways), and e_r measures how many elements correspond to that rule.

$$\begin{aligned}
 R_T &= R_{\text{rule}} + R_{\text{off}} + R_{\text{con}} \\
 R_{\text{rule}} &= \sum_r \omega_r * \log(e_r + 1) \\
 R_{\text{off}} &= \omega_{\text{off}} * t_{\text{non-overlap}} \\
 R_{\text{con}} &= \frac{\omega_{\text{con}}}{e_{\text{unique-prop}}}
 \end{aligned}
 \tag{6.3}$$

In addition to general rule adherence, we propose metrics R_{off} for balancing the relative offset of objects (e.g., favor non-occlusion of text) and R_{con} for increasing the numeric consistency of almost-alike element properties, a sort of snap-to-fit implementation (e.g., favor sizing/spacing). Also, $t_{\text{non-overlap}}$ refers to the non-overlapping text elements, and $e_{\text{unique-prop}}$ refers to the number of unique properties that exist in a design (less is better). These rewards also will global adjustable weights (ω_{off} , ω_{con}), respectively.

Designers will be able to selectively apply this optimization to part of the design or simply run it over the entire output design. In addition, we can optimize specific inferred rules from the source or target while retaining as much structure from the alternative goals as possible by explicitly increasing the weight of those sections. Designers could opt to lock constrained element properties in their design (e.g., size) to ensure those properties are not modified, or extend a manually demonstrated layout change to similar elements.

Chapter 7

Conclusion

I'll conclude this dissertation by reviewing the presented tools and their key design concepts, restating the thesis contributions, discussing future work directions, and finally presenting closing remarks on delegating human designer control with automation-powered tools.

First, I focus on *reflection* – the act of gathering and absorbing feedback or information after some visual media is created. The focused domain is presentations, and I demonstrate SlideSpecs, a novel system for automatically and interactively collating and contextualizing talk feedback. Grounded in our formative study findings, SlideSpecs allows presenters to organize and review their presentation feedback effectively. SlideSpecs improves the revision process by unifying text and spoken feedback into a centralized space for seamless review. I demonstrate the effectiveness of SlideSpecs by deploying it in eight unique presentations across computer vision, programming notebooks, sensemaking, and more. Presenters reported that using SlideSpecs while refining their talk improved their feedback organization, provided valuable context, and reduced redundant comments. Future collaborative revision and feedback systems can benefit users by integrating more automatic contextualization.

Next, I shift to *recomposition* – the act of adapting and transforming some existing visual media. The focused domain is vector graphics, providing a concise yet expressive way to represent a broad range of visual designs. I present VST, a system for interactively and flexibly transferring visual styles between vector graphics. While automatic *theme selection* mechanisms exist, they are often too inflexible or require an underlying data structure not present in many designs. In VST, designers can inspect and correct correspondences and decide which graphics attributes should be transferred for which correspondence pairs. Together, these interactions enable rapid and flexible style transfer. I extend work on automatic graphics correspondence algorithms and contribute a way for designers to work flexibly with such algorithms. I report results from a user study in which designers transferred visual styles between pairs of designs, including designs they provided.

I also present a novel design tool, VLT, that can enable interactive layout transfer optimization. VLT’s process for inferring and transferring layouts (Figure 6.1) integrates automation into the design process while providing several levels of automation-driven semantic control and editing techniques (Table 6.1) for designers to steer and adjust the resulting final layout. I showcase some preliminary generated results (Figure 6.4) and highlight several important next steps for design layout transfer.

7.1 Restatement of Contributions

The contributions of this dissertation include the following:

- Reflection Support for Presentations:
 - SlideSpecs, a novel system for collating audience text and spoken presentation feedback which also presents a novel screenshot-based slide-detecting technique for automatic feedback contextualization.
 - Design implications for group slide-feedback interfaces derived from formative interviews and an evaluation applying SlideSpecs to eight talks across different research groups and topics and an in-depth analysis of our findings.
- Recomposition Support for Vector Graphics:
 - VST, a design tool that introduces a novel user interface for interactive, user-guided, flexible style transfer for vector graphics. Its key interaction principles are: a) enabling users to edit computed correspondences at multiple levels, and b) enabling users to customize how attributes are transferred between designs across the correspondence.
 - Two VST user studies that demonstrate: a) that designers can successfully transfer styles between graphics with VST, and b) that designers without VST can spend more time and effort to produce equivalent design results.
 - A description of a pipeline for interactively optimizing layout transfer across designs and VLT, a novel tool that implements this pipeline, and a gallery of example vector graphics results generated with VLT.

7.2 Future Work - Reflection

There are a few immediate directions for future work to extend. First, I’ll focus on *reflection* before discussing *recomposition*. An exciting research vision aligning with this thesis is integrating AI and NLP techniques to boost the usefulness of reflection-centric tools like SlideSpecs. I’ll review the most promising future work supporting reflection via feedback understanding and organization here.

Reducing the audience workload.

An integral part of feedback classification and distilling key themes within SlideSpecs relies on discussion facilitators. For example, facilitators markup post-presentation discussions with links to comments and new topics, distilling key discussion points into actionable feedback. In addition, audience members contribute to the context by tagging comments, using side references in feedback, and discussion facilitation. Future systems could focus on reducing facilitator and audience work by automatically semantically grouping and labeling feedback. Discussions could be enhanced if a system could automatically recommend topics based on the magnitude of similar feedback. Future work can enhance feedback systems by automating the facilitator’s tasks and reducing audience members’ labor by inferring more context.

Generating actionable feedback.

In our study, I worked with groups that had roughly ten audience members (Table 4.1). While there is no *correct* number of audience members, this size worked well for these presentations. A larger audience size could yield a higher quantity of valuable feedback and also increase the relevance of the automatic summarization of the provided comments.

Future work could automatically generate more context by allocating comments (both written and spoken) to slides as they are entered based on their content. This may pave the way to supporting more dynamic feedback processes that feature common audience interruptions. While increased group awareness can reduce redundancy when receiving feedback from a group, participants still sometimes enter similar redundant comments. Adding an automatic comment summarization and aggregation pipeline into the *Review* phase would further streamline the presenter experience. Finally, LLMs could transform these aggregated feedback points into a concise list of more refined actionable changes.

Reflection for vector graphics.

This dissertation focused on *recomposition* of vector graphics, though just like presentation authors, feedback and *reflection* are essential for graphic designers. What are the sources of feedback (e.g., written comments, spoken discussion, reference designs) are most common and useful for vector graphics? How can that feedback be attached and contextualized within the design’s context? Charrette [107] pushes in this direction for user interface design, supporting direct design annotations, exploring design histories, and facilitating the curation and presentation of these related information domains. Other directions could include leveraging more automatically generated design feedback (e.g., applying design heuristics as a type of linting) or integrating aggregated quantitative human feedback onto designs.

7.3 Future Work - Recomposition

Better element correspondences.

When designs have similar elements, finding a correspondence between the two element sets (M_{AB}) is natural. However, this element correspondence between designs will often be noisier or less accurate for unrelated or immense designs. One direction for future work I envision is being able to dynamically infer a set of joint classes across elements, of which design elements might belong to many, as opposed to a cross-design element map. VLT shows grouped layout rules and property changes, but the level of inference could be smoother and capture a broader set of similarities to enhance designer control.

The vector graphics style transfer issue could potentially be addressed by inferring classes for each design element. VST and VLT generate correspondences with the algorithm described in [136]. A different algorithm could create and infer shared dynamic classes for each design element. Then, the element-to-element correspondence issue instead becomes a class-to-class correspondence, which could be a more straightforward problem. While I geared our recomposition tools towards mixing inspirational ideas from existing documents, this is just one part of the design process. Leveraging a more robust many-to-many correspondence between design elements could yield richer and more varied style transfers.

Styling vectors with images.

Images are a natural way to add vibrancy to a design. A well-picked image instantly sets a mood for the design and can provide a source of consistent branding. However, our style transfer approach mainly applies to native vector elements, such as `<path>` or `<rect>` elements. One suggested modification was to allow sourcing styles from images since this format is more readily available. From VST – RD1: It would be great to apply bitmap styling to my vector design. This use case is more common in my workflow. This would require converting the image to vector graphics or a novel style extraction technique. There are research methods for image-to-vector graphics conversion [129] and commercial tools [1].

When inspecting the structure of the output of these methods, they tend to optimize similarity to the source image rather than having a consistent internal semantic element resolution which makes applying a correspondence graph-kernel technique more challenging [136]. Additionally, rasterizing vector graphics to an image is inherently lossy, so no perfect inverse process can exist. Some features (e.g., flat color) are simple to extract for novel style extraction, while other features like gradients, shape, and font are more challenging. D6 (from VST’s evaluation) reported that they previously tried online tools that attempt to identify fonts from a given image [35, 36]. Given the relative prevalence of images, enabling vector styling (even if limited) via bitmap sources is a promising future direction.

More complex transformations.

Primarily, our style transfer with this prototype addresses element size, font, stroke, and fill. While designers can modify other features, this feature subset visually dominates the result. Future work could serve as a larger-scale unification technique where many designs are edited simultaneously. By constraining our definition of style, I simplify the process of style transfer and reduce our tool’s capacity. For example, the design layout and structure are implicitly constant throughout our style transfer process. Applying the layout from source to target is an exciting and relevant next direction. Exciting work from Hoffswell et al. [53] (visualization remapping for dynamic viewports) and Ye et al. [168] (automatic mathematical diagramming) push in this direction. Future work should continue exploring more complex transformations for vector graphics.

Recomposition for presentations.

Though this dissertation focused mainly on supporting reflection with presentations, considering *recomposition* for presentations. The work is not quite over once the presenter has reviewed and synthesized a list of concrete changes to make. They must still manually review their slides and update the content according to the gathered feedback. Depending on the format of the presentation, it is very feasible to envision some of these transformations being automated. While proprietary formats like PowerPoint and Keynote may be less accessible, some presenters use open text-based slide formats, including L^AT_EX (with Beamer¹), HTML (with Reveal.js²), and Markdown (with Marp³). Existing LLMs could transform this existing text-based slide format based on the feedback gathered from the audience. Future work could transform existing slide decks, including synthesizing new slides, modifying existing slides, cutting out slides, and restructuring or reordering the presentation slides.

7.4 Closing Remarks

This thesis was written at a pivotal time in generative media tools, which leads to a more extensive discussion about ownership and designer control. As tool creators, HCI researchers have a certain level of control over what designers can do and how they think about the process of design based on what the tool supports [122]. While tools designers and researchers are often sensitive to this dichotomy, it inevitably shapes the research and the tools produced, also shaping tool users’ creations and design conceptions.

Significant tensions exist between artists and writers whose output gets leveraged as high-quality examples for reinforcement learning and foundation model training, fueled by the recent emergence and commercial promise of generative AI and LLM startups. With

¹[https://en.wikipedia.org/wiki/Beamer_\(LaTeX\)](https://en.wikipedia.org/wiki/Beamer_(LaTeX))

²<https://github.com/hakimel/reveal.js/>

³<https://github.com/marp-team/marp>

that said, these startups’ broad ability to generate and modify text and images is set to reshape one of the last bastions of labor that resisted automation – white-collar information work, everything from data management to writing to visual design jobs. This thesis does not come bundled with a solution to delegate power and control over their work back to designers and creators, though this tension will get heightened as workers in these fields get pushed out and devalued by corporations seeking to optimize profits; ultimately, humans are expensive. Ideally, computers would help automate the ‘boring’ parts of work to free humans to be more expressive and creative. However, corporations invariably aim to extract and automate those aspects of human culture for financial gain.

There are also questions about what the visual arts have in the role of personal expression compared to being a professional career path to sustain oneself. Will the proliferation of different generative models be akin to the advent of cheap digital cameras in which everyone could become a photographer? There still is a need for professionals and perhaps even a greater cultural appreciation for those with very high skill levels and training in that domain. Perhaps the same will be valid for this next wave of visual media creation tools.

There is a certain sense that art and design will evolve in response to this tension between *optimizing* a design and the resulting homogeneity of having all designs converge on a specific style or feel. There is a shifting metagame in visual communication language where objects/pieces outside of the widespread distribution can be mesmerizing and groundbreaking. This variance might be a small source of hope for the creative labor space.

While generative AI and media mixing tools lower the barrier to entry for creation and expression, there is still human taste and reflection that goes into getting feedback and a new domain of subjectivity and expression that comes with these new, more flexible tools and representations of visual media. This subjectivity helps motivate my work on reflection in visual media to encourage and support more profound thoughts on what people create. In the same way that consumer snap-and-shoot cameras ‘commodified’ the image, perhaps this set of tools for enhancing visual media refinement and reflection can lower the barrier to entry for expression and visual communication in a way that widens the spectrum of produced content.

I’ve presented a range of techniques and systems for enhancing visual media with *reflection* and *recomposition*. This is an exciting moment as automation-powered tools shift the landscape of possible creative processes. The goal of this dissertation is to broaden the focus on media creation to better support iterative design. This broader focus is important for retaining meaningful human influence in digital visual media creation processes.

Bibliography

- [1] Adobe. *Adobe Express: Convert Images to SVG*. Adobe. Aug. 2022. URL: <https://www.adobe.com/express/feature/image/convert/svg>.
- [2] Diego Martin Arroyo, Janis Postels, and Federico Tombar. “Variational Transformer Networks for Layout Generation”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2021, pp. 13637–13647. DOI: [10.1109/CVPR46437.2021.01343](https://doi.org/10.1109/CVPR46437.2021.01343).
- [3] Reza Asadi, Ha Trinh, Harriet J. Fell, and Timothy W. Bickmore. “IntelliPrompter: Speech-Based Dynamic Note Display Interface for Oral Presentations”. In: *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. ICMI ’17. Glasgow, UK: Association for Computing Machinery, 2017, pp. 172–180. ISBN: 9781450355438. DOI: [10.1145/3136755.3136818](https://doi.org/10.1145/3136755.3136818).
- [4] Reza Asadi, Ha Trinh, Harriet J. Fell, and Timothy W. Bickmore. “Quester: A Speech-Based Question Answering Support System for Oral Presentations”. In: *23rd International Conference on Intelligent User Interfaces*. IUI ’18. Tokyo, Japan: Association for Computing Machinery, 2018, pp. 583–593. ISBN: 9781450349451. DOI: [10.1145/3172944.3172974](https://doi.org/10.1145/3172944.3172974).
- [5] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. “Multi-Content GAN for Few-Shot Font Style Transfer”. In: 2017. arXiv: [1712.00516](https://arxiv.org/abs/1712.00516) [cs.CV].
- [6] Olivier Bau and Wendy E. Mackay. “OctoPocus: A Dynamic Guide for Learning Gesture-Based Command Sets”. In: *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. UIST ’08. Monterey, CA, USA: Association for Computing Machinery, 2008, pp. 37–46. ISBN: 9781595939753. DOI: [10.1145/1449715.1449724](https://doi.org/10.1145/1449715.1449724).
- [7] Karim Benharrak, Florian Lehmann, Hai Dang, and Daniel Buschek. “SummaryLens – A Smartphone App for Exploring Interactive Use of Automated Text Summarization in Everyday Life”. In: *27th International Conference on Intelligent User Interfaces*. IUI ’22 Companion. Helsinki, Finland: Association for Computing Machinery, 2022, pp. 93–96. ISBN: 9781450391450. DOI: [10.1145/3490100.3516471](https://doi.org/10.1145/3490100.3516471).

- [8] Edward O. Benson and David R. Karger. “Cascading Tree Sheets and Recombinant HTML: Better Encapsulation and Retargeting of Web Content”. In: *Proceedings of the 22nd International Conference on World Wide Web*. WWW ’13. Rio de Janeiro, Brazil: Association for Computing Machinery, 2013, pp. 107–118. ISBN: 9781450320351. DOI: [10.1145/2488388.2488399](https://doi.org/10.1145/2488388.2488399).
- [9] Gary Bernhardt. *How to Prepare a Talk*. 2018. URL: <https://www.deconstructconf.com/blog/how-to-prepare-a-talk>.
- [10] Tim Brooks, Aleksander Holynski, and Alexei A Efros. “Instructpix2pix: Learning to follow image editing instructions”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 18392–18402.
- [11] Nicola Bruno and James E Cutting. “Minimodularity and the perception of layout.” In: *Journal of Experimental Psychology: General* 117.2 (1988), p. 161.
- [12] Zoya Bylinskii, Nam Wook Kim, Peter O’Donovan, Sami Alsheikh, Spandan Madan, Hanspeter Pfister, Fredo Durand, Bryan Russell, and Aaron Hertzmann. “Learning visual importance for graphic designs and data visualizations”. In: *Proceedings of the 30th Annual ACM symposium on user interface software and technology*. 2017, pp. 57–69.
- [13] Mackinlay Card. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.
- [14] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. “DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation”. In: *CoRR abs / 2007.11301* (2020). arXiv: [2007.11301](https://arxiv.org/abs/2007.11301).
- [15] Shang Chai, Liansheng Zhuang, and Fengying Yan. “LayoutDM: Transformer-based Diffusion Model for Layout Generation”. In: *arXiv preprint arXiv:2305.02567* (2023).
- [16] Chin-Yi Cheng, Forrest Huang, Gang Li, and Yang Li. “PLay: Parametrically Conditioned Layout Generation using Latent Diffusion”. In: *arXiv preprint arXiv:2301.11529* (2023).
- [17] Patrick Chiu, John S Boreczky, Andreas Girgensohn, and Don Kimber. “LiteMinutes: an Internet-based system for multimedia meeting minutes.” In: *WWW* 1 (2001), pp. 140–149.
- [18] Patrick Chiu, Ashutosh Kapuskar, Sarah Reitmeier, and Lynn Wilcox. “NoteLook: Taking notes in meetings with digital video and ink”. In: *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*. ACM. 1999, pp. 149–158.
- [19] Kwangsu Cho and Christian D Schunn. “Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system”. In: *Computers & Education* 48.3 (2007), pp. 409–426.

- [20] Michael Collins. “Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms”. In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP '02. USA: Association for Computational Linguistics, 2002, pp. 1–8. DOI: [10.3115/1118693.1118694](https://doi.org/10.3115/1118693.1118694).
- [21] Yaakov Danone, Tsvi Kuflik, and Osnat Mokryn. “Visualizing Reviews Summaries as a Tool for Restaurants Recommendation”. In: *23rd International Conference on Intelligent User Interfaces*. IUI '18. Tokyo, Japan: Association for Computing Machinery, 2018, pp. 607–616. ISBN: 9781450349451. DOI: [10.1145/3172944.3172947](https://doi.org/10.1145/3172944.3172947).
- [22] Niraj Ramesh Dayama, Simo Santala, Lukas Brückner, Kashyap Todi, Jingzhou Du, and Antti Oulasvirta. “Interactive Layout Transfer”. In: *26th International Conference on Intelligent User Interfaces*. IUI '21. College Station, TX, USA: Association for Computing Machinery, 2021, pp. 70–80. ISBN: 9781450380171. DOI: [10.1145/3397481.3450652](https://doi.org/10.1145/3397481.3450652).
- [23] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibsichman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. “Rico: A Mobile App Dataset for Building Data-Driven Design Applications”. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST '17. Québec City, QC, Canada: Association for Computing Machinery, 2017, pp. 845–854. ISBN: 9781450349819. DOI: [10.1145/3126594.3126651](https://doi.org/10.1145/3126594.3126651).
- [24] Laurent Denoue, Scott Carter, and Matthew Cooper. “Searching Live Meeting Documents ”Show Me the Action””. In: *Proceedings of the 2015 ACM Symposium on Document Engineering*. DocEng '15. Lausanne, Switzerland: Association for Computing Machinery, 2015, pp. 195–198. ISBN: 9781450333078. DOI: [10.1145/2682571.2797082](https://doi.org/10.1145/2682571.2797082).
- [25] Praveen Kumar Dhanuka, Nirmal Kumawat, and Nipun Jindal. “Vector based glyph style transfer”. In: *ACM SIGGRAPH 2019 Posters*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–2.
- [26] divRIOTS. *html.to.design*. divRIOTS. 2023. URL: <https://www.figma.com/community/plugin/1159123024924461424/html.to.design> (visited on 01/30/2023).
- [27] Tom Djajadiningrat, Kees Overbeeke, and Stephan Wensveen. “But how, Donald, tell us how? On the creation of meaning in interaction design through feedforward and inherent feedback”. In: *Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques*. New York, NY, USA: Association for Computing Machinery, 2002, pp. 285–291.
- [28] Mark Dredze, Hanna M. Wallach, Danny Puller, and Fernando Pereira. “Generating Summary Keywords for Emails Using Topics”. In: *Proceedings of the 13th International Conference on Intelligent User Interfaces*. IUI '08. Gran Canaria, Spain: As-

- sociation for Computing Machinery, 2008, pp. 199–206. ISBN: 9781595939876. DOI: [10.1145/1378773.1378800](https://doi.org/10.1145/1378773.1378800).
- [29] Steven M. Drucker, Georg Petschnigg, and Maneesh Agrawala. “Comparing and Managing Multiple Versions of Slide Presentations”. In: *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*. UIST ’06. Montreux, Switzerland: Association for Computing Machinery, 2006, pp. 47–56. ISBN: 1595933131. DOI: [10.1145/1166253.1166263](https://doi.org/10.1145/1166253.1166263).
- [30] Mahika Dubey, Jasmine T. Otto, and Angus Graeme Forbes. “Data Brushes: Interactive Style Transfer for Data Art”. In: *2019 IEEE VIS Arts Program (VISAP) (2019)*, pp. 1–9.
- [31] Jane L. E. Ohad Fried, and Maneesh Agrawala. “Optimizing Portrait Lighting at Capture-Time Using a 360 Camera as a Light Probe”. In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. UIST ’19. ACM. New York, NY, USA: Association for Computing Machinery, 2019, pp. 221–232. DOI: [10.1145/3332165.3347893](https://doi.org/10.1145/3332165.3347893).
- [32] Shirin Feiz, Jason Wu, Xiaoyi Zhang, Amanda Swearngin, Titus Barik, and Jeffrey Nichols. “Understanding Screen Relationships from Screenshots of Smartphone Applications”. In: *27th International Conference on Intelligent User Interfaces*. IUI ’22. Helsinki, Finland: Association for Computing Machinery, 2022, pp. 447–458. ISBN: 9781450391443. DOI: [10.1145/3490099.3511109](https://doi.org/10.1145/3490099.3511109).
- [33] Michael H. Fischer, Richard R. Yang, and Monica S. Lam. “ImagineNet: Restyling Apps Using Neural Style Transfer”. In: *CoRR abs/2001.04932 (2020)*. arXiv: [2001.04932](https://arxiv.org/abs/2001.04932).
- [34] Matthew Fisher, Manolis Savva, and Pat Hanrahan. “Characterizing Structural Relationships in Scenes Using Graph Kernels”. In: *ACM SIGGRAPH 2011 Papers*. SIGGRAPH ’11. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2011. ISBN: 9781450309431. DOI: [10.1145/1964921.1964929](https://doi.org/10.1145/1964921.1964929).
- [35] FontFinder. *FontFinder by What Font Is*. Aug. 2022. URL: <https://www.whatfontis.com/>.
- [36] FontSquirrel. *Identify Fonts - The Font Squirrel Matcherator*. Aug. 2022. URL: <https://www.fontsquirrel.com/matcherator>.
- [37] Camilo Fosco, Vincent Casser, Amish Kumar Bedi, Peter O’Donovan, Aaron Hertzmann, and Zoya Bylinskii. “Predicting visual importance across graphic design types”. In: *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 2020, pp. 249–260.
- [38] C. Ailie Fraser, Tricia J. Ngoon, Ariel S. Weingarten, Mira Dontcheva, and Scott Klemmer. “CritiqueKit: A Mixed-Initiative, Real-Time Interface For Improving Feedback”. In: *UIST ’17 Adjunct (2017)*, pp. 7–9. DOI: [10.1145/3131785.3131791](https://doi.org/10.1145/3131785.3131791).

- [39] Werner Geyer, Heather Richter, and Gregory D Abowd. “Towards a smarter meeting record—capture and access of meetings revisited”. In: *Multimedia Tools and Applications* 27.3 (2005), pp. 393–410.
- [40] Elena L Glassman, Juho Kim, Andrés Monroy-Hernández, and Meredith Ringel Morris. “Mudslide: A spatially anchored census of student confusion for online lecture videos”. In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 1555–1564.
- [41] Lance Good and Benjamin B Bederson. *CounterPoint: Creating jazzy interactive presentations*. Tech. rep. 2001.
- [42] Thomas R. G. Green and Marian Petre. “Usability analysis of visual programming environments: a ‘cognitive dimensions’ framework”. In: *Journal of Visual Languages & Computing* 7.2 (1996), pp. 131–174.
- [43] Jonathan Grudin. “Groupware and social dynamics: Eight challenges for developers”. In: *Readings in Human–Computer Interaction*. Elsevier, 1995, pp. 762–774.
- [44] David Ha and Douglas Eck. “A neural representation of sketch drawings”. In: *arXiv preprint arXiv:1704.03477* (2017).
- [45] Jonathan Harper and Maneesh Agrawala. “Deconstructing and Restyling D3 Visualizations”. In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST ’14. Honolulu, Hawaii, USA: Association for Computing Machinery, 2014, pp. 253–262. ISBN: 9781450330695. DOI: [10.1145/2642918.2647411](https://doi.org/10.1145/2642918.2647411).
- [46] Drew Harry, Joshua Green, and Judith Donath. “Backchan.nl: integrating backchannels in physical space”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2009, pp. 1361–1370.
- [47] Jeffrey Heer and Michael Bostock. “Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’10. Atlanta, Georgia, USA: Association for Computing Machinery, 2010, pp. 203–212. ISBN: 9781605589299. DOI: [10.1145/1753326.1753357](https://doi.org/10.1145/1753326.1753357).
- [48] Lena Hegemann, Niraj Ramesh Dayama, Abhishek Iyer, Erfan Farhadi, Ekaterina Marchenko, and Antti Oulasvirta. “CoColor: Interactive Exploration of Color Designs”. In: *Proceedings of the 28th International Conference on Intelligent User Interfaces*. IUI ’23. Sydney, NSW, Australia: Association for Computing Machinery, 2023, pp. 106–127. DOI: [10.1145/3581641.3584089](https://doi.org/10.1145/3581641.3584089).
- [49] Brian Hempel, Justin Lubin, and Ravi Chugh. “Sketch-n-Sketch: Output-Directed Programming for SVG”. In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. UIST ’19. New Orleans, LA, USA: Association for Computing Machinery, 2019, pp. 281–292. ISBN: 9781450368162. DOI: [10.1145/3332165.3347925](https://doi.org/10.1145/3332165.3347925).

- [50] Scarlett R Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P Bailey. “Getting inspired! Understanding how and why examples are used in creative design practice”. In: *Proceedings of the SIGCHI conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery, 2009, pp. 87–96.
- [51] Yoshinori Hijikata, Hanako Ohno, Yukitaka Kusumura, and Shogo Nishida. “Social Summarization of Text Feedback for Online Auctions and Interactive Presentation of the Summary”. In: *Proceedings of the 11th International Conference on Intelligent User Interfaces. IUI '06*. Sydney, Australia: Association for Computing Machinery, 2006, pp. 242–249. ISBN: 1595932879. DOI: [10.1145/1111449.1111500](https://doi.org/10.1145/1111449.1111500).
- [52] Raphaël Hoarau and Stéphane Conversy. “Augmenting the Scope of Interactions with Implicit and Explicit Graphical Structures”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '12*. Austin, Texas, USA: Association for Computing Machinery, 2012, pp. 1937–1946. ISBN: 9781450310154. DOI: [10.1145/2207676.2208337](https://doi.org/10.1145/2207676.2208337).
- [53] Jane Hoffswell, Wilmot Li, and Zhicheng Liu. “Techniques for Flexible Responsive Visualization Design”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. CHI '20*. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–13. ISBN: 9781450367080. DOI: [10.1145/3313831.3376777](https://doi.org/10.1145/3313831.3376777).
- [54] Enamul Hoque and Maneesh Agrawala. “Searching the Visual Style and Structure of D3 Visualizations”. In: *IEEE Transactions on Visualization and Computer Graphics* 26 (2020), pp. 1236–1245.
- [55] Nathan Hurst, Wilmot Li, and Kim Marriott. “Review of Automatic Document Formatting”. In: *Proceedings of the 9th ACM Symposium on Document Engineering. DocEng '09*. Munich, Germany: Association for Computing Machinery, 2009, pp. 99–108. ISBN: 9781605585758. DOI: [10.1145/1600193.1600217](https://doi.org/10.1145/1600193.1600217).
- [56] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. “LayoutDM: Discrete Diffusion Model for Controllable Layout Generation”. In: *arXiv preprint arXiv:2303.08137* (2023).
- [57] Hiroshi Ishii and Minoru Kobayashi. “ClearBoard: a seamless medium for shared drawing and conversation with eye contact”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1992, pp. 525–532.
- [58] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Los Alamitos, CA, USA: IEEE Computer Society, 2017, pp. 1125–1134.
- [59] Jennifer Jacobs, Joel Brandt, Radomir Mech, and Mitchel Resnick. “Extending manual drawing practices with artist-centric programming tools”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–13.

- [60] Jennifer Jacobs, Sumit Gogia, Radomir Mvech, and Joel R Brandt. “Supporting expressive procedural art creation through direct manipulation”. In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 6330–6341.
- [61] Ajay Jain, Amber Xie, and Pieter Abbeel. “VectorFusion: Text-To-SVG By Abstracting Pixel-based Diffusion Models”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, 2023, pp. 1911–1920.
- [62] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual losses for real-time style transfer and super-resolution”. In: *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part II 14*. Springer. Los Alamitos, CA, USA: IEEE Computer Society, 2016, pp. 694–711.
- [63] Hyeungshik Jung, Hijung Valentina Shin, and Juho Kim. “DynamicSlide: Reference-Based Interaction Techniques for Slide-Based Lecture Videos”. In: *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*. UIST ’18 Adjunct. Berlin, Germany: Association for Computing Machinery, 2018, pp. 23–25. ISBN: 9781450359498. DOI: [10.1145/3266037.3266089](https://doi.org/10.1145/3266037.3266089).
- [64] Young-Wook Jung, Youn-kyung Lim, and Myung-suk Kim. “Possibilities and Limitations of Online Document Tools for Design Collaboration: The Case of Google Docs”. In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM. 2017, pp. 1096–1108.
- [65] Victor Kaptelinin and Bonnie Nardi. “Affordances in HCI: Toward a Mediated Action Perspective”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’12. Austin, Texas, USA: Association for Computing Machinery, 2012, pp. 967–976. ISBN: 9781450310154. DOI: [10.1145/2207676.2208541](https://doi.org/10.1145/2207676.2208541).
- [66] Fawzia Khan. *A survey of note-taking practices*. Hewlett-Packard Laboratories, 1993.
- [67] Kotaro Kikuchi, Mayu Otani, Kota Yamaguchi, and Edgar Simo-Serra. “Modeling Visual Containment for Web Page Layout Optimization”. In: *Computer Graphics Forum*. Vol. 40-7. Wiley Online Library. New York, NY, USA: Wiley Online Library, 2021, pp. 33–44.
- [68] Scott R Klemmer, Michael Thomsen, Ethan Phelps-Goodman, Robert Lee, and James A Landay. “Where do web sites come from?: capturing and interacting with design history”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2002, pp. 1–8.
- [69] Janin Koch, Magda Laszlo, Andres Lucero, and Antti Oulasvirta. “Surfing for Inspiration: digital inspirational material in design practice”. In: *Design Research Society International Conference*. Design Research Society. New York, NY, USA: DRS, 2018, pp. 1247–1260.

- [70] Soheil Kolouri, Navid Naderializadeh, Gustavo Kunde Rohde, and Heiko Hoffmann. “Wasserstein Embedding for Graph Learning”. In: *ArXiv abs/2006.09430* (2020).
- [71] Risi Kondor and Horace Pan. “The Multiscale Laplacian Graph Kernel”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.
- [72] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. “BLT: bidirectional layout transformer for controllable layout generation”. In: *Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*. Springer. 2022, pp. 474–490.
- [73] Dmytro Kotovenko, Matthias Wright, Arthur Heimbrecht, and Björn Ommer. “Rethinking Style Transfer: From Pixels to Parameterized Brushstrokes”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Vol. 1. Los Alamitos, CA, USA: IEEE Computer Society, June 2021, pp. 12191–12200. DOI: [10.1109/CVPR46437.2021.01202](https://doi.org/10.1109/CVPR46437.2021.01202).
- [74] Chinmay E Kulkarni, Michael S Bernstein, and Scott R Klemmer. “PeerStudio: rapid peer feedback emphasizes revision and improves performance”. In: *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. ACM. 2015, pp. 75–84.
- [75] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. “Webzeitgeist: Design Mining the Web”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: Association for Computing Machinery, 2013, pp. 3083–3092. ISBN: 9781450318990. DOI: [10.1145/2470654.2466420](https://doi.org/10.1145/2470654.2466420).
- [76] Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. “Bricolage: Example-based retargeting for web design”. English (US). In: *CHI 2011 - 29th Annual CHI Conference on Human Factors in Computing Systems, Conference Proceedings and Extended Abstracts*. Conference on Human Factors in Computing Systems - Proceedings. United States: Association for Computing Machinery, 2011, pp. 2197–2206. ISBN: 9781450302289. DOI: [10.1145/1978942.1979262](https://doi.org/10.1145/1978942.1979262).
- [77] Jane L., Kevin Y. Zhai, Jose Echevarria, Ohad Fried, Pat Hanrahan, and James A. Landay. “Dynamic Guidance for Decluttering Photographic Compositions”. In: *The 34th Annual ACM Symposium on User Interface Software and Technology*. UIST ’21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 359–371. ISBN: 9781450386357. DOI: [10.1145/3472749.3474755](https://doi.org/10.1145/3472749.3474755).
- [78] Philippe Laban, Elicia Ye, Srujay Korlakunta, John Canny, and Marti Hearst. “NewsPod: Automatic and Interactive News Podcasts”. In: *27th International Conference on Intelligent User Interfaces*. IUI ’22. Helsinki, Finland: Association for Computing Machinery, 2022, pp. 691–706. ISBN: 9781450391443. DOI: [10.1145/3490099.3511147](https://doi.org/10.1145/3490099.3511147).

- [79] Sahiti Labhishetty, Bhavya, Kevin Pei, Assma Boughoula, and Chengxiang Zhai. “Web of Slides: Automatic Linking of Lecture Slides to Facilitate Navigation”. In: *Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale*. L@S '19. Chicago, IL, USA: Association for Computing Machinery, 2019. ISBN: 9781450368049. DOI: [10.1145/3330430.3333668](https://doi.org/10.1145/3330430.3333668).
- [80] Tomas Lawton, Francisco J Ibarrola, Dan Ventura, and Kazjon Grace. “Drawing with Reframer: Emergence and Control in Co-Creative AI”. In: *Proceedings of the 28th International Conference on Intelligent User Interfaces*. IUI '23. Sydney, NSW, Australia: Association for Computing Machinery, 2023, pp. 264–277. DOI: [10.1145/3581641.3584095](https://doi.org/10.1145/3581641.3584095).
- [81] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. “Designing with Interactive Example Galleries”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: Association for Computing Machinery, 2010, pp. 2257–2266. ISBN: 9781605589299. DOI: [10.1145/1753326.1753667](https://doi.org/10.1145/1753326.1753667).
- [82] Dar-Shyang Lee, Berna Erol, Jamey Graham, Jonathan J Hull, and Norihiko Murata. “Portable meeting recorder”. In: *Proceedings of the tenth ACM international conference on Multimedia*. ACM. 2002, pp. 493–502.
- [83] Hsin-Ying Lee, Lu Jiang, Irfan Essa, Phuong B Le, Haifeng Gong, Ming-Hsuan Yang, and Weilong Yang. “Neural design network: Graphic layout generation with constraints”. In: *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer. 2020, pp. 491–506.
- [84] Gang Li and Yang Li. “Spotlight: Mobile UI Understanding using Vision-Language Models with a Focus”. In: *arXiv preprint arXiv:2209.14927* (2022).
- [85] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. *LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators*. 2019. arXiv: [1901.06767](https://arxiv.org/abs/1901.06767) [cs.CV].
- [86] Jingyi Li, Joel Brandt, Radomir Mech, Maneesh Agrawala, and Jennifer Jacobs. “Supporting Visual Artists in Programming through Direct Inspection and Control of Program Execution”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020).
- [87] Jingyi Li, Sonia Hashim, and Jennifer Jacobs. “What We Can Learn From Visual Artists About Software Development”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: [10.1145/3411764.3445682](https://doi.org/10.1145/3411764.3445682).
- [88] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. “Differentiable Vector Graphics Rasterization for Editing and Learning”. In: *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39.6 (2020), 193:1–193:15.

- [89] Leonhard Lichtschlag, Thorsten Karrer, and Jan Borchers. “Fly: a tool to author planar presentations”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2009, pp. 547–556.
- [90] Rhema Linder, Nic Lupfer, Andruid Kerne, Andrew M Webb, Cameron Hill, Yin Qu, Kade Keith, Matthew Carrasco, and Elizabeth Kellogg. “Beyond slideware: How a free-form presentation medium stimulates free-form thinking in the classroom”. In: *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*. ACM. 2015, pp. 285–294.
- [91] Raphael Gontijo Lopes, David R Ha, Douglas Eck, and Jonathon Shlens. “A Learned Representation for Scalable Vector Graphics”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV) 1.1 (2019)*, pp. 7929–7938.
- [92] Kurt Luther, Jari-Lee Tolentino, Wei Wu, Amy Pavel, Brian P Bailey, Maneesh Agrawala, Björn Hartmann, and Steven P Dow. “Structuring, aggregating, and evaluating crowdsourced design critique”. In: *Proc. CSCW’15*. ACM. 2015, pp. 473–485.
- [93] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. “Towards layer-wise image vectorization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16314–16323.
- [94] Mikaël Mayer, Gustavo Soares, Maxim Grechkin, Vu Le, Mark Marron, Oleksandr Polozov, Rishabh Singh, Benjamin Zorn, and Sumit Gulwani. “User Interaction Models for Disambiguation in Programming by Example”. In: *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*. UIST ’15. Charlotte, NC, USA: Association for Computing Machinery, 2015, pp. 291–301. ISBN: 9781450337793. DOI: [10.1145/2807442.2807459](https://doi.org/10.1145/2807442.2807459).
- [95] Joseph F McCarthy, Elizabeth F Churchill, William G Griswold, Elizabeth Lawley, Melora Zaner, et al. “Digital backchannels in shared physical spaces: attention, intention and contention”. In: *Proceedings of the 2004 ACM conference on Computer supported cooperative work*. ACM. 2004, pp. 550–553.
- [96] Will McGrath, Daniel Drew, Jeremy Warner, Majeed Kazemitabaar, Mitchell Karchemsky, David Mellis, and Björn Hartmann. “Bifröst: Visualizing and Checking Behavior of Embedded Systems across Hardware and Software”. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST ’17. Québec City, QC, Canada: Association for Computing Machinery, 2017, pp. 299–310. ISBN: 9781450349819. DOI: [10.1145/3126594.3126658](https://doi.org/10.1145/3126594.3126658).
- [97] William McGrath, Jeremy Warner, Mitchell Karchemsky, Andrew Head, Daniel Drew, and Björn Hartmann. “WiFröst: Bridging the Information Gap for Debugging of Networked Embedded Systems”. In: *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. UIST ’18. Berlin, Germany: Association

- for Computing Machinery, 2018, pp. 447–455. ISBN: 9781450359481. DOI: [10.1145/3242587.3242668](https://doi.org/10.1145/3242587.3242668).
- [98] MDN contributors. *use tag - SVG: Scalable Vector Graphics — MDN*. <https://developer.mozilla.org/en-US/docs/Web/SVG/Element/use>. [Online; accessed 6-July-2023]. 2023.
- [99] Scott Minneman, Steve Harrison, Bill Janssen, Gordon Kurtenbach, Thomas Moran, Ian Smith, and Bill van Melle. “A confederation of tools for capturing and accessing collaborative activity”. In: *ACM Multimedia*. Vol. 95. 1995.
- [100] Kevin Mullet and Darrell Sano. “Designing visual interfaces”. In: *ACM SIGCHI Bulletin* 28.2 (1996), pp. 82–83.
- [101] Gabriel Murray, Steve Renals, and Jean Carletta. “Extractive summarization of meeting recordings.” In: (2005).
- [102] *Tips for making effective Powerpoint presentations*. 2017. URL: <http://www.ncsl.org/legislators-staff/legislative-staff/legislative-staff-coordinating-committee/tips-for-making-effective-powerpoint-presentations.aspx>.
- [103] Ani Nenkova, Kathleen McKeown, et al. “Automatic summarization”. In: *Foundations and Trends® in Information Retrieval* 5.2–3 (2011), pp. 103–233.
- [104] Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. “Summarizing Sporting Events Using Twitter”. In: *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*. IUI ’12. Lisbon, Portugal: Association for Computing Machinery, 2012, pp. 189–198. ISBN: 9781450310482. DOI: [10.1145/2166966.2166999](https://doi.org/10.1145/2166966.2166999).
- [105] Donald A. Norman and Stephen W. Draper. “User Centered System Design: New Perspectives on Human-Computer Interaction”. In: 1988.
- [106] Peter O’Donovan, Aseem Agarwala, and Aaron Hertzmann. “DesignScape: Design with Interactive Layout Suggestions”. In: *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 1221–1224.
- [107] Jasper O’Leary, Holger Winnemöller, Wilmot Li, Mira Dontcheva, and Morgan Dixon. “Charrette: Supporting In-Person Discussions around Iterations in User Interface Design”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM. 2018, p. 535.
- [108] Xavier Ochoa, Federico Domínguez, Bruno Guamán, Ricardo Maya, Gabriel Falcones, and Jaime Castells. “The RAP System: Automatic Feedback of Oral Presentation Skills Using Multimodal Analysis and Low-Cost Sensors”. In: *Proceedings of the 8th International Conference on Learning Analytics and Knowledge*. LAK ’18. Sydney, New South Wales, Australia: Association for Computing Machinery, 2018, pp. 360–364. ISBN: 9781450364003. DOI: [10.1145/3170358.3170406](https://doi.org/10.1145/3170358.3170406).

- [109] Lora Oehlberg, Kyu Simm, Jasmine Jones, Alice Agogino, and Björn Hartmann. “Showing is sharing: building shared understanding in human-centered design teams with Dazzle”. In: *Proceedings of the Designing Interactive Systems Conference*. ACM. 2012, pp. 669–678.
- [110] Thierry Olive. “Working memory in writing: Empirical evidence from the dual-task technique”. In: *European Psychologist* 9.1 (2004), pp. 32–42.
- [111] Kiemute Oyibo and Julita Vassileva. “The effect of layout and colour temperature on the perception of tourism websites for mobile devices”. In: *Multimodal technologies and interaction* 4.1 (2020), p. 8.
- [112] Srishti Palani, David Ledo, George Fitzmaurice, and Fraser Anderson. “”I Don’t Want to Feel like I’m Working in a 1960s Factory”: The Practitioner Perspective on Creativity Support Tool Adoption”. In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI ’22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. DOI: [10.1145/3491102.3501933](https://doi.org/10.1145/3491102.3501933).
- [113] Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. “VidCrit: Video-based Asynchronous Video Review”. In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. ACM. 2016, pp. 517–528.
- [114] Yi-Hao Peng, Jeffrey P Bigham, and Amy Pavel. “Slidecho: Flexible Non-Visual Exploration of Presentation Videos”. In: *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*. ASSETS ’21. Virtual Event, USA: Association for Computing Machinery, 2021. ISBN: 9781450383066. DOI: [10.1145/3441852.3471234](https://doi.org/10.1145/3441852.3471234).
- [115] Yi-Hao Peng, JiWoong Jang, Jeffrey P Bigham, and Amy Pavel. “Say It All: Feedback for Improving Non-Visual Presentation Accessibility”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI ’21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: [10.1145/3411764.3445572](https://doi.org/10.1145/3411764.3445572).
- [116] Annie Piolat, Thierry Olive, and Ronald T Kellogg. “Cognitive effort during note taking”. In: *Applied cognitive psychology* 19.3 (2005), pp. 291–312.
- [117] Alexander Pohl, Vera Gehlen-Baum, and François Bry. “Introducing Backstage—a digital backchannel for large class lectures”. In: *Interactive Technology and Smart Education* 8.3 (2011), pp. 186–200.
- [118] Oleksandr Polozov and Sumit Gulwani. “FlashMeta: A Framework for Inductive Program Synthesis”. In: *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications*. OOPSLA 2015. Pittsburgh, PA, USA: Association for Computing Machinery, 2015, pp. 107–126. ISBN: 9781450336895. DOI: [10.1145/2814270.2814310](https://doi.org/10.1145/2814270.2814310).
- [119] Helen Purchase. “Which aesthetic has the greatest effect on human understanding?” In: *Graph Drawing*. Vol. 97. 1997, pp. 248–261.

- [120] Jonathan Ragan-Kelley, Charlie Kilpatrick, Brian W Smith, Doug Epps, Paul Green, Christophe Hery, and Frédo Durand. “The lightspeed automatic interactive lighting preview system”. In: *ACM SIGGRAPH 2007 papers*. New York, NY, USA: Association for Computing Machinery, 2007, 25–es.
- [121] Matt Ratto, R Benjamin Shapiro, Tan Minh Truong, and William G Griswold. “The activeclass project: Experiments in encouraging classroom participation”. In: *Designing for change in networked learning environments*. Springer, 2003, pp. 477–486.
- [122] Eric Rawn. “Exploring Power In Creativity Support Tools”. In: *Proceedings of the 35st Annual ACM Symposium on User Interface Software and Technology*. UIST ’23. San Fransisco, California, USA: Association for Computing Machinery, 2023, pp. 447–455. ISBN: 9781450359481.
- [123] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J. Mitra. “Im2Vec: Synthesizing Vector Graphics Without Vector Supervision”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 7342–7351.
- [124] Pradyumna Reddy, Paul Guerrero, Matt Fisher, Wilmot Li, and Niloy Jyoti Mitra. “Discovering pattern structure using differentiable compositing”. In: *ACM Transactions on Graphics (TOG)* 39 (2020), pp. 1–15.
- [125] Jun Rekimoto, Yuji Ayatsuka, Hitoraka Uoi, and Toshifumi Arai. “Adding another communication channel to reality: an experience with a chat-augmented conference”. In: *Conference on Human Factors in Computing Systems: CHI 98 conference summary on Human factors in computing systems*. Vol. 18. 23. 1998, pp. 271–272.
- [126] Garr Reynolds. *Presentation Zen: Simple ideas on presentation design and delivery*. New Riders, 2011.
- [127] Quentin Roy, Futian Zhang, and Daniel Vogel. “Automation Accuracy Is Good, but High Controllability May Be Better”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI ’19. Glasgow, Scotland Uk: Association for Computing Machinery, 2019, pp. 1–8. ISBN: 9781450359702. DOI: [10.1145/3290605.3300750](https://doi.org/10.1145/3290605.3300750).
- [128] Timothy Samara. *Making and breaking the grid, updated and expanded: A graphic design layout workshop*. Quarry Books Editions, 2017.
- [129] Othman Sbai, Camille Couprie, and Mathieu Aubry. “Vector Image Generation by Learning Parametric Layer Decomposition”. In: *ArXiv abs/1812.05484* (2018).
- [130] Eldon Schoop, Forrest Huang, and Björn Hartmann. “SCRAM: Simple Checks for Realtime Analysis of Model Training for Non-Expert ML Programmers”. In: *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI EA ’20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–10. ISBN: 9781450368193. DOI: [10.1145/3334480.3382879](https://doi.org/10.1145/3334480.3382879).

- [131] Eldon Schoop, Xin Zhou, Gang Li, Zhourong Chen, Björn Hartmann, and Yang Li. “Predicting and Explaining Mobile UI Tappability with Vision Modeling and Saliency Analysis”. In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI ’22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. DOI: [10.1145/3491102.3517497](https://doi.org/10.1145/3491102.3517497).
- [132] Chanon Seel-audom, Wassana Naiyapo, and Varin Chouvatut. “A search for geometric-shape objects in a vector image: Scalable Vector Graphics (SVG) file format”. In: *2017 9th International Conference on Knowledge and Smart Technology (KST) 1.1* (2017), pp. 305–310.
- [133] Athar Sefid, Prasenjit Mitra, and Lee Giles. “SlideGen: An Abstractive Section-Based Slide Generator for Scholarly Documents”. In: *Proceedings of the 21st ACM Symposium on Document Engineering*. DocEng ’21. Limerick, Ireland: Association for Computing Machinery, 2021. ISBN: 9781450385961. DOI: [10.1145/3469096.3474939](https://doi.org/10.1145/3469096.3474939).
- [134] Amy Shannon, Jessica Hammer, Hassler Thurston, Natalie Diehl, and Steven Dow. “PeerPresents: A web-based system for in-class peer feedback during student presentations”. In: *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*. ACM. ACM, 2016, pp. 447–458.
- [135] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. “Neural Abstractive Text Summarization with Sequence-to-Sequence Models”. In: *ACM/IMS Trans. Data Sci.* 2.1 (2021). ISSN: 2691-1922. DOI: [10.1145/3419106](https://doi.org/10.1145/3419106).
- [136] Hijung V. Shin, Jeremy Warner, Björn Hartmann, Celso Gomes, Holger Winnemöller, and Wilmot Li. “Multi-level Correspondence via Graph Kernels for Editing Vector Graphics Designs”. In: *Proceedings of Graphics Interface 2021*. GI 2021. Virtual Event: Canadian Information Processing Society, 2021, pp. 97–107. ISBN: 978-0-9947868-6-9. DOI: [10.20380/GI2021.12](https://doi.org/10.20380/GI2021.12).
- [137] Steven J. Simske and Rafael Lins. “Automatic Text Summarization and Classification”. In: *Proceedings of the ACM Symposium on Document Engineering 2018*. DocEng ’18. Halifax, NS, Canada: Association for Computing Machinery, 2018. ISBN: 9781450357692. DOI: [10.1145/3209280.3232791](https://doi.org/10.1145/3209280.3232791).
- [138] Ryan Spicer, Yu-Ru Lin, Aisling Kelliher, and Hari Sundaram. “NextSlidePlease: Authoring and delivering agile multimedia presentations”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 8.4 (2012), p. 53.
- [139] Minhyang (Mia) Suh, Emily Youngblom, Michael Terry, and Carrie J Cai. “AI as Social Glue: Uncovering the Roles of Deep Generative AI during Social Music Composition”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI ’21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: [10.1145/3411764.3445219](https://doi.org/10.1145/3411764.3445219).

- [140] Amanda Swearngin and Yang Li. “Modeling mobile interface tappability using crowdsourcing and deep learning”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–11.
- [141] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomir Mvech. “Learning Design Patterns with Bayesian Grammar Induction”. In: *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. UIST ’12. Cambridge, Massachusetts, USA: Association for Computing Machinery, 2012, pp. 63–74. ISBN: 9781450315807. DOI: [10.1145/2380116.2380127](https://doi.org/10.1145/2380116.2380127).
- [142] Edric Tam and David B. Dunson. “Multiscale Graph Comparison via the Embedded Laplacian Distance”. In: *ArXiv abs/2201.12064* (2022).
- [143] TED Staff. *10 tips for better slide decks*. <https://blog.ted.com/10-tips-for-better-slide-decks/>. 2014.
- [144] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. “Face2Face: Real-Time Face Capture and Reenactment of RGB Videos”. In: *Commun. ACM* 62.1 (Dec. 2018), pp. 96–104. ISSN: 0001-0782. DOI: [10.1145/3292039](https://doi.org/10.1145/3292039).
- [145] Rundong Tian, Sarah Stermann, Ethan Chiou, Jeremy Warner, and Eric Paulos. “Match-Sticks: Woodworking through Improvisational Digital Fabrication”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–12. ISBN: 9781450356206. DOI: [10.1145/3173574.3173723](https://doi.org/10.1145/3173574.3173723).
- [146] David Tinapple, Loren Olson, and John Sadauskas. “CritViz: Web-based software supporting peer critique in large creative classrooms”. In: *Bulletin of the IEEE Technical Committee on Learning Technology* 15.1 (2013), p. 29.
- [147] Josh Urban Davis, Fraser Anderson, Merten Stroetzel, Tovi Grossman, and George Fitzmaurice. “Designing Co-Creative AI for Virtual Environments”. In: *Creativity and Cognition*. C&C ’21. Virtual Event, Italy: Association for Computing Machinery, 2021. ISBN: 9781450383769. DOI: [10.1145/3450741.3465260](https://doi.org/10.1145/3450741.3465260).
- [148] Mariya I. Vasileva, Bryan A. Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Alexander Forsyth. “Learning Type-Aware Embeddings for Fashion Compatibility”. In: *ArXiv abs/1803.09196* (2018).
- [149] Jo Vermeulen, Kris Luyten, Elise van den Hoven, and Karin Coninx. “Crossing the Bridge over Norman’s Gulf of Execution: Revealing Feedforward’s True Identity”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’13. Paris, France: Association for Computing Machinery, 2013, pp. 1931–1940. ISBN: 9781450318990. DOI: [10.1145/2470654.2466255](https://doi.org/10.1145/2470654.2466255).
- [150] W3C SVG Working Group. *Scalable Vector Graphics (SVG) 2*. <https://www.w3.org/TR/SVG2/>. [Online; accessed 6-July-2023]. 2018.

- [151] Chenglong Wang, Yu Feng, Rastislav Bodik, Isil Dillig, Alvin Cheung, and Amy J Ko. “Falx: Synthesis-Powered Visualization Authoring”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: [10.1145/3411764.3445249](https://doi.org/10.1145/3411764.3445249).
- [152] Jeremy Warner, Kyu Won Kim, and Björn Hartmann. “Interactive Flexible Style Transfer for Vector Graphics”. In: *Proceedings of the 2023 ACM Symposium on User Interface Software and Technology*. UIST '23. San Francisco, California, USA: Association for Computing Machinery, 2023.
- [153] Jeremy Warner, Amy Pavel, Tonya Nguyen, Maneesh Agrawala, and Björn Hartmann. “SlideSpecs: Automatic and Interactive Presentation Feedback Collation”. In: *Proceedings of the 2023 ACM Conference on Intelligent User Interfaces*. IUI '23. Sydney, Australia: Association for Computing Machinery, 2023. DOI: [10.1145/3581641.3584035](https://doi.org/10.1145/3581641.3584035).
- [154] Jeremy Warner, Shuyao Zhou, and Björn Hartmann. “Interactively Optimizing Layout Transfer for Vector Graphics”. In: *Proceedings of the 2023 International Conference on Machine Learning AI/HCI Workshop*. ICML '23. Hawaii, USA: Association for Computing Machinery, 2023.
- [155] Andy Warr and Eamonn O’Neill. “Understanding design as a social creative process”. In: *Proceedings of the 5th conference on Creativity & cognition*. ACM. 2005, pp. 118–127.
- [156] Pierre Wellner, Mike Flynn, and Maël Guillemot. “Browsing recorded meetings with Ferret”. In: *International Workshop on Machine Learning for Multimodal Interaction*. Springer. 2004, pp. 12–21.
- [157] Steve Whittaker, Patrick Hyland, and Myrtle Wiley. “Filochat: Handwritten notes provide access to recorded conversations”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 1994, pp. 271–277.
- [158] Steve Whittaker, Rachel Laban, and Simon Tucker. “Analysing meeting records: An ethnographic study and technological implications”. In: *International Workshop on Machine Learning for Multimodal Interaction*. Springer. 2005, pp. 101–113.
- [159] Wesley Willett, Jeffrey Heer, Joseph Hellerstein, and Maneesh Agrawala. “CommentSpace: structured support for collaborative visual analysis”. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM. 2011, pp. 3131–3140.
- [160] Hsiang-Yun Wu, Benjamin Niedermann, Shigeo Takahashi, Maxwell J Roberts, and Martin Nöllenburg. “A survey on transit map layout—from design, machine, and human perspectives”. In: *Computer Graphics Forum*. Vol. 39. 3. Wiley Online Library. 2020, pp. 619–646.

- [161] Jason Wu, Amanda Swearngin, Xiaoyi Zhang, Jeffrey Nichols, and Jeffrey P Bigham. “Screen Correspondence: Mapping Interchangeable Elements between UIs”. In: *arXiv preprint arXiv:2301.08372* (2023).
- [162] Jason Wu, Siyan Wang, Siman Shen, Yi-Hao Peng, Jeffrey Nichols, and Jeffrey P Bigham. “WebUI: A Dataset for Enhancing Visual UI Understanding with Web Semantics”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 2023, pp. 1–14.
- [163] Jason Wu, Xiaoyi Zhang, Jeff Nichols, and Jeffrey P Bigham. “Screen Parsing: Towards Reverse Engineering of UI Models from Screenshots”. In: *The 34th Annual ACM Symposium on User Interface Software and Technology*. UIST ’21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 470–483. ISBN: 9781450386357. DOI: [10.1145/3472749.3474763](https://doi.org/10.1145/3472749.3474763).
- [164] Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. “Object-Oriented Drawing”. In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. CHI ’16. San Jose, California, USA: Association for Computing Machinery, 2016, pp. 4610–4621. ISBN: 9781450333627. DOI: [10.1145/2858036.2858075](https://doi.org/10.1145/2858036.2858075).
- [165] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. “DataInk: Direct and Creative Data-Oriented Drawing”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI ’18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–13. ISBN: 9781450356206. DOI: [10.1145/3173574.3173797](https://doi.org/10.1145/3173574.3173797).
- [166] Zhenmin Yang, Yonghao Dong, Jiange Deng, Baocheng Sha, and Tao Xu. “Research on Automatic News Text Summarization Technology Based on GPT2 Model”. In: *2021 3rd International Conference on Artificial Intelligence and Advanced Manufacturing*. AIAM2021. Manchester, United Kingdom: Association for Computing Machinery, 2021, pp. 418–423. ISBN: 9781450385046. DOI: [10.1145/3495018.3495091](https://doi.org/10.1145/3495018.3495091).
- [167] Katherine Ye, Wode Ni, Max Krieger, Dor Ma’ayan, Jenna Wise, Jonathan Aldrich, Joshua Sunshine, and Keenan Crane. “Penrose: from mathematical notation to beautiful diagrams”. In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 144–1.
- [168] Katherine Q. Ye, Wode Ni, Max Krieger, Dor Ma’ayan, Jenna Wise, Jonathan Aldrich, Joshua Sunshine, and Keenan Crane. “Penrose: from mathematical notation to beautiful diagrams”. In: *ACM Trans. Graph.* 39 (2020), p. 144.
- [169] Nur Yildirim, Alex Kass, Teresa Tung, Connor Upton, Donnacha Costello, Robert Giusti, Sinem Lacin, Sara Lovic, James M O’Neill, Rudi O’Reilly Meehan, Eoin Ó Loideáin, Azzurra Pini, Medb Corcoran, Jeremiah Hayes, Diarmuid J Cahalane, Gaurav Shivhare, Luigi Castoro, Giovanni Caruso, Changhoon Oh, James McCann, Jodi Forlizzi, and John Zimmerman. “How Experienced Designers of Enterprise Applications Engage AI as a Design Material”. In: *Proceedings of the 2022 CHI Conference*

- on Human Factors in Computing Systems*. CHI '22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. DOI: [10.1145/3491102.3517491](https://doi.org/10.1145/3491102.3517491).
- [170] Dongwook Yoon, Nicholas Chen, François Guimbretière, and Abigail Sellen. “RichReview: blending ink, speech, and gesture to support collaborative document review”. In: *Proc. UIST'14*. ACM. ACM, 2014, pp. 481–490.
- [171] Dongwook Yoon, Nicholas Chen, Bernie Randles, Amy Cheatle, Corinna E Löckenhoff, Steven J Jackson, Abigail Sellen, and François Guimbretière. “RichReview++: Deployment of a Collaborative Multi-modal Annotation System for Instructor Feedback and Peer Discussion”. In: *Proc. CSCW'16*. ACM. ACM, 2016, pp. 195–205.
- [172] Amy X Zhang and Justin Cranshaw. “Making sense of group chat through collaborative tagging and summarization”. In: *Proceedings of the ACM on Human-Computer Interaction* 2.CSCW (2018), p. 196.
- [173] Amy X. Zhang, Lea Verou, and David Karger. “Wikum: Bridging Discussion Forums and Wikis Using Recursive Summarization”. In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. CSCW '17. Portland, Oregon, USA: ACM, 2017, pp. 2082–2096. ISBN: 978-1-4503-4335-0. DOI: [10.1145/2998181.2998235](https://doi.org/10.1145/2998181.2998235).
- [174] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. “Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels”. In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: [10.1145/3411764.3445186](https://doi.org/10.1145/3411764.3445186).
- [175] Nanxuan Zhao, Ying Cao, and Rynson WH Lau. “What characterizes personalities of graphic designs?” In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), pp. 1–15.
- [176] Nanxuan Zhao, Quanlong Zheng, Jing Liao, Ying Cao, Hanspeter Pfister, and Rynson W. H. Lau. “Selective Region-Based Photo Color Adjustment for Graphic Designs”. In: *ACM Trans. Graph.* 40.2 (Apr. 2021). ISSN: 0730-0301. DOI: [10.1145/3447647](https://doi.org/10.1145/3447647).