AI Steering Interfaces for Design Exploration

David Chuan-En Lin

CMU-HCII-26-110
July 2026

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:

Nikolas Martelaro, Chair, Carnegie Mellon University
Aniket Kittur, Carnegie Mellon University
David Lindlbauer, Carnegie Mellon University
Michael Terry, Google DeepMind

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Human-Computer Interaction

© David Chuan-En Lin, 2026

Keywords: AI Steering Interfaces, Design Exploration, Generative AI, Human-AI Interaction, Creativity Support Tools

Abstract

Recent advancements in AI have made it possible for designers to produce images, videos, and 3D content from natural language. However, text prompts require designers to articulate exactly what they want before they have seen what is possible. Designers' best tools often work the opposite way: sketches, moodboards, and CAD let designers work directly on the design itself, drawing, comparing, and adjusting as the artifact takes shape. I argue that AI as a design material should work the same way. Inside, AI models hold a rich understanding of style, structure, and semantic relationships that go well beyond what a text prompt can reach, and internal model controls that designers should be able to discover and manipulate. This gap matters most in design exploration, where designers need to move through wide possibility spaces, combine and remix ideas, and rapidly iterate.

Effective AI steering interfaces for design exploration come from aligning three things: the tasks designers want to do, the control mechanisms a model exposes, and the interface representations that put those mechanisms in the designer's hands. This thesis refers to this alignment as the Human-AI-Interface (HAI) Triangle. Alignment is operationalized through three vertex conditions: *controllability* (the AI mechanism exposes handles that a designer can act on), *affordance* (the controls match the moves the task requires), and *intuitiveness* (the representation draws on conventions designers already work with). The framework is grounded in four systems, each addressing a distinct design task. VideoMap addresses the *navigate* task: it projects latent video embeddings onto a 2D map that editors can explore to find visual and semantic connections across footage. Jigsaw addresses the *compose* task: it represents pre-trained models as puzzle pieces that snap together, so designers can build multimodal AI pipelines without writing code. Inkspire addresses the *iterate* task: it pairs iterative sketching with AI-generated scaffolds, allowing designers to refine through strokes rather than through prompt edits. GENErator addresses the *remix* task: it lets designers mix and evolve style-specific AI models in a shared semantic space, surfacing aesthetic directions they could not have specified up front. Across user evaluations, participants navigated, composed, iterated on, and remixed AI capabilities through intuitive and controllable interactions, accomplishing tasks they could not with conventional interfaces.

Table of Contents

List of Figures . . . . . **vii**

List of Tables . . . . . **xiv**

1 **Introduction** **1**

1.1 Thesis Statement **2**

1.2 Thesis Organization **2**

2 **Related Work** **4**

2.1 Direct Manipulation and Interaction Paradigms **4**

2.2 Human-AI Interaction **6**

2.3 Creativity Support Tools **7**

2.4 How Designers Explore **8**

2.5 Generative AI and Controllable Generation **10**

2.6 Positioning This Thesis **11**

3 **The Human-AI-Interface Triangle** **13**

3.1 Human, AI, and Interface **13**

3.2 Misalignment Patterns **16**

3.3 Four Design Tasks **19**

4 **VideoMap: Maps as an Interface for Navigating the Latent Space** . . . **22**

4.1 Introduction . . . 22
4.2 Related Work . . . 24
4.3 Principles of VideoMap . . . 27
4.4 Implementation . . . 29
4.5 User Study . . . 36
4.6 Discussion . . . 43
4.7 Extended Applications . . . 45
4.8 Future Work . . . 48
4.9 Conclusion . . . 48
5 **Jigsaw: Puzzles as an Interface for Composing Model Chains** . . . 50
5.1 Introduction . . . 50
5.2 Related Work . . . 53
5.3 Formative Study . . . 56
5.4 Jigsaw System . . . 58
5.5 User Study . . . 65
5.6 Limitations and Future Work . . . 70
5.7 Conclusion . . . 71
6 **Inkspire: Sketching as an Interface for Iterating with Structural Guidance** . . . 72
6.1 Introduction . . . 72
6.2 Related Work . . . 75
6.3 Formative Session with Design Professionals . . . 78
6.4 Inkspire . . . 79

6.5 User Study . . . 86
6.6 Results . . . 89
6.7 Discussion . . . 95
6.8 Conclusion . . . 99
7 **GENErator: Evolution as an Interface for Mixing Style Models** . . . 101
7.1 Introduction . . . 101
7.2 Related Work . . . 104
7.3 GENErator . . . 106
7.4 User Study . . . 114
7.5 Results . . . 116
7.6 Discussion . . . 120
7.7 Conclusion . . . 121
8 **Discussion and Conclusion** . . . 123
8.1 Research Contributions . . . 123
8.2 Design Guidelines for AI Steering Interfaces . . . 124
8.3 Future Directions . . . 129
8.4 Conclusion . . . 134
A.1 List of Implemented Models . . . 135
A.2 Participant Model Mosaics . . . 136
A.3 Semantic Search and Assembly Assistant Examples . . . 137
B.1 Participant Usage Logs . . . 138
B.2 Baseline Interface . . . 139
B.3 Analogical Inspirations by Participant . . . 140

C.1 Technical Validation 141

C.2 Additional Figures 144

Bibliography 148

List of Figures

Figure 1.1 The Human-AI-Interface Triangle: three vertices that must align for effective AI steering. Chapter **3** develops the framework. **3**

Figure 3.1 The Human-AI-Interface Triangle. Three vertices (Human Task, AI Mechanism, Interface Representation) align for effective AI steering, each with its own alignment condition: *affordance* (Human), *controllability* (AI), and *intuitiveness* (Interface). **15**

Figure 4.1 VideoMap is a proof-of-concept video editing interface that operates on video frames projected onto a latent space, enabling users to visually uncover patterns and relationships. We introduce map-inspired navigational elements (node, path, district, landmark) to support users in navigating the map. **27**

Figure 4.2 The pipeline for creating the `shape` lens. We pass video frames through DeepLabv3, an image masking model, to extract shape-related vectors (a). We then apply t-SNE to reduce the vectors to two dimensions (b) and visually rearrange the vectors by video to make the videos more skimmable (c). **30**

Figure 4.3 The editor can search for videos using text prompts (e.g., `torii gates`). . **31**

Figure 4.4 VideoMap's Project Panel component. The figure shows a collection of videos organized under the `semantic` lens. The editor can play through the videos by scrubbing the landmarks from left to right (a). Example semantic clusters of videos are shown in (b) for several videos containing streets and buildings and (c) for concert videos. **33**

Figure 4.5 VideoMap's Paths Explorer component. The editor can select a video frame to display ten video transition suggestions based on the selected lens (a). For example, a video frame with a similar color composition is recommended under the `color` lens (b). **34**

Figure 4.6 VideoMap's Route Planner component. The editor can select several video clips to automatically generate a rough cut video based on the selected lens (a). For example, the editor can create a video of a person running through various backgrounds under the `shape` lens (b). Route Planner automatically finds the optimal video transitions (c1 and c2). . . . **35**

Figure 4.7 VideoMap's Project Panel can be extended to create summary videos. We automatically create "semantic districts" that approximately represent the main activities of a video using k-means clustering under the `semantic` lens. The editor can select several landmarks to specify the activities to include in the summary video. Selections are highlighted with red borders (a) and displayed as a storyboard (b). **44**

Figure 4.8 VideoMap's Paths Explorer can be extended to create highlight videos. The editor can upload a photograph (i.e., a custom landmark) depicting an activity (e.g., skydiving) (a). Our key insight is that photographs taken by photographers tend to capture the most highlight-worthy moments of an activity (e.g., when the skydiver jumps out of the aircraft). We then generate a highlight video using near neighbor video frames to the custom landmark in the `semantic` space (b). **46**

Figure 4.9 VideoMap's Route Planner can be extended to edit videos using text. The editor can describe a desired video using descriptive sentences, like writing a story (a). We then match each sentence to the closest video clip in the `semantic` space and generate a video by finding the shortest route along the clips (b). **47**

Figure 5.1 Jigsaw is a prototyping system that lets designers generate creative content with AI foundation models represented as puzzle pieces. Designers can first search for model capabilities via the Catalog Panel (a). Designers can drag corresponding puzzle pieces onto the Assembly Panel (b) and combine different AI capabilities across varying modalities into a chain by assembling compatible pieces. Designers can specify inputs and observe the intermediate results of a chain via the Input and Output Panels (c). Designers can ask the Assembly Assistant (d) to recommend a chain of AI models to accomplish a task. **51**

Figure 5.2 Users can search for model pieces by describing their task in the semantic search bar (a). Users can hover over a model piece to view a description of its capability, typical runtime, and an example input and output (b). . . **59**

Figure 5.3 The *translation* glue piece converts a piece of text into a prompt format suitable for text-to-x generation models (a). The *ideation* glue piece generates an idea for a design task (b). **60**

Figure 5.4 Users can drag puzzle pieces from the Catalog Panel onto the Assembly Panel (a), select pieces on the Assembly Panel by clicking on them (b), and remove pieces by dragging them to the trash bin or pressing the delete key (c). Users can duplicate pieces, and undo and redo actions using hotkeys (d-f). 61

Figure 5.5 When the user drags a puzzle piece close to another compatible piece, Jigsaw displays a semi-transparent preview of the potential connection. If the user releases the puzzle piece, it will snap into place (a). Conversely, if the user attempts to connect a puzzle piece to an incompatible piece, the new piece will be repelled, ensuring that users do not force a fit (b). Users can move multiple puzzle pieces simultaneously (c). 61

Figure 5.6 Text inputs can be directly typed into the Input Panel (a). Image, video, 3D, and audio inputs can be uploaded either by drag-and-drop or the file browser (b). Sketch inputs can be drawn (c). Text outputs can be viewed and copied by the user (d). Image, video, 3D, audio, and sketch outputs can be viewed in their respective media viewers and downloaded by the user (e-i). 62

Figure 5.7 An example model mosaic for interior design. The designer can use Jigsaw to ideate a design concept from client material (a), design the 2D and 3D mockups (b), and add music to enhance the presentation (c). . . . 64

Figure 5.8 Model mosaic by P2, an illustrator, to create an audio-visual story. P2 uses Jigsaw to create an illustration based on a text description and a reference sketch (a), generate narrations through a cloned voice (b), and generate accompanying sound effects (c). 67

Figure 5.9 Model mosaic by P10, a game designer, to create a video game character. P10 uses Jigsaw to create a character concept and preview the character's visuals (a). P10 then specifies a pose for the character and generates the character's visuals in the specified pose (b). P10 then generates a line for the character to say and animates the character to deliver the line. 68

Figure 6.1 The Inkspire interface. The designer may use the Analogical Panel (a) to ideate analogical inspirations for abstract concepts (e.g., "protective car" → "tortoise car"). The designer may sketch on the Sketching Panel (b) to iteratively guide AI design generations. For each iteration, we display a sketch scaffolding under the canvas. This scaffolding is created through abstracting AI designs into lower fidelity. Finally, the designer may view the history of iterations on the Evolution Panel (c). 80

Figure 6.2 Sketch2Design pipeline, including (a) inspiration generation with LLMs, (b) sketch-guided design generation, and (c) foreground extraction. 82

Figure 6.3 Comparison of our Design2Sketch method with potential alternative methods, such as edge detection, manga line extraction models, and models trained explicitly on pairs of sketches and images. 84

Figure 6.4 Design2Sketch pipeline, including (a) semantic segmentation, (b) soft edge extraction, and (c) computing an intersection. 85

Figure 6.5 Results on creativity measured with the Creativity Support Index (CSI) [1] (7-point Likert scale, higher is better). 87

Figure 6.6 Example designs created by participants using Inkspire for the design tasks of designing a fluid chair and a serene lamp. The final participant-generated sketch is shown on the top, and the generated T2I image is shown on the bottom, along with the selected analogy word chosen by the participant. 88

Figure 6.7 Results on human-AI collaboration measured with Human-Machine Collaboration Questions from [2] (7-point Likert scale, higher is better). . . 89

Figure 6.8 Example user iterations with Inkspire (top section) vs. the baseline condition (bottom section). For each section, the top row shows the prompts, the middle row shows the sketches and scaffolds, and the bottom row shows the generated designs. Using Inkspire, users create diverse designs via analogies. In addition, users start with a single sketch line and continuously build on their sketch, with the guidance of scaffolding. In contrast, using the baseline, users typically draw a full sketch and update their prompt with incremental modifications. This may lead to a higher degree of fixation and a smaller design space explored. 91

Figure 6.9 Results on sketching statistics, including total sketching time, sketching frequency, and time between sketches. 92

Figure 6.10 Example user interaction log. We observe that, using Inkspire, the user started by ideating several analogical inspirations (thought bubble). Next, they performed sketching in a highly iterative manner (pencil+sparkles). Overall, the user generated a significant amount of new designs (sparkles). In contrast, using the baseline condition, the user sketched in large stretches with infrequent new design generations. We see that this pattern generally holds true across participants (please see Figure B.1 for data on all participants). 93

Figure 6.11 Results on design quality (left) and usage experience (right) (7-point Likert scale, higher is better). 95

Figure 6.12 Given a scaffold, the user may choose to build on it (by tracing it) or steer the AI towards a different direction. 97

Figure 7.1 The 100-gene style space projected via PCA, with convex hulls drawn around each of the six categories. Semantically related styles cluster without explicit supervision: e.g., architectural styles (upper left), luminous effects (top), reflective materials (center), and tactile textures (right). 108

Figure 7.2 Semantic search in the gene library. Querying "fluffy" ranks all genes by semantic similarity, surfacing Plush (67%), Wooly (55%), and Cotton Candy (42%). 109

Figure 7.3 Complement vs. contrast mutation across six products. Each row shows a source gene (center) with its complement (left) and contrast (right). . . 110

Figure 7.4 Natural selection evolving a population of three styles toward "luxurious" over three generations. In each generation, the system scores every candidate among the population by similarity to the target, selects the fittest, and mutates them using semantically related genes to produce the next generation. At the final generation, the winner is selected (gold crown). 113

Figure 7.5 Example iteration sequence for a handbag. Starting from Plastic + Plush, the designer adds or removes genes across six iterations, each building on the previous offspring. 113

Figure 7.6 Likert results across all subjective measures. *Left:* Creativity and exploration (adapted from the Creativity Support Index [3]). GENErator rated significantly higher on ease of exploration, creativity, enjoyment, and satisfaction. *Right:* Style-specific design outcomes. GENErator rated significantly higher on diversity, unexpected discovery, and understanding of style relationships. Error bars show standard deviation. 116

Figure 7.7 Per-participant style space coverage in PCA-projected CLIP space. Convex hulls show GENErator (teal) vs. baseline (orange). The faded gray convex hull shows the complete style space (the same one as Figure 7.1). GENErator participants generally covered more of the style space. . 118

Figure 7.8 Interaction event sequence for P12. The baseline (bottom) is dominated by weight adjustments (gray). GENErator (top) shows crossover (green), mutation (orange), evolution (purple), and search (blue), with more bookmarks (stars). See Figure C.6 for all 12 participants. 119

Figure 8.1 Three future directions, each extending one vertex of the HAI Triangle. Engineering steerable mechanisms extends the AI vertex, designing models for controllability from the start. Designing new representations extends the Interface vertex, creating surfaces for emerging AI mechanisms. Domains beyond design exploration extends the Human vertex, testing which representations transfer to new tasks. **129**

Figure 8.2 Three future directions, each along an edge of the triangle. Generative interfaces extend the AI → Interface edge, with the AI producing the steering surface itself given a task and an available mechanism. AI steering interfaces extend the Human → AI edge, continuing the thesis's central direction with new tasks, mechanisms, and contexts. Adaptive interfaces extend the Interface → Human edge, with the representation shifting to each user's abilities and preferences. **132**

Figure B.1 Complete participant usage logs when using Inkspire vs. the baseline. Please see Figure **6.10** for an annotated illustrative example. **138**

Figure B.2 The baseline interface adopts a similar layout as Inkspire, but without analogical inspirations, scaffolding sketch underlays, and stroke-by-stroke generations. The participant would typically manually specify a prompt and draw a complete sketch for the T2I model – a standard workflow of how designers work with T2I models with additional sketch control [**4**]. . **139**

Figure C.1 Natural selection convergence across 10 target aesthetics. Average fitness score increases monotonically from 0.14 to 0.87 over 5 generations, with no session showing regression. The bold teal line shows the average across all runs. **143**

Figure C.2 Mutation trajectories per participant in PCA-projected style space. Each arrow connects a source gene to its mutation offspring. Complement mutations (blue) generally produce shorter arrows that stay near the source and contrast mutations (orange) generally produce longer arrows that jump to distant regions. Stacked bars (top right of each subplot) show each participant's mutation mode distribution, indicating that most participants used all three modes. **144**

Figure C.3 Style exploration trajectories for P12 in PCA-projected semantic space. Dot size encodes time spent prompting and adjusting weights with the same gene combination. GENErator (left, 33 styles) shows rapid movement across the space through diverse operations (crossover, complement, contrast, evolution), with small dots indicating frequent transitions between styles. Baseline (right, 10 styles) shows fewer, larger dots concentrated in a small region, with repeated loops back to the same gene combinations, a characteristic of the fixation pattern where designers spend more time tuning familiar styles than discovering new ones. **145**

Figure C.4 The baseline interface. Designers select styles from the gene library (a), adjust weight sliders to control each style's influence (b), and generate images. A generation history grid (c) displays all previous outputs. Clicking any previous result restores its selected styles and weights, enabling one-click backtracking. This provides stronger backtracking than most real-world tools, which require designers to manually recreate previous configurations. **146**

Figure C.5 Mean interaction counts per session for shared operations between GENErator and baseline (weight adjustments, prompt edits, bookmarks, generations). GENErator significantly reduced weight adjustments and prompt edits while increasing bookmarks (designs participants liked). Error bars show standard error. **146**

Figure C.6 Interaction event sequences for all 12 participants. Baseline (right) is dominated by trial-and-error weight adjustments and manual prompting. GENErator (left) shows a diverse mix of crossover, mutation, evolution, and semantic search operations across all participants. More designs are bookmarked (favorited) when using GENErator. **147**

List of Tables

Table 1.1 The four systems and their HAI Triangle alignments. 3

Table 5.1 Comparison of Jigsaw against related tools. Jigsaw supports non-technical designers with a beginner-friendly block editor and offers AI capabilities across multiple modalities. Jigsaw's Assembly Assistant can help automatically recommend a chain of AI models for a designer-specified task. 55

Table 7.1 Comparison of tools for AI-assisted style exploration. 103

Table 8.1 How each system surfaced latent AI capabilities through interface representations. 125

Table 8.2 How interface requirements vary with the AI's role in the human-AI relationship. 128

Table A.1 Full list of implemented models in Jigsaw. 135

Table A.2 Model mosaics created by participants during the free creation task. . . . 136

Table A.3 Example search queries and returned relevant puzzle pieces using the Catalog Panel's semantic search. 137

Table A.4 Example task queries and recommended combination of puzzle pieces using the Assembly Assistant. 137

Table B.1 Analogical inspirations explored by participants and statistics on inspiration categories. 140

Chapter 1

Introduction

"We shape our tools and thereafter our tools shape us." — John Culkin, 1967

Herbert Simon characterized design as the generation and evaluation of alternatives [5], a process in which designers generate options and refine them toward goals that often shift along the way [6, 7]. Throughout this process, the tools that designers work with expand what they are able to explore: a sketch externalizes a half-formed idea so that it can be revised, a moodboard places references side by side so that styles can be compared and combined, and CAD turns complex constraints into something to manipulate rather than calculate. Each of these tools makes some part of the work visible and workable, and my thesis asks what happens when AI becomes one of them.

Modern AI offers designers a new kind of material [8]. Models trained on large datasets encode capabilities that would take years of experience to internalize, learning semantic relationships between styles, structures, and concepts, and they also expose internal mechanisms such as latent spaces and conditioning signals that can be steered [9, 10]. However, while AI opens up a wide design space, current interfaces leave most of it out of reach. Without the right representation, what the model has learned—which combinations work, which styles harmonize, which structures hold together—remains buried in weights and activations and largely inaccessible to designers. Most interfaces still offer text boxes and sliders that bear little relationship to how designers actually work [11]: a text prompt requires designers to articulate what they want before they have seen what is possible, and a parameter such as "guidance scale" can typically only be tuned by guessing. When the interface hides what the model is capable of, designers are left working around it rather than with it.

The word *steering* appears both in machine learning research on controllable generation and in recent HCI work on directing model behavior, and I review both bodies of work in Chapter 2. I use the term here for a more specific act: shaping what a model produces by acting on its controllable mechanisms, rather than by re-prompting or editing the output after the fact. *AI steering interfaces* are the user-facing tools that put those mechanisms in

designers' hands. For design exploration, that means connecting the moves designers already make (sketching, comparing, recombining) to the controls a model exposes.
By *design exploration* I mean the phase of design work where the goal is not yet fixed and the task is to survey, compare, and recombine possibilities. This is what Buxton describes as "getting the right design" rather than "getting the design right" [12], and the systems in this thesis address the first half.

1.1 Thesis Statement

AI steering interfaces designed by aligning design tasks, AI mechanisms, and interface representations let designers navigate, compose, iterate, and remix AI capabilities.
The three elements are:

- **Design tasks**: what designers need to do
- **AI mechanisms**: what models expose for steering
- **Interface representations**: how capabilities become visible and manipulable

This thesis refers to this alignment as the **Human-AI-Interface Triangle** (HAI Triangle). Chapter 3 defines alignment through three testable conditions, one per vertex. The AI vertex requires *controllability*: the mechanism must expose handles a designer can act on. The Human vertex requires *affordance*: the controls must match the moves the task requires. The Interface vertex requires *intuitiveness*: the representation must draw on conventions designers already work with, so they can anticipate what each control will do. When all three hold, designers steer the model by acting on the representation, without translating intent into prompts or parameter values.
Chapters 4 through 7 demonstrate the framework through four implemented systems and user evaluation studies.

1.2 Thesis Organization

In this thesis, I focus on designing AI steering interfaces for design exploration. Chapter 2 reviews related work across five areas: direct manipulation and interaction paradigms, human-AI interaction, creativity support tools, design space exploration, and modern AI controllability, establishing the interface paradigms and AI control mechanisms most relevant to my work.

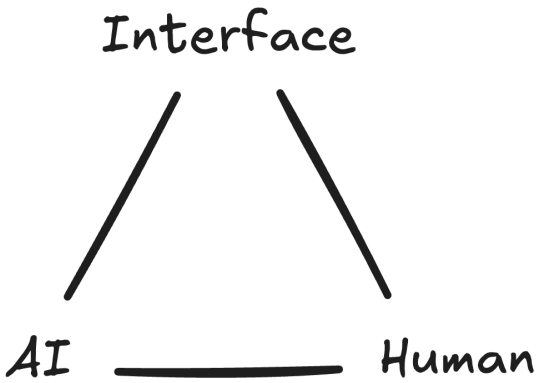


Figure 1.1. The Human-AI-Interface Triangle: three vertices that must align for effective AI steering. Chapter 3 develops the framework.

Chapter 3 then introduces the HAI Triangle, which maps the design space between interface representations and AI control mechanisms. In this chapter I lay out four key tasks for design exploration and describe what goes wrong when the triangle does not hold.
Aligned with this framework, I have designed and built four AI steering interfaces, each addressing a distinct task through aligned AI mechanisms and interface representations. VideoMap (Chapter 4) addresses the *navigate* task by projecting video frames onto a map editors can browse. Jigsaw (Chapter 5) addresses the *compose* task by representing foundation models as puzzle pieces that snap together. Inkspire (Chapter 6) addresses the *iterate* task by pairing sketching with structural conditioning. GENErator (Chapter 7) addresses the *remix* task by treating style adapters as genes that designers cross, mutate, and evolve.

Table 1.1. The four systems and their HAI Triangle alignments.

System	Task	AI Mechanism	Interface Representation
VideoMap	Navigate	Latent space	Maps
Jigsaw	Compose	Pre-trained model chaining	Puzzle pieces
Inkspire	Iterate	Structural guidance	Sketching
GENErator	Remix	Mixing models	Evolution

Chapter 8 synthesizes patterns across all four systems, discusses design guidelines for AI steering, and outlines future directions including generative interfaces and adaptive interfaces.

Chapter 2

Related Work

This thesis builds on prior research in direct manipulation and interaction paradigms, human-AI interaction, creativity support tools, design space exploration, and controllable generation in modern AI. Direct manipulation established that users steer a system when they can see its state and act on it directly. Building on this, human-AI interaction studied how to balance user control with AI initiative, and creativity support research asked what makes tools effective for creative work. Design research, meanwhile, described how designers move through possibility spaces, often unable to articulate goals until they see options. Generative AI completes the picture by producing the controllable mechanisms (latent spaces, conditioning, adapters) that AI steering interfaces can act on. Each thread engages one part of the problem this thesis takes on. The HAI Triangle (Chapter **3**) brings them together; §**2.6** closes this chapter by positioning the thesis among them.

2.1 Direct Manipulation and Interaction Paradigms

The vision of computers as tools for augmenting human capability, rather than replacing human effort, dates to the field's origins. Engelbart's "Augmenting Human Intellect" [**13**] articulated a vision of computers that expand what humans can think and do, treating the computer as a tool that makes people more capable rather than one that replaces their effort. This vision motivates this thesis: AI steering interfaces should augment designers' creative capabilities, helping them explore possibilities they could not access before.

The insight that such augmentation requires making systems visible and manipulable emerged from the earliest interactive computers. Sutherland's Sketchpad [**14**] showed that users could control complex geometric constraints through direct interaction with visible objects rather than through programming. This work established a foundational principle: making system state visible and directly manipulable changes how users can work with computational capabilities.

Shneiderman formalized this principle as *direct manipulation* [15], identifying three properties that make interfaces feel controllable: continuous representation of objects of interest, physical actions rather than complex syntax, and rapid incremental reversible operations with immediately visible effects. These properties reduce the cognitive distance between user intention and system action. Users develop accurate mental models because they can see what the system is doing and immediately observe the effects of their actions.

Hutchins, Hollan, and Norman provided the cognitive analysis of why direct manipulation works [16]. They identified two components of directness: *distance* (how closely the interface language maps to user goals) and *engagement* (whether users feel they are manipulating objects directly rather than manipulating descriptions of objects). Direct manipulation interfaces shorten cognitive distance and raise engagement, lowering the mental resources required for interaction.

Norman extended this analysis through the concepts of *gulfs of execution and evaluation* [17]. The gulf of execution is the difficulty of figuring out how to make the system do what you want. The gulf of evaluation is the difficulty of determining whether the system did what you intended. Effective interfaces bridge both gulfs: they make clear what actions are possible (bridging execution) and make system state visible (bridging evaluation). Norman also adapted Gibson's ecological concept of *affordances* [18]: the relationship between an object's properties and what the agent using it can do. Affordances became central to understanding how interface representations shape what users perceive as possible.

Subramonyam and colleagues extended Norman's framework to AI systems, identifying a third gulf specific to modern AI: the *gulf of envisioning* [19]. Where execution and evaluation address doing and understanding, envisioning addresses imagining. Users struggle to picture what they want when the space of possibilities is large and unfamiliar. The gulf of envisioning captures three misalignments: not knowing what the task should be, not knowing how to instruct the AI, and not knowing what to expect from the output. AI steering interfaces must therefore help users envision possibilities they cannot yet articulate, alongside execution and evaluation.

The gulf of envisioning matters especially for AI steering, because the space designers must envision is encoded inside the model rather than enumerated in advance. Without an interface that surfaces how possibilities relate, designers fall back on guessing, which means describing what they have not yet imagined and accepting whatever the model returns. AI steering interfaces should therefore show structure in the output space, not just present individual results, so that envisioning becomes something designers do by looking rather than by writing.

The information visualization community developed related principles for navigating large information spaces. Shneiderman's Visual Information-Seeking Mantra (overview first, zoom and filter, then details on demand) [20] established that users need spatial representations that show structure at multiple scales. Bederson's work on zoomable interfaces [21] showed that continuous navigation through information spaces supports wayfinding and orientation in ways that discrete pagination cannot.

Recent work has begun applying direct manipulation principles to AI systems. DirectGPT showed that continuous representation, manipulable outputs, and undo mechanisms improve LLM interaction compared to chat interfaces [22]. The same principles that made traditional computing controllable can make AI controllable, if we can find the right representations.

The question this thesis addresses is how to extend direct manipulation to modern AI: how to make AI capabilities visible, and how to create representations that users can manipulate to steer model behavior. The challenge is that AI systems operate on learned representations (latent spaces, attention patterns, adapter weights) that have no natural visual form. That is the design problem this thesis takes on, and the four systems are concrete attempts at answers.

2.2 Human-AI Interaction

Horvitz proposed *mixed-initiative interaction* [23], establishing principles for integrating automated services with direct manipulation. The core insight is that autonomous actions should be taken only when the expected benefit exceeds the cost of potential errors, and that interfaces should allow efficient direct invocation and termination of AI assistance. Users should remain in control of when and how AI participates.

A foundational debate between Shneiderman and Maes [24] established the tension between user control and AI autonomy. Shneiderman argued that direct manipulation provides predictability and supports learning, since users come to understand systems by operating them. Maes argued that some tasks benefit from delegation, especially for users without technical expertise. This debate framed a key question for human-AI interaction: when should users steer, and when should AI act autonomously?

Shneiderman later proposed that this framing presents a false dichotomy [25]. High automation and high human control are not necessarily trade-offs, and systems can be designed to provide both. This reframing motivates this thesis: AI steering interfaces should make powerful AI capabilities accessible through intuitive controls, so that the model can do hard generative work while users keep meaningful direction over what gets produced.

Amershi and colleagues developed 18 guidelines for human-AI interaction, validated through studies with design practitioners [26]. These guidelines address the full interaction lifecycle: making clear what the system can do, supporting efficient invocation, providing feedback about AI state, and letting users correct AI mistakes. The guidelines provide practical design heuristics, though they do not specify how to surface AI capabilities that lack obvious interface representations.

Parasuraman, Sheridan, and Wickens proposed four automation function classes: information acquisition, information analysis, decision selection, and action implementation, each spanning levels from fully manual to fully automatic [27]. The framework established that automation does not simply take over human activity but changes that activity, imposing new coordination demands. Research on trust in automation [28] identified two failure modes: *disuse* (when users don't trust capable systems) and *misuse* (when users over-trust unreliable systems). The goal is *calibrated trust*: users should trust AI based on actual capability. This requires interfaces that help users understand what AI can and cannot do.

Recent work on co-creative systems has examined what happens when AI takes on a partner role in creative work. The formative studies in this thesis (Chapters 4 through 7) and prior co-creative research [25, 26] both find that users prefer high control over AI behavior, even when offered more autonomous assistance. Two reasons recur: uncertainty about what AI can do, and limited transparency into what it is doing. Effective interfaces should therefore make AI's intermediate steps visible and manipulable, so users can verify and redirect behavior throughout the creative process.

This thesis focuses on *steering* for design exploration specifically. The control question takes a particular form in design work. Designers move through possibility spaces iteratively, often unable to articulate goals until they see options, and in doing so they survey, compare, backtrack, and branch. Steering interfaces must therefore support control over how to move through the space of possibilities, on top of control over what gets produced. The systems in this thesis approach AI as a capability designers direct through exploration, not an agent that produces answers on demand.

2.3 Creativity Support Tools

Research on creativity support tools examines what makes computational tools effective for creative work. Boden's foundational analysis identified three types of creativity: combinatorial (novel combinations of familiar ideas), exploratory (traversing an established conceptual space), and transformational (restructuring the conceptual space itself) [29]. Each of these types calls for different tool requirements, and exploratory creativity in particular requires

representations that designers can move through.

Shneiderman's framework [30] laid out properties that support these creative modes: many paths to success, low thresholds for beginners and high ceilings for experts, and support for collaboration. The framework also stresses that creativity tools should support the full creative process, from collecting inspirations through creating artifacts to sharing results.

Research on constraints and creativity pushes back on the assumption that creative freedom requires removing limitations. Stokes showed that creative breakthroughs often emerge from self-imposed constraints that force exploration beyond familiar territory [31]. Acar and colleagues found that constraints push people off the path of least resistance, prompting search for more distant ideas [32]. However, the relationship depends on interpretation, and people who see constraints as opportunities benefit most.

This reframes how to think about AI steering interfaces. Interface representations are themselves constraints, in that they limit how designers can interact with AI capabilities. The design question is whether those constraints are productive ones, opening up creative moves that designers would not otherwise reach.

Schön's concept of *reflection-in-action* [7] describes design as a conversation with materials. Designers make moves, observe results, and reflect on what the situation reveals. Suwa and colleagues found that sketching lets designers notice *unexpected discovery*: they perceive unintended relationships in their own sketches that trigger new goals [33]. Effective design tools support this reflective dialogue by presenting results in forms that invite reinterpretation. AI steering interfaces can support reflection-in-action when they make AI outputs visible in structured ways. The interface representation shapes what reflective moves become possible.

2.4 How Designers Explore

Design research has studied how designers explore possibility spaces. MacLean and colleagues argued that design should produce not just artifacts but *design spaces*, representations of possible solutions and the reasoning behind choices [34]. Their QOC notation (Questions, Options, Criteria) captures design rationale by showing how a design could be different and why it is the way it is. This framing motivates the thesis: design as exploring a space of possibilities, not optimizing toward one solution.

Buxton's work on sketching established that exploration requires rapid generation of many possibilities [12]. The core tension in design is between "getting the design right" (refining a chosen direction) and "getting the right design" (exploring to find promising directions). Sketching supports the latter by letting designers try ideas quickly and at low commitment. Designers think through sketching, and the act of making externalized representations reveals

possibilities that were not apparent before.

Research on *design fixation* [35] showed that designers can become anchored to initial examples, constraining their exploration of alternatives. Even flawed examples influence solutions, because designers incorporate features from what they have seen. Goldschmidt and Smolkov found that the type and modality of stimuli matter: abstract stimuli may support creativity more than concrete visual examples that invite direct copying [36].

Dow and colleagues showed that *parallel prototyping*, creating multiple alternatives before feedback, outperforms serial iteration [37]. Serial iteration can trap designers in local optima, whereas parallel exploration maintains broader coverage of the option space. This has direct implications for AI steering interfaces: tools should support exploring multiple AI outputs side by side rather than generating and refining one output at a time.

Research on examples and inspiration [38, 39] found that designers use examples a lot, through both active search (seeking specific references) and passive browsing (staying open to unexpected inspiration). Interactive example galleries that present browseable, adaptively-selected examples help users create better designs. The key is supporting deliberate use of examples rather than defaulting to the most readily available stimuli.

Recent work has extended inspiration research to AI-augmented design. Kang, Lin, and colleagues developed BioSpark, which uses LLMs to go beyond showing analogical examples to actively scaffolding the cognitive work of transfer. The system helps designers identify mechanisms, map them to their problem domain, and elaborate details [40]. This suggests that AI's role in supporting exploration can extend from presenting possibilities to helping users work with them.

Gonçalves and colleagues found that designers tend to "hijack their own inspiration process" by reaching for familiar stimuli [41]. More conscious and deliberate search leads to more creative results, and ambiguous or partial images foster more novel designs than concrete examples do. This suggests that AI steering interfaces should present possibilities in forms that invite interpretation rather than direct copying.

Recent work has begun applying design space concepts to modern AI. Luminate uses LLMs to generate key dimensions from user prompts, then produces structured responses for systematic exploration rather than single outputs [42]. Studies with professional writers showed improved design space exploration compared to unstructured generation. This suggests that AI interfaces can move beyond "generate one thing" toward "explore a space of possibilities," the same shift that design research advocates.

From this tradition, I identify four tasks central to design exploration: *navigate* (survey possibilities, discover structure), *compose* (build complex solutions from simpler parts), *iterate* (refine through rapid feedback), and *remix* (discover goals through variation). These four

tasks are the design lens used throughout this thesis, with Chapter 3 developing each in detail.

2.5 Generative AI and Controllable Generation

Generative AI research has developed mechanisms for controlling model outputs. The rise of diffusion models [10, 43] and large-scale text-to-image systems [44] created new generative capabilities, but also revealed a gap between what models can do and what users can access through standard interfaces. Subsequent research developed mechanisms at four levels, each described in turn below: structural conditioning, parameter-efficient fine-tuning, latent space navigation, and attention manipulation.

Structural conditioning lets users constrain generation through spatial signals. ControlNet [4] injects edge maps, depth maps, poses, and sketches into diffusion models, preserving structural intent while letting the model fill in details, T2I-Adapter [45] provides similar control through lightweight adapter modules, and GLIGEN [46] adds layout control through bounding boxes. These mechanisms make it possible to say "generate an image with this structure" rather than hoping the model infers structure from text alone.

A second family of mechanisms adapts models to specific styles or subjects through parameter-efficient fine-tuning. LoRA (Low-Rank Adaptation) [47] trains small adapter modules that modify model behavior without changing base weights. DreamBooth [48] fine-tunes models to generate specific subjects from a few example images. Textual Inversion [49] learns new "words" in the model's embedding space that stand in for concepts. With these mechanisms, users can point to examples rather than describe style in words.

A third family operates inside the model's learned representations. GANSpace [50] discovered that principal components of GAN latent spaces correspond to interpretable attributes like viewpoint and lighting. InterFaceGAN [51] found linear boundaries that separate attributes like age and gender. Generative models learn structured spaces where directions correspond to meaningful variations, and moving through these spaces provides a form of control distinct from prompting.

A fourth family manipulates attention to support localized editing. Prompt-to-Prompt [52] showed that swapping cross-attention maps allows targeted changes while keeping the rest of the composition intact. InstructPix2Pix [53] accepts natural language editing instructions and changes the relevant part of the image.

Despite the range of mechanisms these four families provide, current interfaces expose them poorly. Commercial and research tools have converged on a small set of patterns: prompt-based generation (text box and generate button), parameter sliders (guidance scale, sampling

steps, style strength), gallery browsing (scrolling through pre-made styles or outputs), and node-based workflows (connecting processing blocks in directed graphs). Each pattern addresses part of the control problem, but none surfaces the full structure of what AI models can do. Prompt interfaces demand verbal articulation before visual exploration. Sliders expose parameters with no explanation of what those parameters actually do. Galleries let users browse, but the relationships between options stay implicit. Node editors expose composition at the cost of demanding that users understand model internals.

Prompt engineering remains the dominant interaction mode, even though prompts produce variable results and force users to translate visual ideas into words. Yang and colleagues showed that non-experts explore prompts opportunistically rather than systematically, the same pattern researchers have documented in end-user programming [11]. Dang and colleagues found that for most users, writing effective prompts remains trial-and-error with no systematic support, and proposed design goals for prompting interfaces that help users understand model capabilities and iteratively refine their inputs [54].

Terry proposed that effective human-AI interaction requires alignment along three dimensions: specification (communicating objectives), process (controlling execution), and evaluation (understanding outputs) [55]. Current interfaces address none of these dimensions systematically: most tools offer the same generic interface, a text box and a generate button, regardless of what control mechanisms the underlying model supports. Style exploration happens through trial and error rather than structured navigation. Users cannot see how outputs relate to each other or how their inputs map to variations in the output space.

Research on AI tools for creative work has begun addressing this gap. Opal [56] showed that structured exploration pipelines help users find usable results more reliably than open-ended generation. PromptChainer [57] introduced visual chaining of LLM prompts, making pipeline composition tangible. Sketch-based generation tools like Creative Sketching Partner [58] address the modality mismatch between visual thinking and verbal prompting. Yet studies of these tools reveal persistent tensions. DesignAID [59], for instance, found that generative AI can increase design fixation rather than reduce it. These tensions motivate the four systems in this thesis, each exploring a new interface representation meant to bridge the gap between what models can do and what designers can reach.

2.6 Positioning This Thesis

Each of the five areas reviewed above engages one aspect of the problem this thesis takes on, and none of them engages all of it at once. Direct manipulation research developed the principle that users control what they can see and act on directly, however it largely predates

the AI mechanisms that now need such interfaces. Work on human-AI interaction studies how people relate to AI systems, but it rarely commits to a specific representation through which they might steer one. Creativity support research specifies what makes a tool effective for creative work, while leaving open how a large AI possibility space should be rendered so that a designer can move through it. Design research describes how designers move through option spaces, though the spaces it typically studies are hand-curated rather than learned. Generative AI research produces the controllable mechanisms themselves, but it generally treats interface design as someone else's problem.

These five literatures fit together into a single design problem, which I describe in Chapter **3** as the HAI Triangle. A design task, an AI mechanism, and an interface representation must align for steering to work, and each tradition supplies a partial view of that alignment. Design research identifies the tasks, generative AI supplies the mechanisms, and direct manipulation and creativity support together specify what may make a representation usable. My contribution is therefore not the recombination of these literatures, but the observation that AI steering interfaces may succeed or fail at the points where the three meet.

I treat the four systems in this thesis as tests of that idea. For example, VideoMap pairs information visualization with diffusion latents to support the navigate task, while GENErator pairs evolutionary design with style adapters to support remix. Each system is an alignment hypothesis: a specific combination of task, mechanism, and representation, built into a working tool. The user studies in Chapters **4** through **7** then examine whether that alignment allows designers to do work they may not otherwise be able to do.

Chapter 3

The Human-AI-Interface Triangle

AI models hold far more possibilities than a designer can keep track of, and the interface decides which of those possibilities a designer can actually reach. In this chapter, I introduce the HAI Triangle, a framework for designing AI steering interfaces that surface latent capabilities in forms that designers can work with.

3.1 Human, AI, and Interface

I frame the Human-AI-Interface Triangle around three vertices: **Human**, **AI**, and **Interface**. Below, I define each vertex, describe what alignment between them requires, and give tests for whether a tool has achieved it. The framework is intended to be both *diagnostic* and *generative*, in that it may be used to diagnose why an existing tool falls short and to guide the design of a new one.

- **Human**: What do designers need to do? The Human vertex is about design tasks, the goals designers pursue and the actions they take. It draws on Gibson and Norman's concept of affordances [17, 18]: the relationship between what designers want to accomplish and what the system invites them to do.
- **AI**: What can the AI actually do? The AI vertex is about controllable mechanisms inside the model. The focus is on the handles underneath: latent spaces that can be navigated, conditioning inputs that can be structured, weights that can be adapted.
- **Interface**: How does the designer act on what the AI can do? The Interface vertex defines what I call the *interface representation* — the visible, manipulable form (a map, puzzle pieces, a sketch, an evolution tree) through which the designer reaches the AI's mechanisms.

Alignment is the central claim of the framework: for an AI interface to support steering, three properties have to hold at once, one per vertex of the triangle. The AI mechanism has

to expose handles a designer can act on, the controls the interface presents have to afford the human's design task, and the interface representation has to draw on conventions designers already work with, so that the effect of each control may be anticipated before the designer touches it. Each is a testable alignment condition: *controllability* at the AI vertex, *affordance* at the Human vertex, and *intuitiveness* at the Interface vertex.

Controllability (AI vertex). The AI mechanism exposes handles a designer can act on, which typically means interpretable dimensions that interpolate smoothly, discrete inputs that can be substituted, or weights that can be composed. The controllability test fails when the only way into the model is a text prompt or a single generate button. The model may still have considerable capability in such cases. However, its internal degrees of freedom never reach the designer.

Affordance (Human vertex). The interface controls support the moves the design task requires. Affordance is inherently relational, sitting between the human's task and the interface's controls. I anchor it at the Human vertex because the task is what determines which affordances are needed. A navigation task may call for pan, zoom, and filter, while a composition task may call for connection, ordering, and substitution. The affordance test fails when the controls force the designer into unnatural moves. For example, someone who wants to compare options may be given only a serial generate-and-refine loop.

Intuitiveness (Interface vertex). The representation draws on conventions designers already work with, so the effect of a control can be anticipated before manipulation. A map invites pan and zoom; a sketch invites strokes that build up structure; a puzzle piece invites snapping into a compatible neighbor. Because the form is familiar, designers generally reach for the right move and can predict what will happen when they make it. The intuitiveness test fails when an interface presents controls that look like steering but have no readable connection to design practice, such as a numeric guidance scale or a sampling-step counter, where designers can often only learn what a control does through trial and error.

These three conditions are jointly necessary, and a tool can pass one while failing the others. When the conditions blur in practice, it helps to ask them in order. Does the model have a handle to act on at all? Does the control match the move the task requires? Can the designer anticipate what manipulating the control will do from prior experience with similar representations?

Consider, for example, a "creativity" slider on a generative tool. There is no internal axis of the model that maps cleanly to creativity, so controllability fails, and no design task is simply "increase creativity," so affordance fails as well. The slider also draws on no design convention that would let someone predict its effect, and intuitiveness therefore fails too. Each misalignment pattern in the next section can be described as the failure of one or two

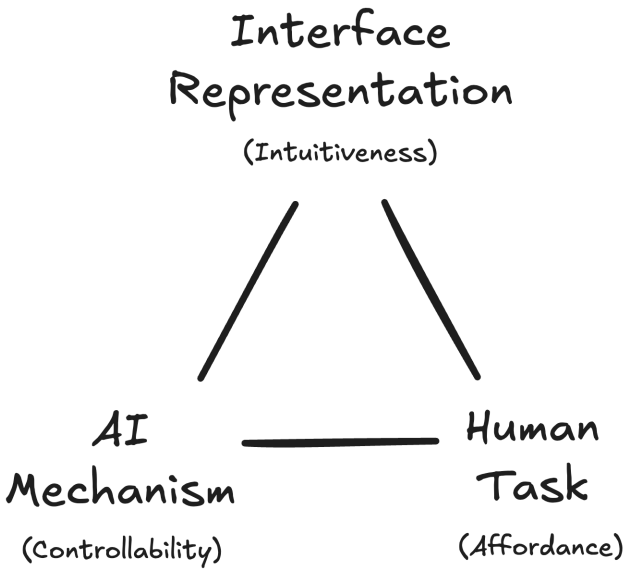


Figure 3.1. The Human-AI-Interface Triangle. Three vertices (Human Task, AI Mechanism, Interface Representation) align for effective AI steering, each with its own alignment condition: *affordance* (Human), *controllability* (AI), and *intuitiveness* (Interface).

of these conditions.

Triadic models are familiar in HCI: Activity Theory uses subject-tool-object mediation [60], and Instrumental Interaction frames things in terms of user, instrument, and domain [61]. The HAI Triangle works in the same tradition. However, it does two things differently. First, it gives three explicit conditions to check (controllability, affordance, intuitiveness), where prior frameworks leave alignment as something to reason about case by case. Second, it treats the tool itself as a moving target. A spreadsheet's computations are not latent the way a diffusion model's style space is. For AI tools, the design problem is therefore not only one of mediation but one of surfacing, and it is around surfacing that the HAI Triangle is built.

3.1.1 Representations as Working Models

The interface representation does the central work in the triangle. A designer does not interact with the AI directly, but rather with the representation, which in turn translates between what the designer intends and what the model can do. Two properties are required for this translation to work.

First, the representation must mean something coherent on the designer's side. A map may mean a spatial overview of related items, a puzzle piece a unit that combines with

compatible others, and a sketch a low-commitment expression of structural intent. Each of these meanings carries its own interaction expectations, whether to pan, to snap, or to draw, and familiar representations lower the cost of entry because designers already have these patterns of practice ready to apply.

Second, the representation must correspond to something the model can act on. A map only works if there is a meaningful 2D projection of the AI's internal state, a puzzle piece only works if the underlying capabilities have well-defined input and output types, and a sketch only works if the model accepts structural conditioning. When the representation maps to a real mechanism, designer moves on one side become model operations on the other; when it does not, the metaphor is decoration rather than control.

This is closer to Norman's notion of a working mental model than to a faithful description of internals [17]. A useful representation does not have to expose how the AI works inside, as long as the designer can predict what their actions will do. A text prompt is thin in precisely this sense: it accepts whatever designers can type, but it commits to no particular structure on the model's side, and small changes in phrasing produce qualitatively different results [11]. The designer cannot predict consequences, so prompting collapses to trial and error. The four representations I develop in this thesis, namely maps, puzzle pieces, sketches, and evolution trees, all attempt the opposite: a fixed, manipulable surface whose effect on the model may be consistent enough to plan against.

3.2 Misalignment Patterns

I also find the triangle useful for diagnosing why AI tools fail. When one vertex's condition fails, alignment breaks along a specific edge of the triangle, and the three patterns trace a directed cycle around it. Each is treated in turn below: affordance failure breaks the Human-AI edge, controllability failure breaks the AI-Interface edge, and intuitiveness failure breaks the Human-Interface edge.

3.2.1 Human-AI Misalignment (Affordance Failure)

In this pattern the mechanism is powerful and the interface presents it well. However, nothing about the pairing connects to what designers need to do. The controllability test passes and the intuitiveness test passes; what fails is affordance, because the moves the interface invites do not match moves the design task requires. For example, neural style transfer turns a photo into a painting, which is impressive once but generally goes nowhere in a workflow built on iteration and comparison. Generative logo tools often sit in the same trap, producing variation after variation with no way for the designer to reason about brand

identity. Text-to-3D systems generate striking models, but the outputs may not be editable, riggable, or droppable into a production pipeline, and the capability therefore never lands in actual work.

3.2.2 AI-Interface Misalignment (Controllability Failure)

In this case the designer has a real task and the interface presents controls, but the mechanism underneath has no clean axis along which those controls can act. The affordance and intuitiveness tests can pass at first glance, with controls that look like steering and metaphors that look readable; what fails is controllability. The most common version is the raw-parameter interface, where a "style strength" knob may produce qualitatively different outputs at 0.7 and at 0.8 because the underlying weight blending is non-monotonic, and where guidance scales and sampling steps only make sense to someone who has read the paper. In each case the slider implies a smooth dimension that the mechanism does not actually have. Designers may then end up moving the values around simply to see what changes, which is closer to guessing than to steering.

3.2.3 Human-Interface Misalignment (Intuitiveness Failure)

Here the mechanism can do what the designer needs. However, the interface offers no representation that the designer can read. This appears to be the dominant failure mode in current AI tools. A blank prompt field asks the designer to say what they want before they have seen what is possible, when in practice designers typically work the other way around, by looking, comparing, and narrowing down. The model may encode how styles relate, which combinations cohere, and where the interesting territory sits, but the textbox flattens all of it into "describe what you want," a form that draws on no design convention for surveying a space. Style libraries with thousands of options sit behind keyword search, and latent spaces are often reachable only through random sampling. Everything the designer would want to explore is already in the system, and the interface simply gives no readable way in.

3.2.4 A Worked Diagnostic: Midjourney

Consider Midjourney, one of the most widely used generative image tools. Midjourney accepts a text prompt and returns four images. Additional controls cover variation, upscaling, style references (`--sref`), character references, image weight, and aspect ratio. A `--stylize` parameter, ranging from 0 to 1000, governs the model's interpretive freedom. The HAI Triangle predicts where Midjourney works well and where it struggles by examining its three conditions in turn.

Controllability is partial. The model encodes a vast style space, compositional knowledge, and semantic relationships, but Midjourney exposes only a few of these as handles. Style references give users an axis on aesthetic direction, and `--stylize` provides a single knob over the model's interpretive freedom. The broader space of what the model can produce has no surfaced axis. There is no handle for compositional structure, no way to traverse style neighborhoods, no control over which semantic dimensions to vary.

Affordance depends on which design task the user is doing. For surveying possibilities (the *navigate* task), Midjourney is reasonable: the four-image gallery and style references afford comparison and selection. For composing across modalities (*compose*), Midjourney has no affordance at all, since it does not chain capabilities. For iterating on a specific direction (*iterate*), affordances are weak: each refinement is a fresh generation from an edited prompt, not a continuation. The design loop runs through words, not direct manipulation. For remixing styles (*remix*), `--sref` provides a partial affordance but does not expose the space of style combinations the model can produce.

Intuitiveness is uneven. Style references and aspect ratio behave intuitively: a given `--sref` code consistently produces images in the same aesthetic family, and aspect ratio is a familiar control from existing tools. Prompts and `--stylize` are not. Small phrasing changes can produce qualitatively different outputs, and `--stylize` is a numeric parameter that draws on no familiar design convention, so users cannot anticipate what raising it from 200 to 400 will do without trying it.

The framework predicts that Midjourney users doing the *navigate* task (looking for inspiration, surveying styles) will have the most successful experience, and that users doing the *iterate* task (refining a particular design intent) will struggle most, and empirical reports match this prediction. Midjourney is widely adopted for inspiration work, where users welcome surprise, and widely criticized for production work, where designers need a specific outcome to refine. The framework explains the pattern: the partial controllability and intuitiveness Midjourney offers are sufficient for navigate, where users survey aesthetic directions; they are not sufficient for iterate, where users need to refine in anticipable steps.

The same analysis applies to other tools in this space. ChatGPT's DALL·E integration is a cleaner case of intuitiveness failure with no structural controls at all. Stable Diffusion's parameter-rich web interfaces sit between controllability and intuitiveness failures: the parameters exist, but they are exposed as raw numbers rather than through a representation designers can read. The four systems in the rest of this thesis are attempts to satisfy all three conditions for one specific task at a time.

3.3 Four Design Tasks

In this thesis, I focus on AI-assisted design exploration, which is not a single activity. Designers survey possibilities, build with what they find, refine promising directions, and discover variations by trying things, and most projects involve all four of these, with the emphasis shifting as the work progresses.

I call something a *design task* when it meets three criteria. First, it works on a space of possibilities, not a single target. Second, it has a recognizable interaction pattern: overview-and-zoom, connect-and-substitute, sketch-and-revise, or select-and-recombine. Third, a matured AI mechanism gives that interaction something to act on. Taken together, these three filters yield the four tasks I develop in this thesis: navigate, compose, iterate, and remix.

The first two filters draw on design and HCI research; the third reflects the current state of generative AI. Tasks that satisfy the first two but not the third are real exploration moves that no system in this thesis addresses. For example, critique is a way of moving through a possibility space by responding to artifacts. However, the AI side of critique is still developing, in that models producing useful, designer-readable feedback are not yet reliable. Constraint specification, where designers narrow a space by adding rules, would be a fifth task. However, current models accept constraints only indirectly through prompts, so the mechanism side does not yet provide a clean control surface. Comparison sits inside each task rather than beside them: it is what designers do once a task surface shows them options.

I did not derive this taxonomy independently of the systems and then test it. The four tasks were developed in conversation with the four systems, and my claim is not that this is the only correct partition. Rather, the claim is that the same human–AI–interface analysis applies to each task, and that it yields tools which do work existing alternatives cannot. A finer or coarser partition would not break the framework. What matters is that each task gets an end-to-end controllability–affordance–intuitiveness analysis, not that the four tasks are exhaustive.

Each of these tasks makes different demands on the AI mechanism and on the interface representation that sits in front of it, and I discuss each in turn below.

3.3.1 Navigate: Wayfinding in Design Space

Wayfinding through a large collection of possibilities depends on two things: a view wide enough to see the terrain, and a way to move through it on purpose. Map-based interfaces support both, through overview, zoom, and filter. Shneiderman's Visual Information-Seeking Mantra [20] captures the pattern: overview first, zoom and filter, then details on demand.

Pad++ and other zoomable UIs pushed this further with multiscale navigation and semantic zoom [21]. What the designer generally wants here is a spatial overview that exposes relationships and gives them a way to find their place.

Latent space navigation is what makes this tractable on the AI side. GAN and diffusion latents contain semantically smooth directions that can be used for both organizing and editing [62, 63], and a line of work including GANSpace [50], InterFaceGAN [51], and SeFa [64] has shown that interpretable axes can be recovered for meaningful attributes. In other words, the learned representation can be projected and walked along directions that line up with concepts the designer already has names for.

3.3.2 Compose: Building with Modular Pieces

Foundation models can be chained together to do things no single model could do alone [57, 65], with one model's output becoming another's input. The interface question is therefore how to give the designer hold of those connections rather than burying them in code. Flow-based interfaces have a long history of doing exactly this: LabVIEW made graphical dataflow programming widespread [66], and Yahoo Pipes and Node-RED later brought pipeline composition to end users [67]. The shape designers want is a workflow broken into discrete pieces that can be wired up and rearranged on the canvas.

3.3.3 Iterate: Rapid Refinement Cycles

Iteration is about tightening the loop between what the designer means and what the system gives back. Sketching has long been the natural medium for this kind of work, because it lets designers think while drawing, hold things loosely, and obtain feedback early. SILK and DENIM put sketching at the center of UI prototyping [68, 69], and Sketch-n-Sketch showed how a drawing and its underlying code can stay in sync as each is edited. Structural conditioning is what lets diffusion models join this loop. ControlNet adds branches that accept edges, depth, pose, or a sketch as conditioning [4], and GLIGEN takes object layouts the same way [46]. Spatial and structural constraints carry the designer's intent forward while the model fills in the rest.

3.3.4 Remix: Discovering Through Variation

Some goals only become clear by being moved toward, and evolution-based interfaces let designers guide a search by picking favorites instead of writing specifications. Karl Sims showed this for artificial evolution of visual forms [70], and Picbreeder scaled the same idea

into collaborative online evolution [71]. Low-Rank Adaptation [47] provides the AI counterpart: lightweight finetuning toward a particular style, with multiple LoRAs composable at fine-grained weights so aesthetic directions become parameters the designer can mix.

Chapter 4

VideoMap: Maps as an Interface for Navigating the Latent Space

This chapter presents VideoMap, the first of the four systems in this thesis. VideoMap addresses the *navigate* task: video editors working with large collections of footage need to see the whole space of clips and move through it deliberately. VideoMap projects video frames onto a latent space and uses the resulting layout as a map editors can browse. Within the HAI Triangle, the design task is navigating a large collection of footage, the AI mechanism is latent space embeddings, and the interface representation is a map.

This chapter was adapted from: Lin, D. C. E., Caba Heilbron, F., Lee, J. Y., Wang, O., & Martelaro, N. (2024). VideoMap: Supporting Video Editing Exploration, Brainstorming, and Prototyping in the Latent Space In Proceedings of the 16th Conference on Creativity & Cognition (pp. 311-327).

4.1 Introduction

Video editing is creative and complex. Consider a typical workflow. The editor sifts through large amounts of footage and develops an understanding of the material. They conceptualize a narrative. They select clips, find connections between them, and weave them together with suitable transitions. As the edit takes shape, they experiment with different ideas.
Popular video editing interfaces still rely on a sequential timeline, with designs rooted in the metaphors of hand cutting and splicing film [72]. Footage sits in a grid-like asset management panel, listed by filename or creation time [73]. These interfaces support assembling frames in a specific order. But they offer no explicit mechanisms for the *creative exploratory aspects of editing*: developing a holistic view of the footage, finding connections between clips, and experimenting with ideas. We believe that there is an opportunity to build a new video editing interface that better supports this creative process. We focus on the *early stages* of

editing: exploration, brainstorming, and prototyping.

We take inspiration from latent space exploration tools that help users find patterns and connections within complex datasets [74]. Latent space exploration tools leverage data processing and recent machine learning techniques to transform data into vector representations and enable users to visually explore the data on a spatial interface. Building upon this insight, we ask if we can similarly project video frames onto a latent space and allow editors to edit videos within this space.

To investigate this possibility, we developed VideoMap, a proof-of-concept video editing interface based on latent space representations of video frames. VideoMap encodes frames into latent spaces tailored to specific editing tasks. For example, one latent space places frames with similar shapes near each other, helping editors find scenes that link into seamless transitions. Editors may switch between different types of latent space maps that encode information meaningful to video to focus on different types of tasks, a concept we call "swappable lenses." To facilitate more intuitive navigation of the map, we introduce a set of map-inspired navigational elements: nodes, districts, landmarks, and paths [75]. On top of the map, we built three components: (1) Project Panel for exploring footage, (2) Paths Explorer for finding transitions, and (3) Route Planner for prototyping rough cuts.

In a user study, we invited both professional and non-professional video editors to test VideoMap. We found that video editors were able to effectively use VideoMap's components to perform the editing tasks listed above. Editors expressed that VideoMap provides a user-friendly editing experience, reduces tedious grunt work, enhances the overview capability of video footage, helps identify suitable video transitions, and enables a more exploratory approach to video editing. We further explored the design space of VideoMap by showcasing how VideoMap can be customized and extended to support additional applications, including summarizing videos, finding highlight moments within videos, and text-based video editing. This research thus makes the following contributions:

- We introduce **VideoMap, a proof-of-concept video editing interface that operates on video frames projected onto a latent space**. We support intuitive navigation of the latent space through map-inspired navigational elements and facilitate transitioning between different latent spaces using swappable lenses.
- We built **three VideoMap components** to support editors in three common video tasks: exploring video footage, finding suitable video transitions, and quickly prototyping rough cuts.
- We demonstrate the effectiveness of VideoMap in supporting video editors complete creative editing tasks through a **user study** (N=14). Editors felt that VideoMap

provides a user-friendly editing experience, reduces tedious grunt work, enhances the overview capability of video footage, promotes editing continuity, and enables a more exploratory approach to video editing.

- We further demonstrate the versatility of VideoMap by implementing **three extended applications**, showcasing how future developers may customize and extend VideoMap for additional use cases.

4.2 Related Work

Our work is situated among literature on video editing tools, video browsing interfaces, and latent space exploration tools.

4.2.1 Video Editing Tools

VideoMap contributes to extensive literature on enhanced video editing tools. In this section, we discuss some past techniques, such as leveraging video metadata, pen input, tangibles, text-based editing, and automation.

Early research on video editing tools explored how data embedded within and associated with videos could be used to create new editing interfaces. One of the seminal works, SILVER [76], uses video metadata to support editors with additional panels to interface with and edit video footage, such as storyboards, editable transcripts, and timeline views. SILVER shows how the use of video metadata allows for smart editing capabilities beyond simple cutting, trimming, and arranging of video frames on the timeline. [77] analyzes video content for fast motion or zooming and computes a "suitability" score for each video frame to help editors decide which clips to include in the edit. This suitability score takes information about the video embedded in the pixels and makes it available as metadata for editing purposes, showing how metadata derived computationally from the frames can be leveraged to support creative editing work. VideoMap builds upon these works and leverages new kinds of metadata derivable with modern machine learning techniques.

Early researchers have also looked into alternative ways to arrange and trim videos. [78] proposes a way to edit video where the user gives voice comments over the video and the system creates a newly edited video based on the comments. Other research has explored augmenting video editing using pen-based technology, allowing editors to edit videos with direct manipulation [79–81]. Another thread of research explores using tangible interfaces to allow co-located users to edit videos collaboratively [82–84].

More recent research have mostly explored developing video editing tools that cater to a specific domain. A large thread of work focuses on building editing tools for text-focused

videos, such as voiceover videos and talking head videos. [85] helps users edit dialogue-driven scenes by matching video clips to relevant dialogue. [86] lets users edit talking-head videos by editing a text transcript. [87] helps recommend b-roll video clips via interactive transcripts. Similarly, [88] lets users edit text and automatically recommends video clips they filmed to use in the edit. [89] helps users create videos by recommending matching images over text and transcripts through the notion of word concreteness (i.e., the extent to which a word describes something that can be visually experienced). [90] allows users to add images to podcasts by using natural language processing to identify important geographic locations and descriptive keywords. Today, several commercial, text-based video editing tools allow users to edit videos via a paired transcript, including Descript [91] and Type Studio [92]. In Section 4.7.3, we demonstrate how VideoMap may also be extended to support text-based video editing.

There have also been efforts to automate the video editing process using algorithmic techniques, including identifying highlight segments [93–95], mimicking professionally-created videos [96–99], instructing editing objectives in natural language to an LLM-powered agent [100], and automatically synthesizing sound based on video content [101]. In addition, researchers have also explored automatically converting various forms of content into videos, such as web page to video [102], markdown to video [103], and physical demonstrations to video [104]. Although the main objective of VideoMap is not on automatic video editing, we designed a feature that supports editors in automatically generating an initial draft of a video (see Section 4.4.4), suggesting its potential to support new kinds of video editing automation.

4.2.2 Video Browsing Interfaces

A related research area to video editing tools focuses on interfaces to support video browsing. Browsing and selecting videos from a large set of raw footage is an important first step in the video editing process. A large body of research investigates the use of direct manipulation techniques [105] for video browsing. Many works have explored video playback by allowing the user to directly click and drag objects within the video, such as clicking on a car and dragging it along the road to play the video over time [106–109]. [110] creates a timeline with dynamic playback speeds to emphasize important content. Another thread of research supports users in browsing through videos by revealing additional information, such as additional thumbnails [111, 112], visualizing various types of metadata such as brightness intensity and hue histograms [113], enhancing the playback of software tutorial videos by leveraging crowdsourced data from previous user interactions with the tutorials, [114], and improving the playback of surgical videos by surfacing related surgical videos that perform

similar surgical steps [115]. In VideoMap, we leverage the content embedded within videos to enable enhanced browsing of video clips and visualization of connections between video clips. This allows editors to explore videos based on their content, rather than merely viewing them as a list of files.

In addition to visual browsing methods, researchers have also explored adopting text-based browsing and navigation for content-heavy videos, such as for lecture videos [116, 117], movies [118], presentation videos [119], and how-to videos [120]. Such text-based browsing methods allow editors to quickly find the right moment in the footage, helping them focus more on composing the video than on searching for content. In VideoMap, we also leverage text-based search of features derived from a semantic latent space to allow editors to find content within large amounts of footage (see Section 4.4.2.1).

4.2.3 Latent Space Exploration Tools

With advancements in machine learning, researchers in visualization and interpretable machine learning communities have recently explored methods for visualizing the embeddings of neural networks. These embeddings are vector representations of data that have been learned by neural networks. A common initial step of such works involves reducing the dimensions of these vector representations, which are often high-dimensional, to just two dimensions. This allows for the data to be visually plotted using x and y coordinates, resulting in a 2D latent space. Various techniques can be used to reduce the dimension of embeddings while retaining as much information as possible. Some popular methods include t-SNE [121], HSNE [122], PCA [123], and UMAP [124]. In VideoMap, we adopt dimensionality-reducing preprocessing to develop a new interface for viewing and working with video footage.

Given a 2D latent space, researchers have investigated the development of interactive interfaces to support user exploration [125]. [74] develops an interface to help users interactively discover meaningful relationships among data points in the latent space. [126] allows users to discover structural relationships in data, such as the association of items within groups and the hierarchies of items between groups items. Several works have investigated interfaces for visualizing semantic relationships of text data [127, 128]. [129] analyzes cooking processes at scale by clustering recipes with respect to their structural similarities and facilitates pairwise comparisons between recipes. [130] allows users to visualize a latent space of gestures. [131] support users in visualizing a latent space of human interpretable concepts, such as cars. Many researchers have also worked on tools for comparing multiple latent spaces, such as by highlighting visual changes [132] and through side-by-side comparisons [133, 134]. Most relevant to our work, [135] train video-specific autoencoders (i.e., autoencoder models overfitted to the task of regenerating the videos) to visualize video frames on a latent space,

which could be used to perform video manipulations such as inpainting and creating video textures (i.e., looping videos). In VideoMap, we draw inspiration from latent space visualization interfaces to assist video editors in identifying patterns and connections within video footage. We create latent spaces that are meaningful for video editors, representing various aspects of video editing, such as concepts of semantics, color, and shape.

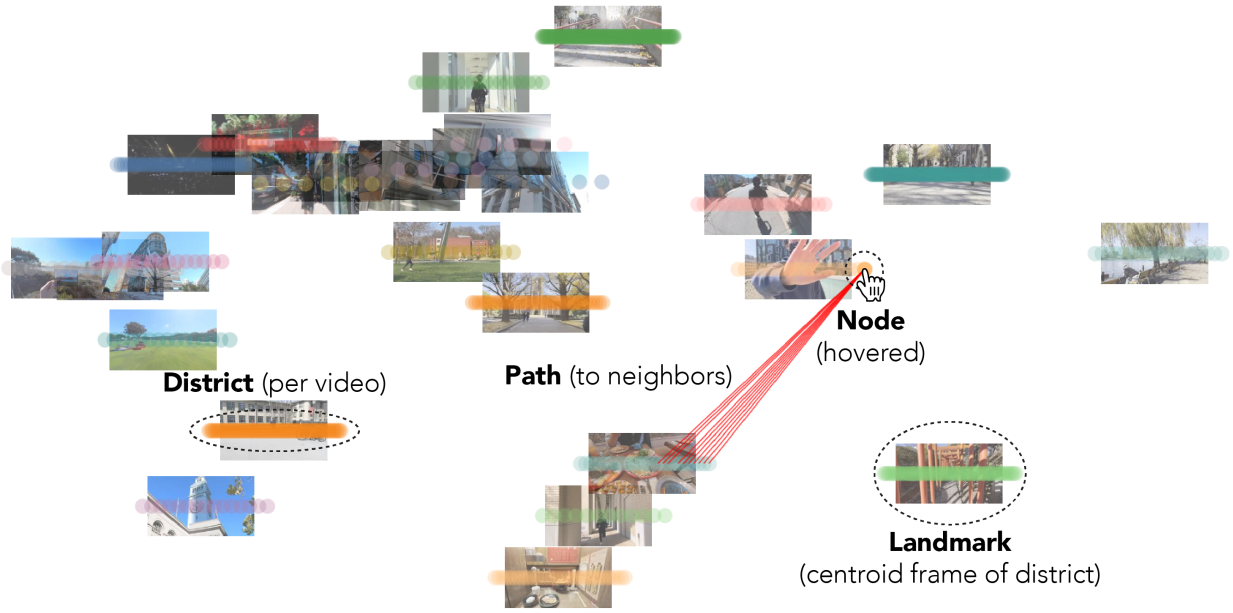


Figure 4.1. VideoMap is a proof-of-concept video editing interface that operates on video frames projected onto a latent space, enabling users to visually uncover patterns and relationships. We introduce map-inspired navigational elements (node, path, district, landmark) to support users in navigating the map.

4.3 Principles of VideoMap

We distill three principles to ground the development of VideoMap based on theories and practices in video editing. These guiding principles include providing an overview of the video footage, maintaining editing continuity with seamless transitions between clips, and facilitating exploration and experimentation throughout the editing process.

4.3.1 Principle 1: Overview

As a time-based medium, video forces editors to spend considerable time playing through and reviewing footage sequentially on a timeline to understand their material. In VideoMap, we overcome the sequential nature of video browsing by projecting clips onto a 2D latent

space that offers a comprehensive overview at a glance. Craft and Cairns [136] suggest that patterns and themes in data can only emerge from a vantage point that comprises the whole view. Map-like interfaces, such as plots on a 2D latent space, can provide such an overview of data and have been shown to allow people to better understand the thematic information across a dataset [137]. We posit that with an enhanced overview, editors will be able to more easily identify patterns and structure and conceptualize a narrative thread from a set of video clips. Furthermore, we designed "swappable lenses" for editors to easily switch between different types of overviews (i.e., different latent spaces) to uncover different types of patterns within the footage, such as patterns in semantics, color, or shape (see Section 4.4.1.2). We designed the Project Panel component to test how VideoMap can support editors through an enhanced overview capability.

4.3.2 Principle 2: Continuity

Continuity is one of the key principles of video editing. In the 2012 Asia-Pacific Symposium on Creative Post-Production, Richard Crittenden, author of the book Film and Video Editing, comments: "*It remains true that good editing tends to be the art that conceals art*". The goal of continuity is to connect clips seamlessly so that the edits are "invisible" to the viewer [138]. One editing technique that editors use to achieve continuity is called match cutting [139]. For example, the editor could cut from a video frame of a bagel to a video frame of a donut — using two different objects with similar shapes to transition between present and past scenes, for example. In VideoMap, we aim to assist editors in achieving continuity by recommending seamless video transitions. Since VideoMap operates in the latent space, we suggest video transitions for a video clip or frame by finding near neighbors in the latent space. Taking the shape-based match cut example: By creating a latent space where video frames containing similar shapes are organized near each other, the bagel video frame and the donut video frame will be near neighbors in the latent space. We designed the Paths Explorer component to test how VideoMap can support editors in achieving continuity in editing.

4.3.3 Principle 3: Exploration

Even with a well-defined narrative and a principle of continuity to follow, video editing remains a process that involves exploration and experimentation. Film director Ridley Scott likened editing to the process of painting: "*Editing is like painting, only with film. You are constantly making choices and trying to find the best way to tell the story.*" In VideoMap, we aim to provide editors with a wide range of exploration mechanisms. First, we created four

map-inspired navigational elements to help editors explore the latent space: nodes, paths, districts, and landmarks [75]. Second, we implemented a semantic searching mechanism in the Project Panel component, allowing editors to quickly locate and discover clips using text prompts. Third, we recommend multiple alternative video transition options for editors in the Paths Explorer component. Finally, we designed the Route Planner component, which can automatically assemble video cuts from user-selected clips, enabling editors to quickly test and preview the results of different editing ideas.

4.4 Implementation

Our three principles are manifested in VideoMap and guide its implementation. The following outlines the implementation of our system. We first detail how we built the base layer latent space map that powers the VideoMap interface. We then go through how we designed VideoMap's main components, including the Project Panel for exploring video footage, Paths Explorer for finding suitable video transitions, and Route Planner for prototyping video route cuts.

4.4.1 Building the Map

We implemented VideoMap's base layer by first creating the 2D latent space. We then enable flexible reconfiguration of different latent spaces using swappable lenses, and support intuitive navigation of the map through map-inspired metaphors.

4.4.1.1 Creating the Latent Space

VideoMap is based on a latent space representation of video frames. Specifically, we encode video frames into vector representations such that they are meaningful for particular video editing tasks. For example, we can create a latent space map to help editors find shape match cuts, a type of video transition which links together two scenes that contain similar shapes to create a smooth transition (**see an example from Kubrick's 2001: A Space Odyssey**). To create this map, we pass video frames through DeepLabv3 [140], an image masking model, with a ResNet [141] backbone and extract a 512-dimensional vector representation right after the last visual layer (Figure 4.2a) Our insight is that due to the mask generation objective, which extracts the dominant shape(s) of an image, vector representations extracted from DeepLabv3 implicitly encode valuable shape-related information. In other words, video frames that contain similar shapes should have more similar vector representations.
Next, we project the vector representations of all video frames into a multi-dimensional latent space. To allow editors to browse through the latent space, we apply t-Distributed

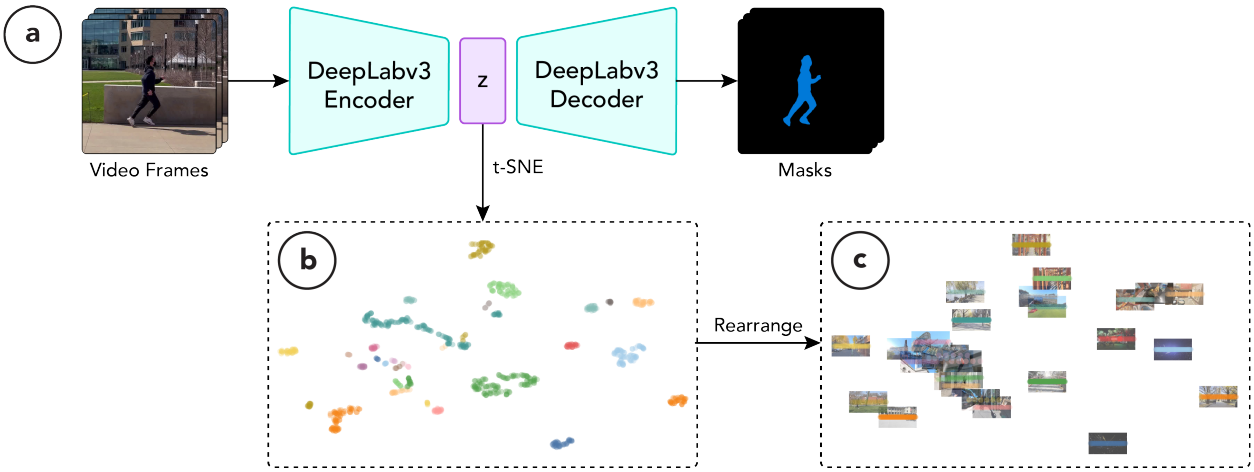


Figure 4.2. The pipeline for creating the `shape` lens. We pass video frames through DeepLabv3, an image masking model, to extract shape-related vectors (a). We then apply t-SNE to reduce the vectors to two dimensions (b) and visually rearrange the vectors by video to make the videos more skimmable (c).

Stochastic Neighbor Embedding (t-SNE) [121] to reduce the latent space to two dimensions. In the first iteration of VideoMap, we visualized the vector embeddings directly according to the coordinates generated from the t-SNE (Figure 4.2b). However, in our pilot tests, editors felt that this was too messy. They also wished to discern groups of points that belong to the same videos more easily and be able to "play through" the videos. Therefore, in the current iteration of VideoMap, we visually rearrange the points on the map by video (Figure 4.2c). We first compute the centroid of all points for each video, then we then visualize frames as points horizontally across the centroid at equal distances. We also display the centroid video frames. Editors can thus play through each video by scrubbing through the points horizontally. From our user study, we found that editors considered this design to be easy to use (see Section 4.6).

4.4.1.2 Developing Swappable Lenses

We call the different methods of encoding video frames into meaningful vector representations as different *lenses*. In Section 4.4.1.1, we create a `shape` lens. Under the `shape` lens, video editors can identify shape-based match cut opportunities by looking at neighbors on the latent space. Similarly, one may develop new lenses by finding other ways of encoding useful vector representations for other use cases. In our proof-of-concept, we implemented a `color` lens (video frames organized by color) and a `semantic` lens (video frames organized by meaning). To create the `color` lens, we first extract 3D histograms from each RGB video frame using 8 bins per channel and normalizing with $range = [0, 256]$, then flatten the

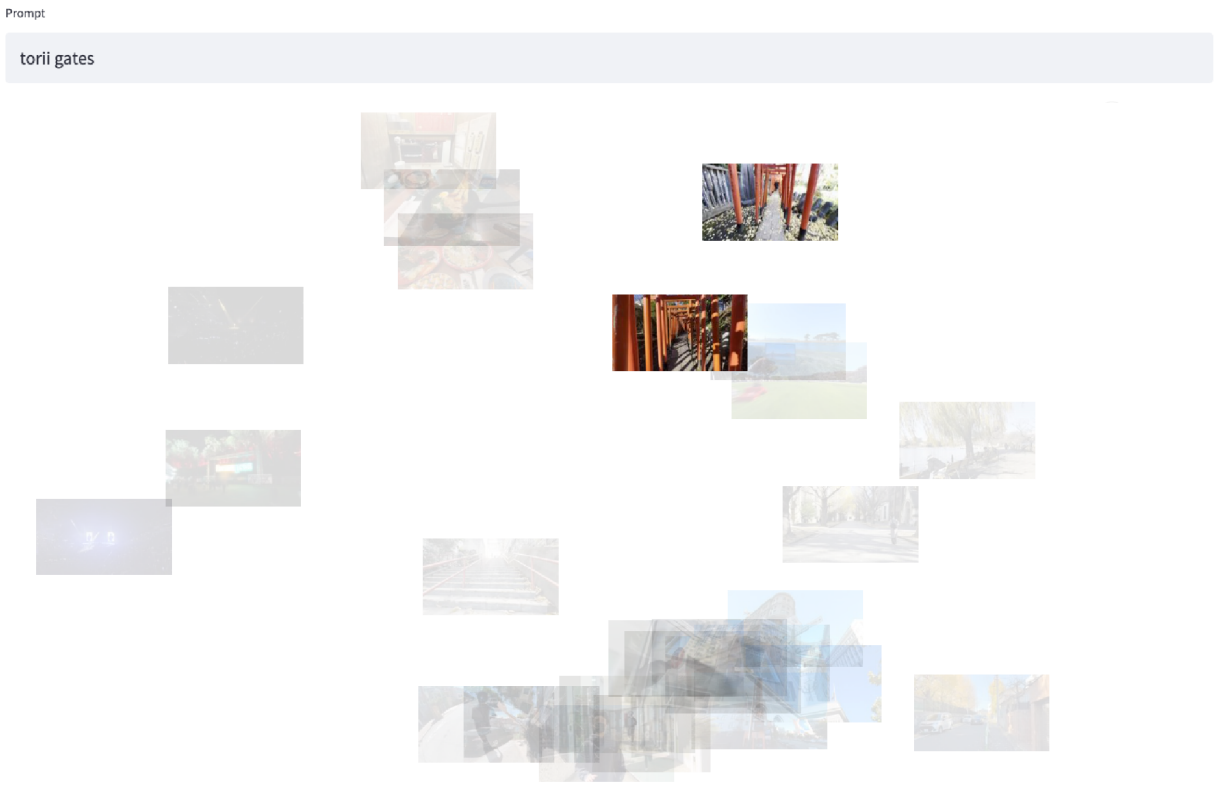


Figure 4.3. The editor can search for videos using text prompts (e.g., `torii gates`).

histogram to generate a 512-dimensional color-based vector. To create the `semantic` lens, we encode each video frame through the image encoder of CLIP [9], a neural network that has learned semantic concepts, to generate a 512-dimensional semantics-based vector. Rather than creating a one-size-fits-all map, we allow editors to flexibly swap between different lenses and reconfigure VideoMap according to different criteria. We demonstrate examples of how video editors creatively made use of the different lenses for different editing tasks in Section 4.5.4.3.

4.4.1.3 Navigating with Nodes, Paths, Districts, and Landmarks

We designed four map-inspired navigational elements to help editors navigate through the map (Figure 4.1). The four elements are nodes, paths, districts, and landmarks. These elements are inspired by Lynch's seminal The Image of the City [75], which studied the elements people use to form mental maps of a city.

Nodes. The node is a point (i.e., a vector representation of a video frame) that the user hovers over with their cursor. We follow Schneiderman's details-on-demand mantra [142] by expanding the node's detailed information, such as its corresponding video frame, video

filename, and timecode.

Paths. Paths are lines that connect from a user-selected node to its neighboring points on the latent space. When a user clicks on a node, we display ten paths that start from the node and connect to the ten closest neighboring points, which are the points with the smallest cosine distances to the node. We exclude neighboring points that belong to the same video as the node to ensure that the paths represent video transitions. To obtain an accurate representation of neighbors, we calculate the distances based on the original vector representations rather than the rearranged coordinates on the visual map (see Section 4.4.1.1).

Districts. Districts are groups of points with common characteristics. Throughout most of this paper, we present the version of VideoMap where points that belong to the same video are grouped into a district. Note that paths connect across two different districts (i.e., video transitions). We color-code the points by district to visually separate them. We also consider alternative methods for defining districts, such as through groupings of semantically similar video frames, in Section 4.7.1.

Landmarks Landmarks are reference markers on the map. With landmarks, the user can obtain an overview of the map at a glance (Principle 1). We visualize the corresponding video frames of some points on the map as landmarks. Throughout most of this paper, we present the version of VideoMap where we generate one landmark for each district by visualizing the video frame of the point closest to the district's centroid. We also explore the option of allowing users to define custom landmarks in Section 4.7.2.

4.4.2 Project Panel

The Project Panel is the first component that the editor can use on VideoMap (Figure 4.4). We designed Project Panel to enable editors to explore video footage more easily (Principle 1). Traditional video editing software typically feature a grid-like interface that displays all of the editor's video files ordered by filename or time of creation. In VideoMap's Project Panel, instead of displaying a standard grid, we rearrange the display of the video files based on the selected lens. For example, Figure 4.4 displays a collection of videos in the Project Panel under the `semantic` lens. The editor can swap between different lenses to rearrange the layout and discover new perspectives on the video footage. To reproduce the video thumbnails seen in traditional video editing software, we display the landmark for each video file. The editor may play through the video files by scrubbing the landmarks from left to right. The editor may also zoom in or out of the map by scrolling to view regions of the map with less clutter.

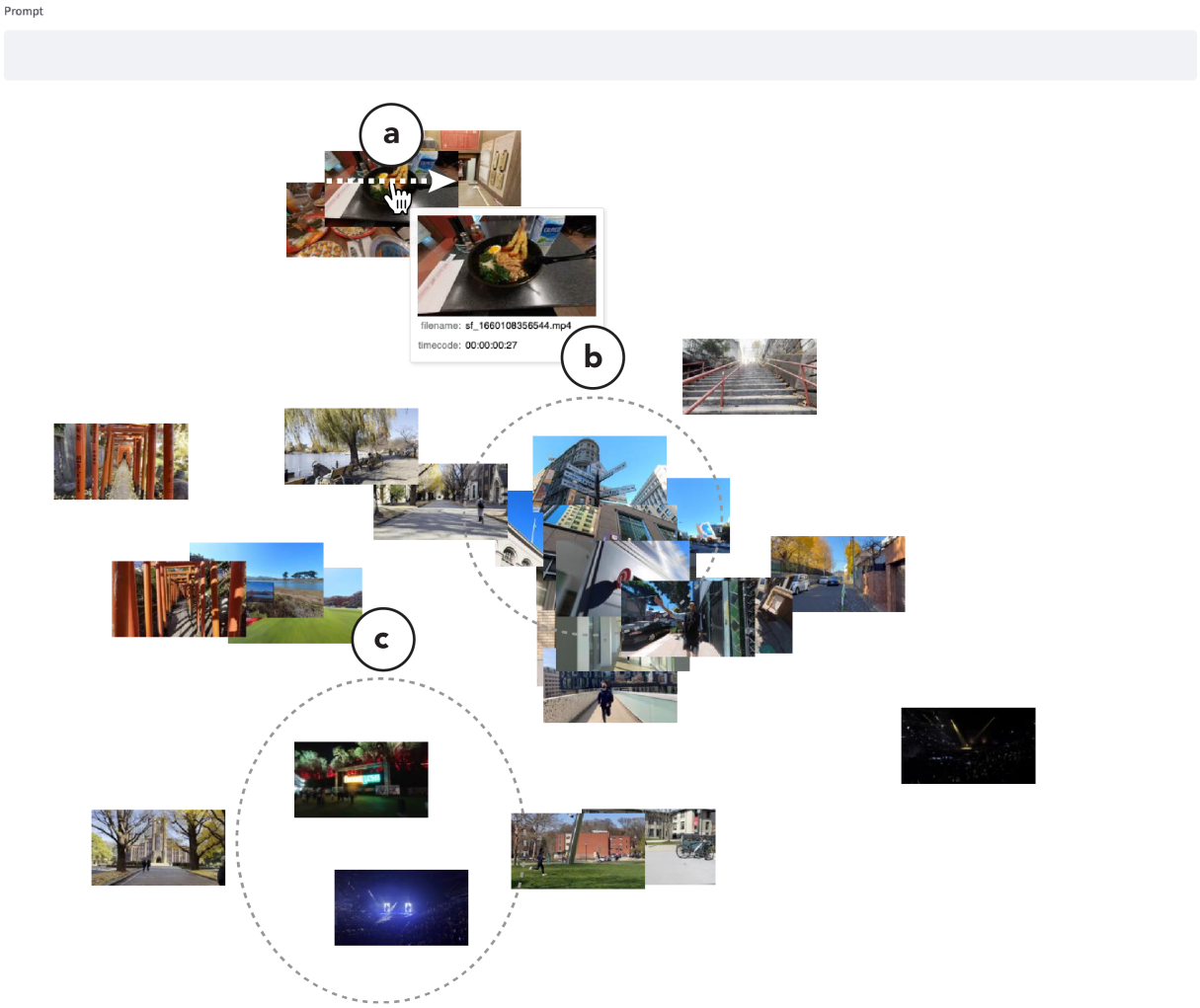


Figure 4.4. VideoMap's Project Panel component. The figure shows a collection of videos organized under the `semantic` lens. The editor can play through the videos by scrubbing the landmarks from left to right (a). Example semantic clusters of videos are shown in (b) for several videos containing streets and buildings and (c) for concert videos.

4.4.2.1 Prompts

The editor can filter videos with text prompts (Principle 3) (Figure 4.3). For example, by typing `torii gates`, the editor can highlight these videos and fade out the rest. We encode the text prompt using a CLIP text encoder and compare the text embedding to the image embeddings of video frames, encoded by a CLIP image encoder (i.e., the vector representations under the `semantic` lens). From our user study, we found that editors felt that this method handled prompt searches effectively (see Section 4.5.4.2).

4.4.3 Paths Explorer

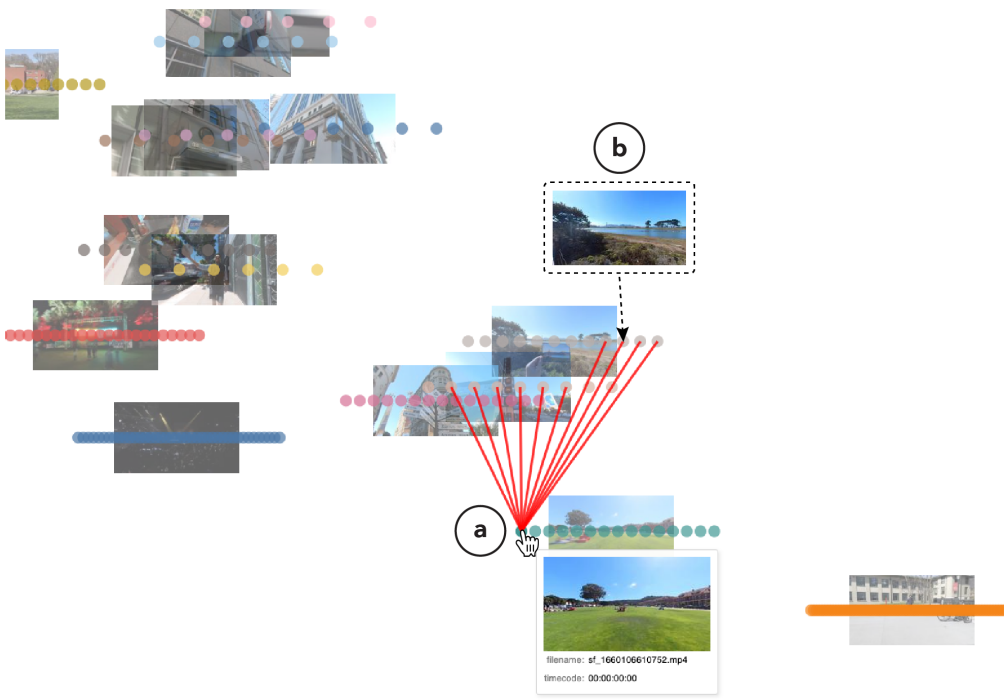


Figure 4.5. VideoMap's Paths Explorer component. The editor can select a video frame to display ten video transition suggestions based on the selected lens (a). For example, a video frame with a similar color composition is recommended under the `color` lens (b).

Paths Explorer is the second component that the editor can use on VideoMap (Figure 4.5). We designed Paths Explorer to assist editors in finding suitable video transitions (Principle 2). Editors typically arrange videos in a sequential timeline, which can make it difficult for editors to identify patterns of video frames at different times and across different videos. With Paths Explorer, we overcome the inherent sequential nature of videos by arranging them on a 2D map, thereby removing the limitations of viewing video frames along a 1D timeline (Principle 1). While we only display each video's landmark in Project Panel, we display each video's individual video frames as points in Paths Explorer. We color-code the points by video (districts) and fade the landmarks to reduce visual clutter. When the editor clicks on a point, the interface draws 10 paths connecting to nearest neighbor vector representations from other video(s) (Figure 4.5b). These paths represent potential video transitions that the system suggests based on the selected lens. For example, Figure 4.5 displays recommended paths that connect between videos with similar color composition (having two prominent blocks of color divided at the horizon) under the `color` lens. The editor can switch between the lenses to discover different types of video transitions.

4.4.4 Route Planner

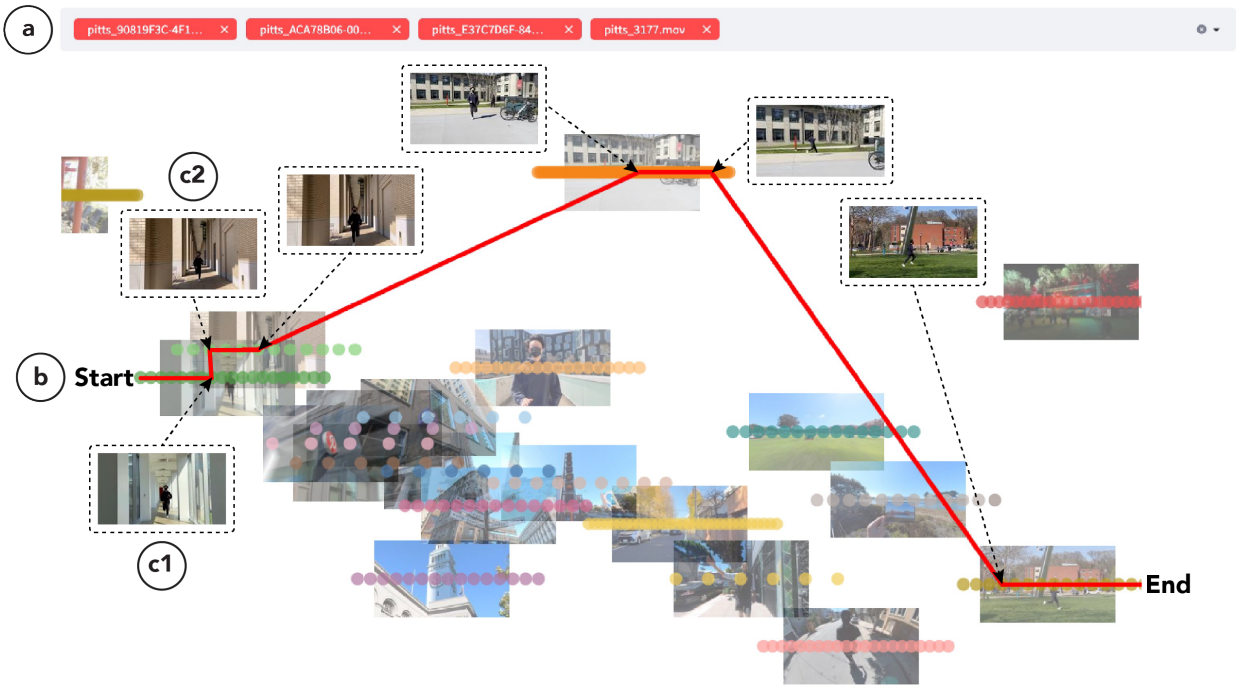


Figure 4.6. VideoMap's Route Planner component. The editor can select several video clips to automatically generate a rough cut video based on the selected lens (a). For example, the editor can create a video of a person running through various backgrounds under the `shape` lens (b). Route Planner automatically finds the optimal video transitions (c1 and c2).

Route Planner is the third component that the editor can use on VideoMap (Figure 4.6. We designed Route Planner to help editors automatically generate rough cuts, which is an initial rudimentary edited version of a video, to quickly explore different video edits (Principle 3). With Route Planner, the editor can select several video clips and the system will automatically find an optimal route across paths (see Section 4.4.3) that connect these video clips, based on the selected lens. For example, Figure 4.6 displays an example route that connects across multiple videos containing the same running action under the `shape` lens. One can think of this component as being analogous to the Google Maps route planner [143]. The paths connecting video clips are like "streets". Route Planner finds the fastest route that travels through the streets and passes through each of the selected video clips once. Since the "distances" between video clips on the map are determined based on the selected lens, the editor can switch between the lenses to try out different rough cuts based on different criteria, such as switching to `color` to connect clips with similar color grading

or `semantics` to connect clips depicting similar objects or activities.
To implement Route Planner, we first find the shortest path for all combinations of video pairs (i.e., optimal video transitions). These paths serve as the "streets" that connect across different videos. Next, we find the shortest route along the streets that passes through all videos (i.e., the Shortest Hamiltonian Path Problem) with a dynamic programming approach. Finally, we visualize the route on the interface and render the rough cut video by traversing through the route. Since the streets are undirected, some videos may be reversed during the traversal. This can lead to interesting results where videos are connected with one playing forward and one playing in reverse.

4.5 User Study

We conducted a user study to observe how professional and non-professional video editors might use VideoMap to complete creative editing tasks, including developing an overview of the video clips they have, finding unique transitions and cuts between clips, and crafting a rough cut of a video. We aimed to address three research questions with the study:

RQ1. How can VideoMap assist editors in exploring video footage?

RQ2. How can VideoMap assist editors in brainstorming suitable video transitions?

RQ3. How can VideoMap assist editors in rapidly prototyping rough cuts?

4.5.1 Study Design

To address our three research questions, we asked video editors to perform the three video editing tasks using VideoMap. Specifically, we asked participants to use the Project Panel to explore and familiarize themselves with a collection of video footage, use Paths Explorer to find video transitions, and use Route Planner to create rough cuts. During the study, we asked participants to share their computer screens and perform a think-aloud [144] describing what they were doing and thinking while using VideoMap.
We collected data using three methods. First, participants completed a questionnaire with written descriptions of their task work (e.g., how the clips are organized) and feedback comparing VideoMap to the editing software they normally use (see Questionnaire in Supporting Documents). Additionally, participants attached screenshots of the interface (e.g., a screenshot of a video transition shown in Paths Explorer that they found interesting) and video outputs from completing the tasks (e.g., a rough cut video generated by Route Planner). Second, we recorded the participants' computer screens and transcribed their spoken audio

throughout the sessions. Third, the first author silently observed participants working and noted down interesting insights, such as participants' thoughts while using the system, interaction observations from the screen captures, and the creations participants made while completing the tasks. Together, these data gave us rich insight into participants' experiences with VideoMap and visual evidence of how it supported common editing tasks.

4.5.1.1 Video Collection

We gathered a sample of video clips for participants to use during the editing tasks in our studies. This collection comprises 30 diverse videos, filmed by the authors in various locations including San Francisco, Pittsburgh, and Tokyo. The content covers a variety of categories, including objects like food and architecture, scenery like streets and grassy fields, and actions like running and scootering. The videos were captured using different camera equipment, such as mirrorless cameras, smartphones, and 360 action cameras. The video clip lengths vary from a couple of seconds to several minutes long. Figure 4.4 provides an overview of the sample video collection displayed in VideoMap's Project Panel.

4.5.2 Participants

We recruited 14 participants (8 male, 6 female) aged from 23 to 55 (μ=28.21, σ=8.78). Seven of the participants are professional video editors recruited from Upwork [145], a platform for hiring creative freelance workers, and seven of the participants are non-professional video editors recruited at our institution via word-of-mouth and snowball sampling. We recruited both professional and non-professional individuals to test VideoMap's ability to support professional editing work as well as make video editing more approachable for non-professionals. We conducted a background survey with the participants before each study to assess their video editing experience. Professional participants have high self-rated familiarity with video editing (μ=5.86, σ=1.46) (7-point Likert scale) and many years of experience (μ=8.07, σ=2.55). Non-professional participants have moderate self-rated familiarity with video editing (μ=4.43, σ=1.27) and several years of experience (μ=3.00, σ=1.15). Participants use various video editing software such as Adobe Premiere Pro, Final Cut Pro, Movavi Video Suite, iMovie, CapCut, and Google Photos Movie Editor.

4.5.3 Procedure

We conducted the studies over video conference (Zoom). We asked participants to try out the three components of VideoMap (Project Panel, Paths Explorer, and Route Planner) and complete a video editing task using each component.

After obtaining the participant's consent, we collected background information about their prior video editing experience. Participants then tried VideoMap. First, they used the Project Panel to familiarize themselves with the sample video collection and think about how they could create a video montage using it. We also asked participants to try the different lenses (semantic, color, shape) and try searching for clips by typing different prompts in the prompts box. Second, the participant used Paths Explorer to find video transition opportunities. Third, they used Route Planner to create rough cuts from the provided clips, picking selections via Project Panel and recording one rough cut they found interesting. After trying all three components, participants completed a post-study survey about their experiences and contrasted them with their usual editing software. The study lasted for approximately 1 hour.

4.5.4 Results

We analyzed our data, including observation notes, audio transcripts, and form responses, with inductive coding to identify common themes. The following presents our findings. We note direct quotes from professional participants as P and non-professionals as N.

4.5.4.1 Project Panel

Participants were able to use Project Panel to explore the video footage by familiarizing, grouping, and uncovering structure among the clips (RQ1). In addition, participants also noted additional use cases of Project Panel, such as supporting visual ideation and helping with planning new shots to take. As part of the first task, we asked participants to brainstorm narrative ideas that utilize the sample video collection. Some ideas from participants include a video starting from wide-angle shots and then showing details of single activities, a video that first shows campus life and then has a fancy transition to life during the holidays (traveling, concerts), and a video introducing various Japanese cuisines.

Familiarize with video footage. Participants were able to use Project Panel to familiarize themselves quickly with the video collection. P3 expressed that "*the beginning process of editing a video can be really daunting*" and Project Panel "*takes a lot of the time out of needing to familiarize yourself with the content that you have*". N3 enjoyed using Project Panel as a "*good preprocessing step*" and expressed that it "*can save people time trying to eyeball videos manually themselves.*" Participants felt that Project Panel "*corresponds well with [their] mental model of how [they] would look for footage (e.g., grouping common themes)* (N4)" and facilitates a "*faster and more intuitive way to explore a collection of video assets* (P4)".

Organizing large projects. Participants noted that Project Panel could be especially useful for organizing large projects, such as being "*a unified hub for large projects involving many team members* (P1)". In particular, many participants enjoyed that Project Panel helps save time and effort required to manually assign tags to video clips or bin video clips into folders (P1, P3, P5, P6, P7, N7): "*It takes forever to tag clips. It's either that or subfolders. It's such a pain to go through a whole day's worth of shooting.* (P7)"

Uncover structure. Participants stated that Project Panel helps "*reveal underlying structure within the recorded content* (P4)", especially those that "*may not be entirely apparent at first*" (P4). Participants enjoyed the "*inspiration and surprise factor* (N5)" and helps create "*a fresh perspective on clips they might have passed up* (P7)": "*Getting a new take or generating different ideas is what AI is great for and is how we can use tools like these to enhance our editing.* (P7)" Participants were also able to use Project Panel to easily identify outlier videos (P2, N5): "*This one color is definitely sticking out* (P2)".

Support visual ideation. Participants expressed that having the videos visually arranged in a 2D space helped them "*have a better sense of the clips that they are working with and see the overall vision of what they want to create* (P5)" (Principle 1). For example, some participants wanted to create a video that takes one scene from each cluster on the map (P1, P5) while others wanted to create a video that focused only on one cluster on the map (P6, N2). N1 noted that "*seeing similar footage organized in the same place helps [them] figure out where to look if [they are] searching for certain elements* (N1)".

Plan the shots. Participants reported that Project Panel helped them with planning the (additional) shots that need to be taken (P5, N2). P5 stated that the "*biggest pro [of Project Panel] was seeing the overall visuals of clips and the amount per grouping. I would use this at the very beginning of my workflow where I decide what other clips I need to take to complete the story. For example, if each scene of a trip is in there or if I need more concert clips.*"

4.5.4.2 Prompts

Participants were able to make use of the prompt box in Project Panel for various purposes, such as quickly searching for clips, searching for similar clips, and searching for cinematic elements. As part of the first task, we asked participants to write down three prompts they found the most interesting during their explorations. The average length of the prompts is around 2 words (μ=2.24, σ=2.00). The most popular prompt was food, which was included by 5 participants. We grouped the prompts into 6 categories. Overall, participants searched for objects 13 times (e.g., `bike`), scenery 10 times (e.g., `outside in a grassy field`), location 6 times (e.g., `restaurant`), action 7 times (e.g., `cooking`), time 3 times (e.g., `daytime`), and cinematography elements 3 times (e.g., `establishing shot`). Some example

prompts from participants include `videos of people` (P2), `running` (N2), `blue sky` (P6), `autumn colors` (N5), `motion blur` (N5), and `establishing shot` (P3).

Search for a clip (instantly). Participants used the prompt box to search for clips quickly: "*I would be editing at the hotel after a day of filming. Suppose I went to Rome today and I remember I went to the Trevi fountain. It'll be way easier to find it [by searching 'trevi fountain'].* (P2)" P5 commented that they could use the prompt box to more easily search for "*clips with triangular elements while not remembering where those clips were*" and P3 commented that they could use it to "*find good b-roll shots*" for interview-style videos.

Search for similar clips. P7 commented that they could use the prompt box to search for similar clips: "*There's a concert clip. I wonder if there are any other concert-related clips.*"

Search for cinematic elements. P3 and P4 commented that they could use it to find specific cinematic elements to use for their videos, such as an "*establishing shot to start off a video* (P3)" or a "*wide-angle shot to sprinkle in some drone footage* (P4)".

4.5.4.3 Swappable Lenses

Participants found the Swappable Lenses concept useful and expressed that it helped shine a light on different perspectives and allowed VideoMap to be able to cater to people with different editing styles. Participants came up with many creative ways of utilizing Swappable Lenses. P4 used the `color` lens to find videos of similar seasons, such as autumn. P5 used the `color` lens to arrange videos chronologically, starting with the day and ending with night scenes. P7 used the `color` lens to identify videos with warmer and cooler tones. N5 used the `color` lens to identify black frames to serve as transition points. N6 expressed the potential of using the `color` lens to maintain a steady color grading scheme. P7 used the `shape` lens to find videos with similar horizon positions. N6 used the `shape` lens to identify videos that contain people running. P4 used the `shape` lens to find videos containing circular shapes. P5 used the `shape` lens to find videos that have similar angles and are rotating in similar motions.

Shine light on different perspectives. Participants expressed that Swappable Lenses helped shine light on new perspectives (N1, N4, N5, P7): "*They give different perspectives that focus on different traits of the footage* (N1)". For example, the `color` lens helped reveal videos with the same color palette or aesthetic (P1, P3, P4, P6): "*In the* `color` *lens, I could see the videos that have warmer colors versus cooler colors, which I didn't notice this in the* `semantic` *lens.* (P7)" In addition, participants found the `shape` lens useful for finding videos containing people doing similar actions (N1, N5, N6, P5, P7) or videos with similar compositions (P4, P7). P7 felt that "*the different lenses created subtle filters to the footage which inspired [their] decision-making process.*"

Something for everyone. Participants expressed that Swappable Lenses allow VideoMap to be flexible in supporting editors who might prefer different types of editing techniques: "*It also serves users' preferences in different video editing techniques. Users can choose ways that work the best for them.* (N3)" Among the 14 participants, 8 participants preferred using the `semantic` lens the most, 3 participants preferred using the `color` lens the most, and 3 participants preferred using the `shape` lens the most.

4.5.4.4 Paths Explorer

Participants were able to use Paths Explorer to identify suitable video transitions by recognizing interesting match cuts with minimal time and effort (RQ2). In addition, participants also expressed that Paths Explorer also helped encourage and facilitate more exploration. As part of the second task, we asked participants to record three video transitions they found the most interesting during their explorations. Some video transitions from participants include using the `shape` lens to cut between scenes that have buildings with similar architectural designs, using the `shape` lens to cut between scenes with similar composition (both having two prominent blocks of color divided at the horizon), using the `semantic` lens to cut between food scenes, and using the `color` lens to cut between scenes featuring trees with similar autumn tones.

Identify match cuts. Participants expressed that Paths Explorer is a "*cool and unique way to conceptually look at footage and generate match cut ideas* (P7)". Participants were able to easily link together different environments with similar vibes (Principle 2) (P1, P7, N1, N2, N3, N7): "*[Paths Explorer] helps find associated concepts across different videos more easily.* (N2)" Non-professional editors found that Paths Explorer changed how they think about video transitions. For example, N3 noted that Paths Explorer "*made [them] put more thought into transitions*" where they "*previously had not thought too much about them.*" N1 commented that Paths Explorer "*inspired a lot of new ideas about how a transition could be.*"

Save time and effort. Participants noted that Paths Explorer helped save time and effort in finding transitions (P3, P7, N1, N5): "*Scrolling over footage and going back and forth can be very time-consuming and tedious* (P7)." P3 commented that "*having a lens that specifically found the paths between matching shapes shaved off hours of looking through footage.*" N5 appreciated being able to see "*the exact frames that can be connected together*".

Encourage exploration. Participants felt that Paths Explorer helped encourage more exploration compared to traditional video editing software (Principle 3): "*It's a more exploratory approach than with the Premiere interface. I can discover relationships between scenes easier. It's a great way to anticipate multiple possible transitions without committing*

to an idea. (P4)." N1 appreciated the ability to view multiple transition suggestions together: "*Oh these are kinda cool. This is probably better. I'll choose this one.* (N1)" N2 enjoyed the ability to "*simultaneously explore frames at different time points and not being constrained by the temporal ordering of video content.*"

4.5.4.5 Route Planner

Participants were able to use Route Planner to quickly prototype rough cuts with automatic editing of clips (RQ3). In addition, participants also commented that Route Planner is able to help their editing workflows by generating ideas, facilitating quick previewing of ideas, and suggesting enhancements. As part of the third task, we asked participants to record a rough cut generated by Route Planner that they found the most interesting. Some rough cuts by participants include using the `shape` lens to generate a video of a person running with different backgrounds, using the `shape` lens to generate a video with a seamless foot transition, and using the `color` lens to generate a video that starts from daytime and ends with nighttime.

Automatic editing. Participants enjoyed being able to quickly generate edited rough cuts using Route Planner (P2, P3, N1, N2, N3, N6): "*It reduced the hassle of trying to arrange clips in a way that makes visual sense. Using traditional software would have made this process take hours.* (P3)" P2 and N6 commented that Route Planner could be helpful for publishing video content on social media platforms: "*It could be great for platforms that need content generated quickly, like TikTok. It could easily help create content like a supercut or quick preview.* (P2)" Participants felt that Route Planner generated quality cuts: "*It's a very cool video. That foot part didn't even seem like a transition. It just seemed so seamless that I thought it was the same video.* (N3)"

Generate ideas. Participants commented the Route Planner can be used for idea generation. For example, N1 commented that they could throw clips into Route Planner before jumping into editing to "*quickly plan out an editing plan.*" On the other hand, N5 commented that they would primarily use Project Panel and Paths Explorer and "*use Route Planner if [they] run out of creative ideas.*"

Preview ideas quickly. Participants felt that Route Planner helps editors "*try out different ideas faster than the traditional software* (N6)" (Principle 3). P7 commented that "*[Route Planner] was a really good way for [them] to visually come up with and execute ideas. When using Route Planner, I had thought of many different ways the videos and clips could have been edited together.*"

Suggest enhancements. Participants noted that Route Planner could be used as a tool for suggesting editing enhancements. P7 commented that they could first "*manually do a*

rough cut, then give Route Planner the clips that [they] used and see if it could come up with a better cut."

4.6 Discussion

Participants generally felt that VideoMap offers a user-friendly approach to video editing (P3, P5, P6, N1, N2, N6). Specifically, non-professionals remarked that VideoMap "*makes editing tasks much easier* (N6)", possesses a "*smoother learning curve* (N1)", and offers a "*more fun way to edit videos* (N2)." Additionally, participants expressed that VideoMap helps reduce the non-creative grunt work involved in video editing, allowing for greater focus on creative aspects (P5, N5, N6): "*Trying to write my process in Premiere made me realize how complicated it actually is to do this task, [and] how much manual searching and scrolling through frames I do.* (P5)". N6 explained that their workflow of organizing video footage using traditional editors involves "*manually sorting the videos in a file browser, picking the videos that [they are] most interested in, then manually uploading them into the editing software* (N6)". With Project Panel, they could "*have all videos uploaded at once, then filter with prompts using the ideas [they] have in mind* (N6)".

One clear benefit of VideoMap is its enhanced overview capability (Principle 1). By breaking the sequential nature of browsing videos, viewing videos across a latent space in the Project Panel helps editors explore raw footage, enabling them to better familiarize themselves with and uncover structure within the footage (RQ1): "*[VideoMap] helped me get a more comprehensive overview of the materials more quickly* (P4)". P7 reflected: "*I think the traditional video editor can leave much to be desired. [VideoMap] is a fresh take on the old concept, allowing for the clips to shine and visually show [me] what they are and how [I] might think to compose them.*" N5 commented that "*[VideoMap] is a more inspirational way of organizing video clips*" and N3 noted that "*[VideoMap] helps editors tell better stories with connected themes, color, and shape*". Given an enhanced overview, participants were also able to visually ideate narratives, such as creating a video that weaves together one clip from each cluster on the map (P1, P5), or conversely, creating a video that only focuses on clips from one cluster (P6, N2). In addition, P5 was also able to utilize the overview to "*decide what other clips [they] need to take to complete the story*", such as capturing more concert footage after noticing relatively small semantic concert clusters on the map.

By operating on the latent space, VideoMap helps visualize semantic or visual neighbors, thereby incentivizing editors to maintain continuity in their edits (Principle 2). Paths Explorer assists editors in brainstorming opportunities to create continuous transitions through transition recommendations to latent space neighbors (RQ2): "*[Paths Explorer] is a cool and*

unique way to conceptually look at footage and generate match cut ideas (P7)". N5 explained that "*using a traditional video editing software might involve jumping back and forth between the videos or overlaying frames on top of each other with a lower transparency to see if there are some common shapes*". P3 contrasted editing transitions using Paths Explorer with their typical workflow: "*It really changes the process of editing. Usually, when you're doing this [with traditional editing software], you already have everything onto a timeline, and you're like, alright, now let's find the transitions.*" Furthermore, non-professionals felt that Paths Explorer "*made [them] put more thought into transitions [where they] previously had not thought too much about them* (N3)", and helped "*inspire a lot of new ideas about how a transition could be* (N1)".

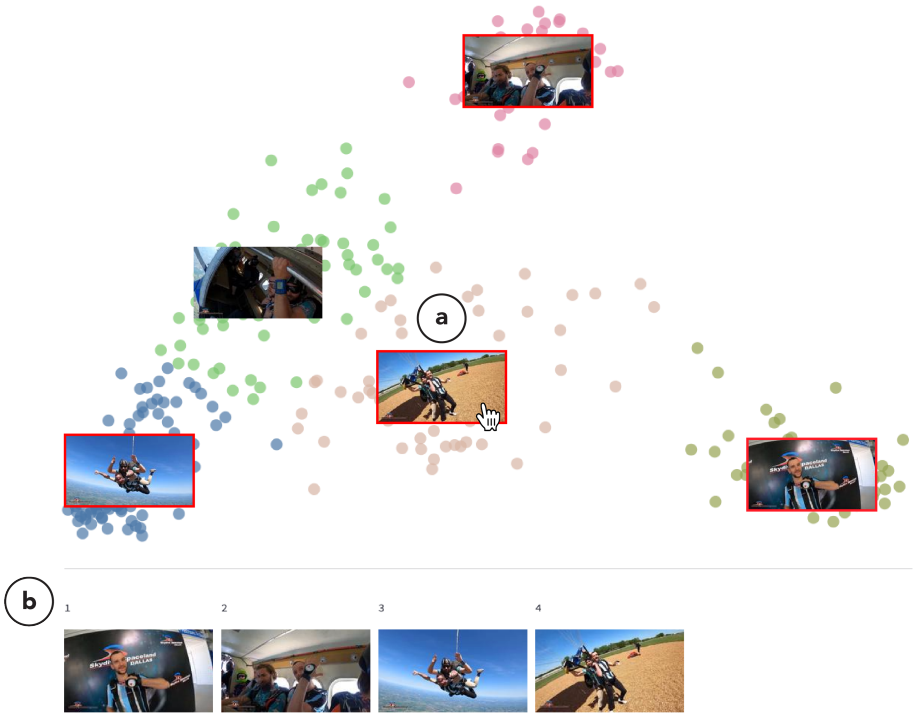


Figure 4.7. VideoMap's Project Panel can be extended to create summary videos. We automatically create "semantic districts" that approximately represent the main activities of a video using k-means clustering under the `semantic` lens. The editor can select several landmarks to specify the activities to include in the summary video. Selections are highlighted with red borders (a) and displayed as a storyboard (b).

VideoMap enables a more exploratory approach to editing through various components and mechanisms, such as automatic rough cuts with Route Planner, map-inspired navigation elements, prompt searching in Project Panel, and recommending multiple possible transitions in Paths Explorer (Principle 3). By automatically generating rough cuts, Route Planner

helps editors quickly test and preview editing ideas (RQ3). Participants utilized the generated rough cuts for idea generation ("*quickly plan out an editing plan* (N1)"), previewing the results of editing ideas ("*I had thought of many different ways the videos and clips could have been edited together* (P7)"), and suggesting enhancements ("*see if it could come up with a better cut* (P7)"). Participants commented positively on various interactions and map-inspired navigation elements, such as scrubbing landmarks to play through videos (P5, P6, N6) and selecting nodes to visualize paths to neighbors (P4, N1, N2). Participants enjoyed using prompts in the Project Panel to semantically search for clips in mind or discover new clips: "*It is like the difference between using Google Search and searching information from books* (N2)." Additionally, P4 commented that "*Paths Explorer helped [them] discover and plan possible transitions much faster*" and felt like a "*more exploratory approach than with [a traditional editing interface]*", given the ability "*to anticipate multiple possible transitions without committing to an idea*".

Beyond its three core lenses (`semantics`, `color`, and `shape`), VideoMap is designed to be extensible. We plan to open-source our code so that developers can build new lenses to support additional editing tasks. The following section demonstrates three such extensions we implemented to test the design space.

4.7 Extended Applications

We implemented three extended applications of VideoMap to demonstrate how it can be customized for additional use cases. These applications include (1) video summarization with semantic districts, (2) video highlights with custom landmarks, and (3) text-based video editing.

4.7.1 Video Summarization

We show how VideoMap Project Panel can be extended to help editors create summary videos (Figure 4.7). We first project the source video's frames into a `semantic` latent space. Instead of creating districts per video, we apply k-means clustering to automatically partition the video frames into "semantic districts." Since we are working in a semantic space, each district approximately represents a main activity of the source video. We determine the number of districts (i.e., k value) using the elbow method [146]. We color-code the districts and visualize the centroid frame of each district as semantic landmarks. To create the summary video, the editor can add semantic districts (i.e., main activities) by clicking on the landmarks, in the order they want them to appear. Selected landmarks are highlighted with red borders (Figure 4.7a) and displayed as a storyboard at the bottom (Figure 4.7b). We

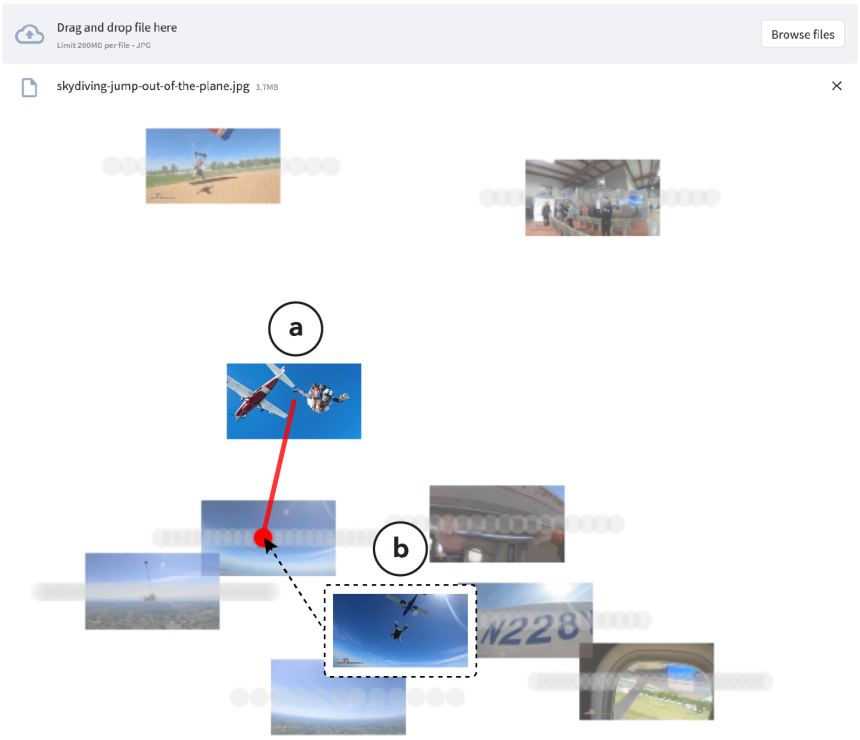


Figure 4.8. VideoMap's Paths Explorer can be extended to create highlight videos. The editor can upload a photograph (i.e., a custom landmark) depicting an activity (e.g., skydiving) (a). Our key insight is that photographs taken by photographers tend to capture the most highlight-worthy moments of an activity (e.g., when the skydiver jumps out of the aircraft). We then generate a highlight video using near neighbor video frames to the custom landmark in the `semantic` space (b).

generate the summary video by taking three seconds of video centered around the landmark for each semantic district).

4.7.2 Video Highlights

We show how VideoMap Paths Explorer can be extended to help editors create highlight videos (Figure 4.8). The editor can first upload a photograph depicting an activity (e.g., skydiving). We then project the photograph into the latent space under the `semantic` lens and visualize it as a "custom landmark" (Figure 4.8a). Next, we draw a path from the custom landmark to the nearest video frame in the latent space (Figure 4.8b). Videogenic [94] found that photographs taken by photographers tend to capture the most highlight-worthy moments of an activity (e.g., when the skydiver jumps out of the aircraft). Thus, we take the photograph's neighboring video frames as the highlight video. We generate the

highlight video by taking five seconds of video centered around the nearest video frame).

4.7.3 Text-Based Video Editing

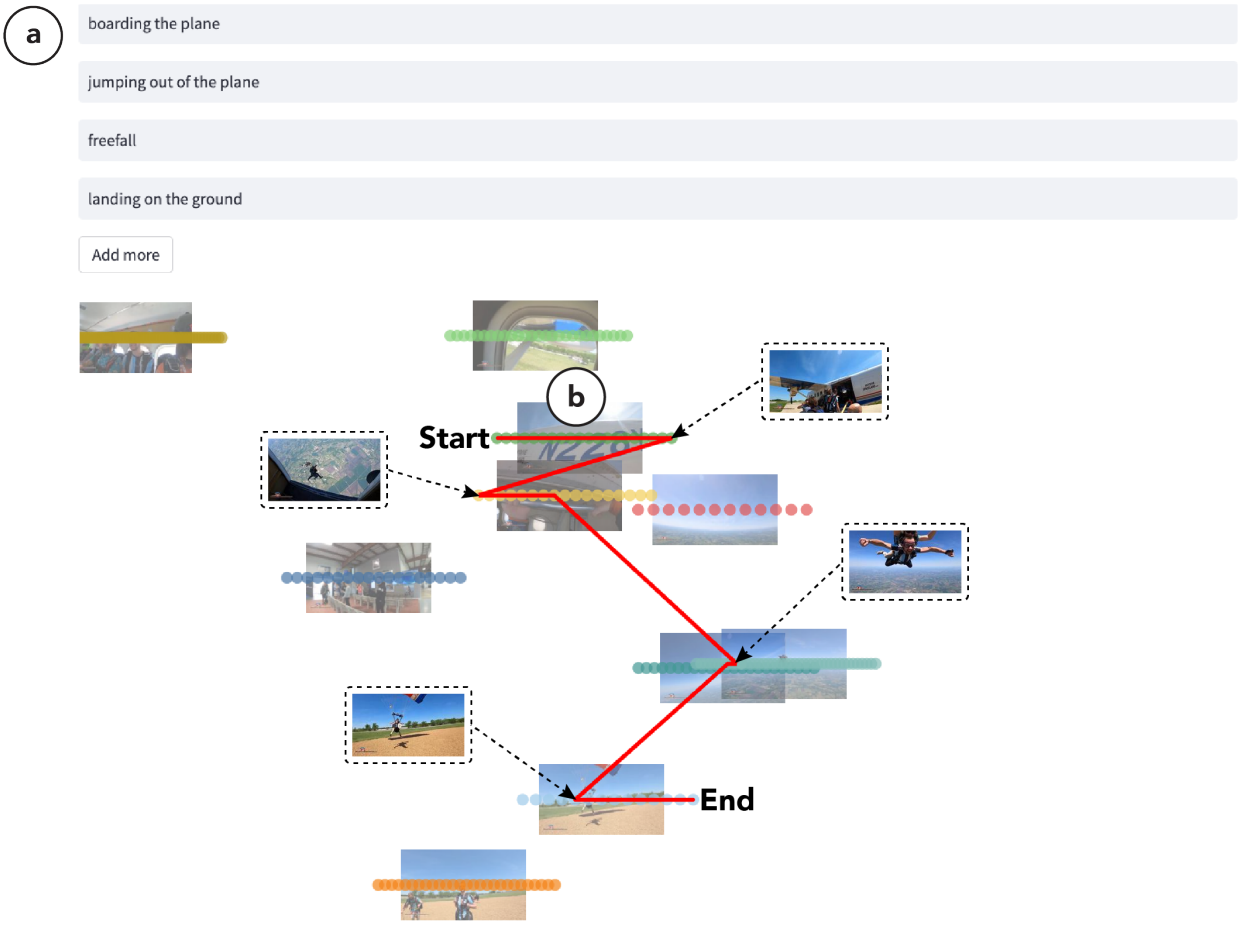


Figure 4.9. VideoMap's Route Planner can be extended to edit videos using text. The editor can describe a desired video using descriptive sentences, like writing a story (a). We then match each sentence to the closest video clip in the `semantic` space and generate a video by finding the shortest route along the clips (b).

We show how VideoMap Route Planner can be extended to support editors in text-based video editing (Figure 4.9). The editor can first describe the video they want to create using descriptive sentences, like writing a story (Figure 4.9a). For each sentence, we match the closest video in the semantic space, using the same CLIP-based method for comparing text and image embeddings as detailed in our implementation for Prompts (see Section 4.4.2.1). We then create the shortest route along the videos using the same dynamic programming approach as detailed in Route Planner (see Section 4.4.4) with an additional ordering con-

straint (i.e., in the order of the story) (Figure 4.9b).

4.8 Future Work

While VideoMap was positively received in our user study, there are several avenues for improvement that we plan to address for future work. First, VideoMap currently supports three types of lenses that facilitate the formation of narratives (`semantic`) and the creation of visual transitions (`shape` and `color`). In our study, participants suggested additional lenses, such as a cinematography lens to cluster videos by different shot types (e.g., by training a shot type classification model), an Image Signal Processor (ISP) lens to cluster videos by white balance, focus, and exposure values (e.g., by utilizing image analysis algorithms), and a motion lens to cluster videos by distinct camera movements (e.g., by computing motion vectors with optical flow [147]). For future work, we plan to open-source the pipeline code for developers to build new lenses by generating new latent spaces that are meaningful for video editing. This pipeline could involve generating embeddings using *AI foundation models* [148] or training custom models tailored to specific video editing objectives. Moreover, we could also explore the creation of "hybrid lenses." For instance, we could create a lens with weightings of 80% shape and 20% semantics to enable the discovery of shape-matching cuts that also possess related semantics. Hybrid lenses could provide editors with the flexibility of blending together the properties of multiple lenses, allowing them to fine-tune the latent space according to their editing goals. Second, due to time constraints, the current design of our user study consists of users trying out VideoMap without a comparative baseline system. For future work, it may be interesting to compare editing using VideoMap against editing on traditional sequential editing timelines and analyze how VideoMap could change the way people engage in video editing. Finally, while VideoMap currently caters to primarily maintaining the continuity principle in video editing, sometimes editors may wish to intentionally break this principle and connect two contrasting scenes to create a shocking effect. To support this, we may extend the Paths Explorer to find far-away cuts. Being able to find contrasting yet sensible transitions could also be an interesting exploration area.

4.9 Conclusion

This research explores a new paradigm of video editing based on latent space representations of video frames. We distilled a set of principles grounded on editing practices and theories and built VideoMap, a proof-of-concept latent space video editing interface. We facilitate navigation of the latent space through map-inspired navigational elements and enable switching between different latent spaces using swappable lenses. We implemented three VideoMap

components to assist editors in three editing tasks: exploring video footage, brainstorming suitable video transitions, and rapidly prototyping rough cuts. In a user study with both professionals and non-professionals, we found that VideoMap provides a user-friendly editing experience, reduces tedious grunt work, enhances the overview capability of video footage, promotes editing continuity, and enables a more exploratory approach to video editing. We implemented three extended applications to demonstrate how future developers may customize VideoMap for additional editing use cases. Recently, there has been an explosion of diverse AI foundation models [149] capable of encoding video data into a variety of latent space representations (semantics, 3D depth, human poses, audio, ...) [150]. We hope VideoMap's capabilities can be extended over time through the creation of new AI-powered lenses and support creative ways of co-creating video content with AI on a latent space interface.

Reading VideoMap through the Human-AI-Interface Triangle: the latent space gives a continuous, navigable axis over visual and semantic relationships in the footage (controllability), pan, zoom, and lens switching support the moves the navigate task requires (affordance), and the map metaphor draws on a convention editors already use for spatial layouts, so pan and zoom do what they look like they should do: traverse the space, re-project on lens switch (intuitiveness). Navigation works on a space of existing material. The next chapter takes up composition, where designers build new workflows by chaining model capabilities; Jigsaw asks what representation lets them do that without writing code.

Chapter 5

Jigsaw: Puzzles as an Interface for Composing Model Chains

This chapter presents Jigsaw, which addresses the *compose* task. Most creative AI work involves combining capabilities across modalities. But each model typically lives in its own tool, and stitching them together requires writing code. Jigsaw represents foundation models as puzzle pieces that snap together by matching input and output modalities, letting designers chain them without coding. Within the HAI Triangle, the design task is composing AI capabilities into multimodal workflows, the AI mechanism is pre-trained model chaining, and the interface representation is the puzzle piece.

This chapter was adapted from: Lin, D. C. E., & Martelaro, N. (2024). Jigsaw: Supporting Designers to Prototype Multimodal Applications by Assembling AI Foundation Models In Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems (pp. 1-15).

5.1 Introduction

AI foundation models have advanced rapidly in the past year [148]. These models, which are pre-trained on vast quantities of data, can perform many tasks "off the shelf" without further training. This turns each model into an input-output system and hides much of the complexity of working with AI [151]. Until recently, building an AI-enabled system meant curating data, training a model, and sometimes modifying its architecture for the use case [152].

With powerful plug-and-play capabilities, many designers have begun embracing AI foundation models to enhance their creative workflows. New foundation models support a wide variety of tasks and modalities, including large language models [153] such as GPT [154] for text generation and processing, image generation models such as Stable Diffusion [10],

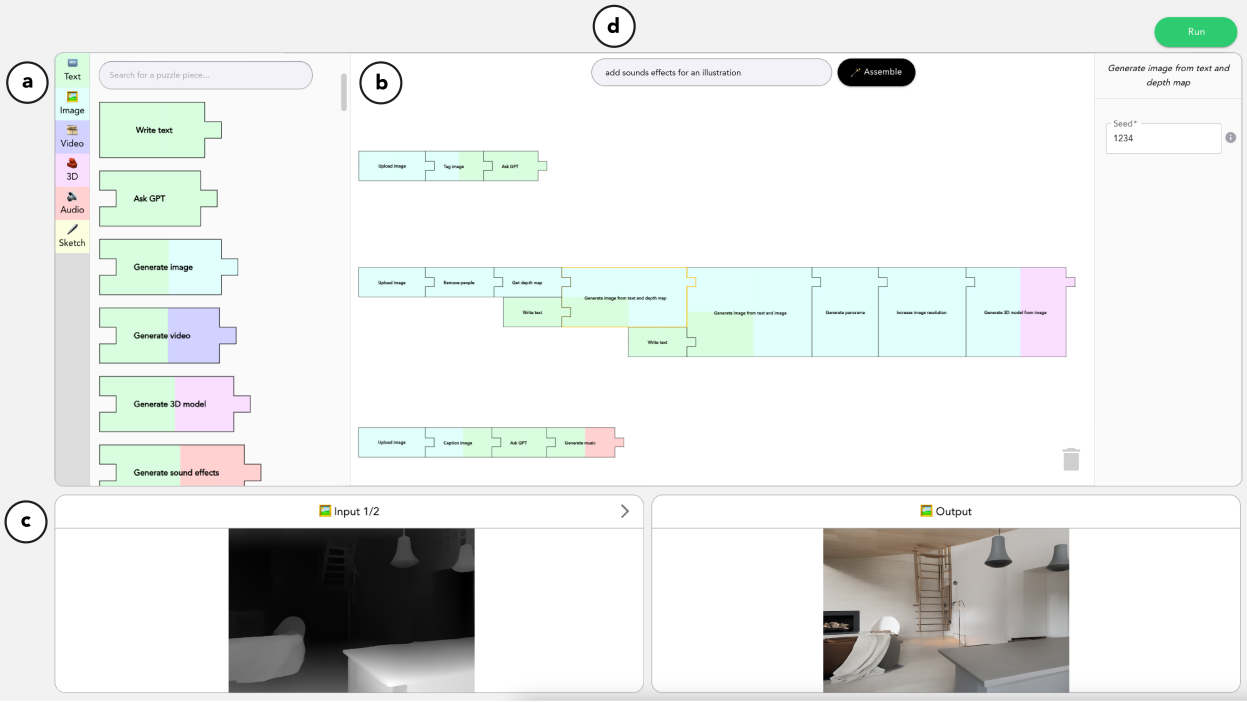


Figure 5.1. Jigsaw is a prototyping system that lets designers generate creative content with AI foundation models represented as puzzle pieces. Designers can first search for model capabilities via the Catalog Panel (a). Designers can drag corresponding puzzle pieces onto the Assembly Panel (b) and combine different AI capabilities across varying modalities into a chain by assembling compatible pieces. Designers can specify inputs and observe the intermediate results of a chain via the Input and Output Panels (c). Designers can ask the Assembly Assistant (d) to recommend a chain of AI models to accomplish a task.

image segmentation models such as Segment Anything [155], and models for video [156], 3D model [157], audio [158] generation.

But integrating foundation models into a creative process is hard. Our initial observations suggest that these models are often used for one-off tasks or as standalone applications. For instance, a designer might use ChatGPT [159] to brainstorm and generate ideas, or they might use Midjourney [160] to generate visual prototypes. To bring results into the rest of their work, they copy and paste between tools. Most designers also stick to a few highly publicized models (ChatGPT, Stable Diffusion, MidJourney) and miss what lesser-known models can do.

To gain a deeper understanding of designers' challenges when using current AI models in their creative processes, we conducted a formative study with ten designers. From our formative study, we identified four key challenges:

1. Designers are often unaware of the full range of capabilities offered by different types of foundation models.

2. Designers struggle with the need to be "AI-friendly," which includes difficulties in forming effective prompts and selecting optimal parameters.

3. Designers find it challenging to cross-integrate foundation models that exist on different platforms and are specialized for different modalities.

4. Designers find prototyping with these models to be a slow and arduous process.

Based on the findings from the formative study, we derived four design goals, which informed the development of Jigsaw, a block-based prototype system that represents foundation models as puzzle pieces and allows designers to combine the capabilities of different foundation models by assembling compatible puzzle pieces together. Jigsaw includes features that help designers discover available foundation model capabilities and find the right model for their use case. Jigsaw also includes "glue" puzzle pieces that translate design ideas into prompts for other models, clear explanations of parameters to help users make model adjustments, and an Assembly Assistant that recommends potential combinations of foundation models to accomplish a task specified by the designer. To assess the utility of Jigsaw, we invited ten designers from the formative study to test the system. We evaluate how well designers create creative AI workflows given a design brief and during free exploration. The results show that Jigsaw helps designers better understand the capabilities offered by current foundation models, provides intuitive mechanisms for using and combining models across diverse modalities, and serves as a visual canvas for design exploration, prototyping, and documentation.

This research thus contributes:

- A **formative study** with ten designers that identifies the challenges designers face when using AI foundation models to support their work.
- **Jigsaw**, a prototype system that assists designers in combining the capabilities of AI foundation models across different tasks and modalities through assembling compatible puzzle pieces.
- A **user study** that demonstrates the utility of Jigsaw to designers and informs areas for future block-based prototyping systems for prototyping with AI foundation models.

5.2 Related Work

This work draws on prior research in AI foundation models, visual programming interfaces, and designer-AI interaction.

5.2.1 AI Foundation Models

The term "foundation models" characterizes an emerging family of machine learning models [148], often underpinned by the Transformer architecture [161] and trained on vast amounts of data. The researchers who introduced this term defined foundation models as "models trained on broad data (generally using self-supervision at scale) that can be adapted to a wide range of downstream tasks." [149] The strength of foundation models lies in their capacity for out-of-the-box usage across various tasks. This signifies an improvement from the previous AI landscape, where users had to create their own datasets for custom use cases and fine-tune models [152].

Prominent examples of foundation models include large language models such as GPT [154] which can perform a variety of text generation tasks and image generation models such as Stable Diffusion [10] which can generate a diverse range of images from text-based prompts. Foundation models also go beyond generative models and include models for tasks such as classification [9], detection [162], segmentation [155], spanning a range of modalities including text [154], image [10], video [156], 3D models [157], and audio [158]. Many foundation models perform tasks *across modalities*, such as text-to-x generative models and x-to-text classification models. In turn, this allows foundation models to be treated as x-to-x input-output systems. Such abstraction greatly simplifies how people can use and combine such models in larger AI-enabled systems. Our research aims 1) to inform designers about the capabilities offered by foundation models that can be useful for creative tasks, and 2) to incorporate these capabilities into their creative workflows. In particular, we are interested

in exploring how designers can *combine* the capabilities of multiple models *across different tasks and modalities* by connecting them together on a visual interface.

5.2.2 Visual Programming Interfaces

Visual programming interfaces (VPIs) have been extensively studied as tools to aid users in designing and implementing systems through graphical elements rather than text-based code [163]. A key benefit of VPIs is their lower entry barrier for novice programmers [164]. There are primarily two main paradigms for VPIs. The first, the dataflow paradigm, lets users specify how a program transforms data from step to step by connecting nodes in a directed graph. Pioneering work in this area includes Prograph [165] and LabVIEW [66]. The second paradigm utilizes block-based function representations and lets users create programs by connecting compatible components together. Notable works in this area include Scratch [166] and Blockly [167]. Many commercial creative applications have adopted VPIs, including game engines such as Unity [168], CAD tools such as Grasshopper [169], and multimedia development tools such as Max/MSP [170].

VPI concepts have been applied to machine learning applications. For example, *Teachable Machine* [171] uses a visual interface to help students learn to train a machine learning model. *ML Blocks* [172] assists developers in training, evaluating, and exporting machine learning model architectures. Very recently, researchers in both academia and industry have worked on VPIs that support the creation of AI workflows through the combination of pre-trained models. Several works have investigated node-based interfaces for building Large Language Model (LLM) pipelines, including *PromptChainer* [57], *FlowiseAI* [173], and *Langflow* [174]. Most closely related to our work are *Rapsai* by Du et al. [175] and *ComfyUI* [176]. Both tools provide a node-based interface for machine learning researchers and enthusiasts to build multimedia machine learning pipelines. These tools are catered more toward users with at least some background knowledge in AI programming, giving users the flexibility to customize the tools through programming at the expense of exposing more technical elements to the user.

Our work builds upon prior and concurrent VPI tools and research. However, we made several design choices for our tool to help better support non-technical designers (Table 5.1). First, our tool leverages a block-based VPI paradigm, which has been shown to be effective in supporting novice programming learners [166]. Second, in the same spirit as other creative AI tools such as *RunwayML* [177], our tool supports AI capabilities of a diverse range of modalities. Third, our tool offers integrated AI assistance features for designers, such as the Assembly Assistant (Section 5.4.4), semantic search (Section 5.4.1.3), and glue pieces (Section 5.4.1.4). We build on recent advances in the reasoning capabilities of LLMs to

Table 5.1. Comparison of Jigsaw against related tools. Jigsaw supports non-technical designers with a beginner-friendly block editor and offers AI capabilities across multiple modalities. Jigsaw's Assembly Assistant can help automatically recommend a chain of AI models for a designer-specified task.

Tool name	Interface design	Supported modalities	Target audience	Assembly assistant	Release year
Jigsaw	**Block editor**	**Text, Image, Video, 3D, Audio, Sketch**	**Designers**	**Yes**	2023
PromptChainer	Node editor	Text	Developers	No	2022
FlowiseAI	Node editor	Text	Developers	No	2023
Langflow	Node editor	Text	Developers	No	2023
Rapsai	Node editor	Text, Image	ML researchers	No	2023
ComfyUI	Node editor	Text, Image, Video	ML enthusiasts	No	2023
RunwayML	Standalone models	Text, Image, Video, 3D, Audio	**Designers**	No	2018
Visual ChatGPT	Chatbot	Text, Image	General public	**Yes**	2023

power these features [178]. To the best of our knowledge, this research is the first to study 1) supporting *non-technical designers* in prototyping design workflows with AI through a block-based visual interface and 2) utilizing the *plug-and-play capabilities of AI foundation models* that have emerged over the past year, covering a *diverse range of tasks and modalities*.

5.2.3 Designer-AI Interaction

Several works from the HCI design community have examined the ways in which designers perceive and interact with AI. Chiou et al. [59] follow a Research through Design (RtD) [179] approach and find that AI can offer designers new perspectives and avenues of design exploration. Shi et al. [180] conduct a landscape analysis of AI and suggest the opportunity to build more tools that enable co-creativity between designers and AI. Yang [181] proposes the vision of designers working with AI as a "design material". This research follows this thread of work to build a tool to help designers prototype new design workflows using AI and with the support of AI.

Subramonyam et al. [182] argue that a challenge with using AI as a design material is that the properties of AI only emerge as part of user experience design. They thus employ data probes with user data to help elicit AI properties and facilitate working with AI as a design material. Yang et al. [183] identify that designers often find designing with AI difficult due to uncertainty about the AI's capabilities and the complexity of the AI's outputs. Gmeiner et al. [184] identify the primary challenges for designers when co-creating with AI design tools as understanding and manipulating AI outputs and communicating design goals to the AI. In this research, we offer mechanisms to help designers overcome these challenges, such as conveying AI capabilities, supporting easy inspection and manipulation of AI outputs with real data, and allowing users to communicate design goals to the AI using natural language.

Liu et al. [185] find that when designers use the "right" prompts, they achieve significantly higher quality results from generative models. However, Zamfirescu et al. [186] find that people generally struggle with writing effective prompts. In this research, we introduce a puzzle piece (translation glue) to help designers automatically translate pieces of text into prompts. Yang et al. [187] find that designers are more successful when they collaborate with data scientists. Using RtD, Yildirim et al. [188] identify that designers develop boundary objects to communicate design intentions with data scientists. In this research, we let designers document their creative process on a canvas (Assembly panel), which designer participants in our user study found to be a useful boundary object for sharing and explaining ideas.

5.3 Formative Study

We conducted a formative interview study with ten designers to understand how designers attempt to use AI in their work and inform the development of a new tool to support creative work with AI.

5.3.1 Participants and Procedure

We interviewed ten designers (P1-P10, 6 male and 4 female, aged 24-39), recruited through known contacts and word of mouth. The designers come from diverse specializations, such as interior design, product design, graphic design, and video game design. All participants have more than five years of design work experience and use AI tools to support aspects of their design processes, such as ChatGPT, Midjourney, and DALL-E [189]. We conducted one-hour interviews remotely over video conferencing, asking participants to describe their typical creative workflow, how they use AI to support their work, the specific AI tools they use, and the pain points they face using AI. Following the interviews, the first author conducted a thematic analysis of interviews and summarized participants' key challenges.

5.3.2 Findings and Discussion

We identify four key challenges designers face when using AI to support their work.

5.3.2.1 C1: Limited Knowledge of AI Capabilities

Despite the broad spectrum of AI foundation models available, designers felt they had limited knowledge of existing models and their capabilities. As a result, they felt like they were underutilizing the creative support these models could offer. Designers found it hard to *"understand the capabilities of various models in a crowded market (P1)"*, which made it

difficult to pick the right model for a task. Designers also wanted to *"easily view a few example results from the models (P5)"* to quickly assess whether a model fits their use case.

5.3.2.2 C2: Tedious to be AI-friendly

Once designers picked a model, they found it hard to be "AI-friendly." This means crafting effective prompts (for generative models) and tuning model parameters to get good results. Designers said it took time to *"master the art of prompt creation (P2)"* and that translating a design idea into a working prompt was itself a major task. As P5 put it, *"behind every stunning image generated by Stable Diffusion lies a designer's patience and a relentless pursuit of the right prompt."* Participants were also confused about how different parameters affect results, leading to *"endless parameter tweaks (P10)"*.

5.3.2.3 C3: Difficult to Combine Multiple Models

Designers felt that current models cater mostly to simple, singular functions. But realistic design workflows involve multiple tasks across many modalities, so designers ended up switching between separate AI platforms. This fragmented the design process. As P9 put it, *"switching between AI platforms felt like needing a different kitchen gadget for every step in a recipe."* Designers also ran into compatibility issues when trying to combine models, leading to time-consuming troubleshooting. They wanted to *"clearly know which models are compatible with one another (P6)."*

5.3.2.4 C4: Slow Prototyping and Iteration

Designers noted that *"seamless prototyping and iteration is crucial to the design process (P6)"*. But when working with AI, they often found it hard to quickly build prototypes and view results. Setting up and switching models is a lengthy process that blocks rapid experimentation. When chaining models, designers also often see only the final result. This makes it hard to understand how each model contributes, and hard to explain design decisions to clients without tangible intermediate outputs.

5.3.3 Design Goals

To tackle the challenges designers encounter when using AI, we distill four design goals.

5.3.3.1 D1: Catalog of AI Foundation Models

To help designers understand what is available, we compile a catalog of existing foundation models. For each model, we provide a plain-English explanation of its capabilities and

example results. We also offer mechanisms for finding models that match a designer's task.

5.3.3.2 D2: User-friendly instead of AI-friendly

We aim to reduce the need for designers to adapt to the quirks of AI models. First, we offer assisted prompting that translates design ideas into prompts. Second, we explain model parameters in plain language, including how changing values affects results.

5.3.3.3 D3: Intuitive Interface for Combining Models

We offer designers an interface that makes it easy to combine task-specific models across many modalities. The interface visually shows which models can connect. We also provide an assistive tool that suggests model combinations.

5.3.3.4 D4: Facilitate Effective Prototyping

Designers place high importance on experimentation and iteration. We make it effortless to try different model combinations and view results. We also let designers inspect intermediate results within a chain to help diagnose errors and communicate ideas to clients.

5.4 Jigsaw System

The following outlines Jigsaw's four major components: the (1) Catalog Panel, (2) Assembly Panel, (3) Input and Output Panels, and (4) Assembly Assistant. We then describe how a designer can use Jigsaw with an example interior design workflow.

5.4.1 Catalog Panel

The *Catalog Panel* assists designers in selecting suitable models for their tasks with a catalog of foundation model components (D1).

5.4.1.1 Curating a catalog of foundation models

We identified six common modalities used in creative work, namely, text, image, video, 3D, audio, and sketches. Jigsaw curates available models across all possible pairwise permutations of modalities (e.g., `text-to-text`, `text-to-image`, `text-to-video`, ...). Jigsaw also includes foundation models that with dual input channels, such as ControlNet [4]. For tasks supported by multiple models, we prioritize models based on: 1) inference speed (ideally less than a minute to run), 2) zero-shot capability (plug-and-play use), and 3) the quality of results (models ranked highly on machine learning benchmarks). Overall, we implemented a catalog of thirty-nine models across six modalities (see Appendix **A.1** for a full listing).

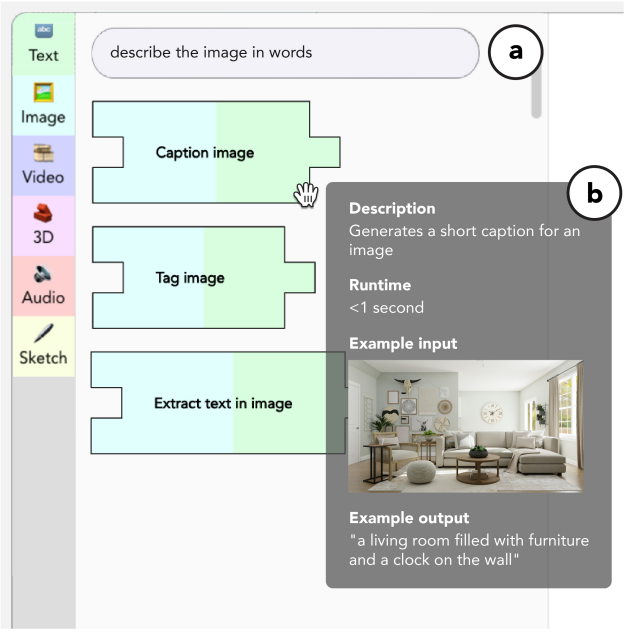


Figure 5.2. Users can search for model pieces by describing their task in the semantic search bar (a). Users can hover over a model piece to view a description of its capability, typical runtime, and an example input and output (b).

5.4.1.2 Representing foundation models as puzzle pieces

Treating foundation models as input/output systems, we represent them as puzzle pieces with input and output arms. There are two types of pieces. The *model* piece represents a model with customizable parameters. The *input* piece accepts user input via text, media file, or sketch (see Section **5.4.3**). Jigsaw color-codes puzzle pieces based on their input and output modalities to signals what pieces are compatible with one another (D3). For example, a `text-to-image` piece would be colored green on the left and blue on the right. When a user hovers over a puzzle piece, a tooltip provides a description of its capability, typical runtime, and an example of an input and output (D1).

5.4.1.3 Helping users find the right piece

Jigsaw provides two mechanisms for users to find model pieces for their tasks (D1): 1) puzzle pieces are grouped by input modality and 2) users can describe the task in the semantic search bar. The search returns model pieces with high semantic similarity to the query, scored using CLIP [**9**] text embeddings.

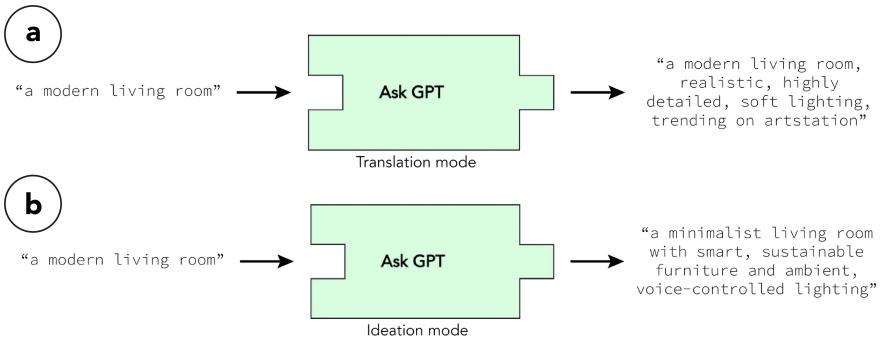


Figure 5.3. The *translation* glue piece converts a piece of text into a prompt format suitable for text-to-x generation models (a). The *ideation* glue piece generates an idea for a design task (b).

5.4.1.4 LLMs as glue

Sometimes models do not align directly — for example, when intermediate reasoning is required. Drawing inspiration from *Socratic Models* by Zeng et al. [190], we use Large Language Models (LLMs) as connectors between model pieces. We call these *glue* pieces. The user first attaches a model piece capable of conveying the content of a modality in text (x-to-text). Next, the user attaches the LLM glue piece for language-based reasoning (text-to-text). Finally, the user attaches a model piece which translates text back into another modality (text-to-x). To help users connect model pieces in common use cases, Jigsaw includes three types of glue pieces:

1. The *custom* glue piece accepts any custom user instruction.

2. The *translation* glue piece converts a piece of text into a prompt that better aligns with text-to-x models (e.g., Stable Diffusion) (D2) (Figure 5.3a). We ask GPT to transform an input into a prompt via the following prompt:

   ```
   Here are example prompts for a text-to-<modality> generation model:
       <list of example prompts>.
   Transform <input data> into a prompt. Answer in only the
       transformed prompt.
   ```

3. The *ideation* glue piece accepts a design task specified by the user and generates an idea (Figure 5.3b). We ask GPT to generate an idea via the following prompt.

   ```
   Generate an idea for <task> based on <input data>. Answer in one
       short sentence.
   ```

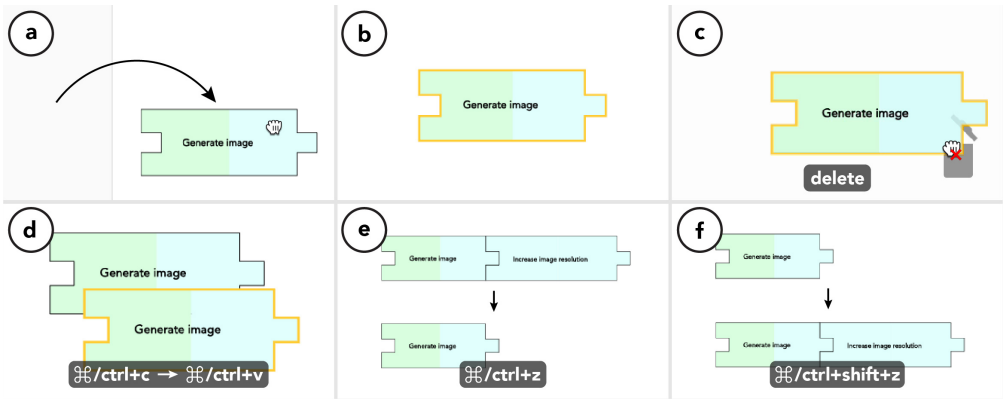


Figure 5.4. Users can drag puzzle pieces from the Catalog Panel onto the Assembly Panel (a), select pieces on the Assembly Panel by clicking on them (b), and remove pieces by dragging them to the trash bin or pressing the delete key (c). Users can duplicate pieces, and undo and redo actions using hotkeys (d-f).

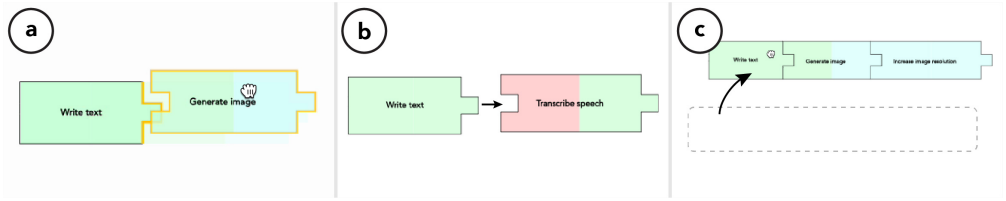


Figure 5.5. When the user drags a puzzle piece close to another compatible piece, Jigsaw displays a semi-transparent preview of the potential connection. If the user releases the puzzle piece, it will snap into place (a). Conversely, if the user attempts to connect a puzzle piece to an incompatible piece, the new piece will be repelled, ensuring that users do not force a fit (b). Users can move multiple puzzle pieces simultaneously (c).

5.4.2 Assembly Panel

The *Assembly Panel* offers an infinite canvas for combining compatible foundation model puzzle pieces (D3) (Figure 5.4). When the user clicks on a model piece, a *parameters sidebar* allows users to customize a model's specific parameters. Jigsaw pre-populates each model's parameters with sensible defaults and bounds the ranges, so users can experiment freely without breaking the model (D2). Tooltips explain, in plain English, how the parameter influences model results and recommends optimal values for common scenarios. Users can build multiple chains on the canvas and can run each chain separately, allowing parallel explorations and complex workflows.

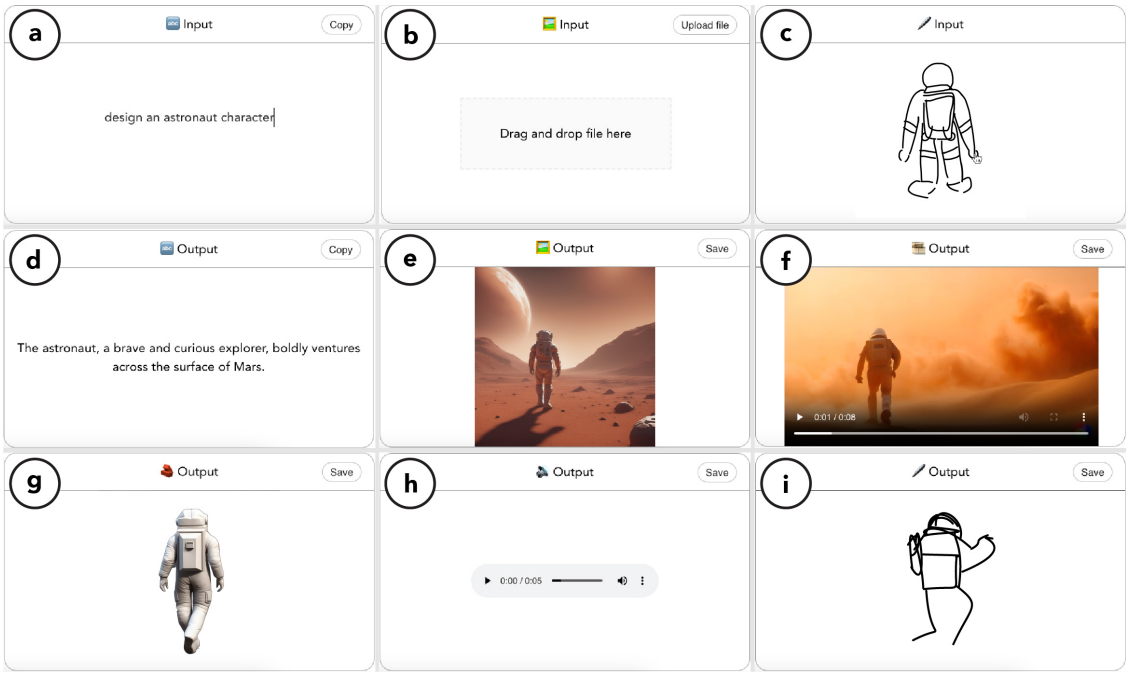


Figure 5.6. Text inputs can be directly typed into the Input Panel (a). Image, video, 3D, and audio inputs can be uploaded either by drag-and-drop or the file browser (b). Sketch inputs can be drawn (c). Text outputs can be viewed and copied by the user (d). Image, video, 3D, audio, and sketch outputs can be viewed in their respective media viewers and downloaded by the user (e-i).

5.4.3 Input and Output Panels

The *Input and Output Panels* allow users to input, view, and download media across modalities. Users can type into the input panel (Figure 5.6a), upload files (Figure 5.6b), or draw sketches (Figure 5.6c). The Output Panel shows the result of the chain and lets users copy (Figure 5.6d) or download outputs (Figure 5.6e-i).

The user can select a puzzle piece to view the intermediate inputs and outputs at that specific piece. Within a chain, the user can also hold shift to select multiple pieces and see how inputs at one piece affect results several steps downstream.

5.4.4 Assembly Assistant

The *Assembly Assistant* recommends a chain of puzzle pieces for a user-specified task. The designer would first provide a natural language description of a task, such as "Add sound effects for an illustration." Jigsaw then asks GPT to use a chain of models to accomplish the task via the following prompt:

```
You are given a set of AI models to complete a user's task.
There are thirty-nine models: <1. text2text() has reasoning capability.
    2. text2img() can generate an image from text. 3. text2video() can
    generate a video from text, ...>
You can only use the models given. You do not have to use all the
    models. You will answer in a JSON format. Here is an example answer:
    <example combination of puzzle pieces written in a JSON format for
    the frontend to parse>. Your task is to <user-specified task>.
```

Prior work has found that asking GPT to evaluate its own results can improve the correctness of its responses [191]. We thus ask GPT to evaluate its own answers based on four criteria via the following prompt:

```
<Prompt from the previous step>
<Answer from the previous step>
Here are four criteria that the answer needs to satisfy. If any
    criteria are not satisfied, please give me the corrected answer in
    JSON format.
1. Whether the user's task was understood and completed.
2. Whether no models outside of the provided ones were used.
3. Whether the output and input of each step can be connected.
4. Whether it follows the correct JSON format.
```

Jigsaw then passes the chain of puzzle pieces provided in JSON format to the frontend and adds them onto the Assembly Panel. The designer can make further edits to the chain, just like any manually-created chain.

5.4.5 System Walkthrough

We illustrate the interactions supported by Jigsaw using an interior design example. Figure 5.7 displays the final arrangement of puzzle pieces, referred to as the "model mosaic" in this paper for succinctness.

5.4.5.1 Ideating a design concept from client material

Zaha is an interior designer tasked with creating and presenting a redesigned interior for a client's new home. She received a photograph of interior from the client. To begin, Zaha plans to create a design concept using the client's photo as a reference (Figure 5.7a). She drags an `Upload image` puzzle piece from the Catalog Panel onto the Assembly Panel, and uploads the client's photo in the Input Panel.

Zaha first identifies existing features in the client's home, such as built-in structures, furniture, and lights. Thus, Zaha uses the semantic search bar in the Catalog Panel to find a puzzle piece that can *"identify the objects inside the image"*. Jigsaw returns the `Tag image`

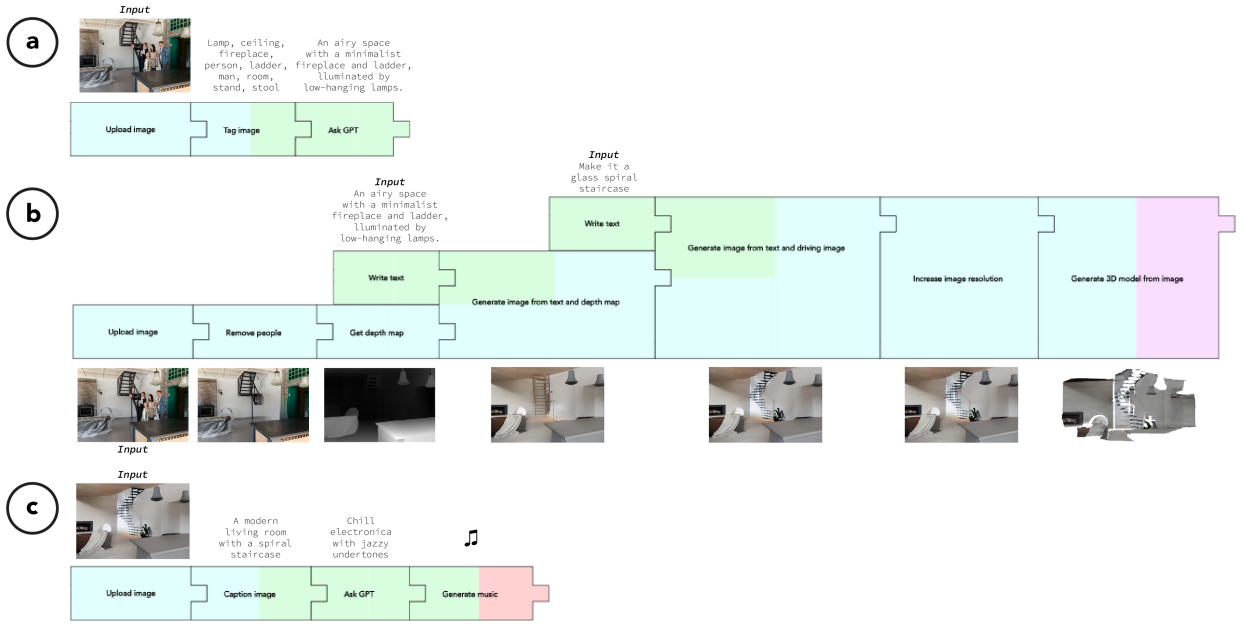


Figure 5.7. An example model mosaic for interior design. The designer can use Jigsaw to ideate a design concept from client material (a), design the 2D and 3D mockups (b), and add music to enhance the presentation (c).

piece as the top result. She adds `Tag image` after `Upload image` to identify the objects. Zaha then uses the `Ask GPT` piece in *ideation* mode, with *"contemporary interior concept"* in the *Task* box, to assist her in brainstorming a concept. After clicking *Run*, Jigsaw has suggested a design concept of *"An airy space with a minimalist fireplace and ladder, illuminated by low-hanging lamps."*

5.4.5.2 Designing the 2D and 3D mockups

With a design concept in hand, Zaha proceeds to create the visual design (Figure 5.7b). Zaha quickly duplicates the `Upload image` piece to start a new chain, using the client's photo as a reference again. She notices that the reference photo includes people, whom she wants to remove, and adds the `Remove people` piece. Zaha is interested in experimenting with AI image generation tools but acknowledges that the room's structure must remain intact for the design to be technically feasible. She uses the semantic search bar in the Catalog Panel to find a puzzle piece that can help *"preserve the structure of the room"*. Jigsaw returns `Get edge map` and `Get depth map` as the top results. Zaha begins by testing the `Get edge map` piece and adds the `Generate image from text and edge map` piece, which takes in both image and text inputs. For text, she inputs the design concept suggested in the previous ideation chain.

Zaha feels that the generated image fails to retain the desired room structure. Recognizing this, she drags the pieces using edge maps into the trash bin. She tries `Get depth map` and `Generate image from text and depth map` instead, which better preserves the room's structure. To try different design variations, Zaha inspects the parameters tooltip for the `Generate image from text and depth map` model in the parameters sidebar. She discovers that she can tinker with the seed value to generate different variations.

Zaha is now satisfied with the redesign, except for the wooden ladder. She believes replacing it with a spiral glass staircase would better fit the contemporary concept. Thus, she searches for a puzzle piece that can modify an image using text instructions and finds the `Generate image from text and image` piece. Zaha instructs it to *"replace the wooden ladder with a glass spiral staircase."* The newly generated redesign now features a contemporary glass spiral staircase.

Zaha would like to visualize a 3D mockup. She discovers the `Generate 3D model from image` piece and attaches this to the chain, but finds that the generated results are low resolution. Thus, Zaha searches the *Image* section of the Catalog Panel for a piece that can help enhance the image. She finds the `Increase image resolution` piece.

5.4.5.3 Presenting the design to the client

To communicate the contemporary design concept, Zaha would like to incorporate a musical background to complement the design aesthetics. To achieve this, Zaha asks the Assembly Assistant to *"help add music based on the image"*. The Assembly Assistant suggests the following chain of puzzle pieces: `Caption image` to understand the image, `Ask GPT` with *ideation* mode to brainstorm a fitting music description, and `Generate music` to generate the music (Figure 5.7c). This outcome is a chill electronic music piece.

5.5 User Study

We conducted a user study to understand how Jigsaw could address designers' pain points in working with multiple foundation models, its potential to be integrated into design workflows, and identify improvement areas.

5.5.1 Participants and Procedure

We invited the ten designers from our formative interviews to participate in a one-hour remote user study. They were not exposed to Jigsaw's system or concept prior to the user study. Participants accessed Jigsaw through a web browser, shared their screen, and verbally explained what they were doing and thinking (think-aloud).

5.5.1.1 Introduction (10 minutes)

Participants provided informed consent, and then received an introduction to Jigsaw's components, as described in Section 5.4.

5.5.1.2 Reproduction Task (15 minutes)

Participants were asked to reproduce the interior design model mosaic described in Section 5.4.5. Participants used the starter image shown in Figure 5.7a and a detailed design brief of the various steps they would need to create (see Section 5.4.5).

5.5.1.3 Free Creation Task (20 minutes)

Participants were asked to freely explore Jigsaw and create their own model mosaics. We encouraged participants to build workflows beyond a simple chain and try out puzzle pieces involving multiple modalities.

5.5.1.4 Post-Study Interview (15 minutes)

After the creation activities, we conducted a semi-structured interview asking about participants' experiences using Jigsaw, whether they could see Jigsaw being integrated into their design workflow, and to identify areas for improving the system.

5.5.2 Results, Discussion, and Future Work

All participants completed the reproduction and free creation tasks. The qualitative themes below describe what Jigsaw enabled, what participants struggled with, and where they suggested improvements.

5.5.2.1 Helping designers discover and utilize AI capabilities

Participants located AI capabilities via the Catalog Panel's semantic search bar or by filtering pieces by modality. Many participants mentioned that they were able to *"discover new AI abilities [they were] previously unaware of (P9)"*. For example, P2, an illustrator, discovered the capabilities of ControlNet [4], a model that allows users to add additional control to a text-to-image model, such as a guiding sketch. Figure 5.8 shows the model mosaic created by P2, who used Jigsaw to create an audio-visual story. In Figure 5.8a, she created visuals for her story. Instead of using a text-to-image model, she discovered and used ControlNet to generate images based on a starting sketch. In total, participants used 8 model puzzle pieces on average (μ=7.9, σ=1.29), and all participants explored beyond well-known models such as GPT and Stable Diffusion (see Appendix A.2 for details). As the number of foundation

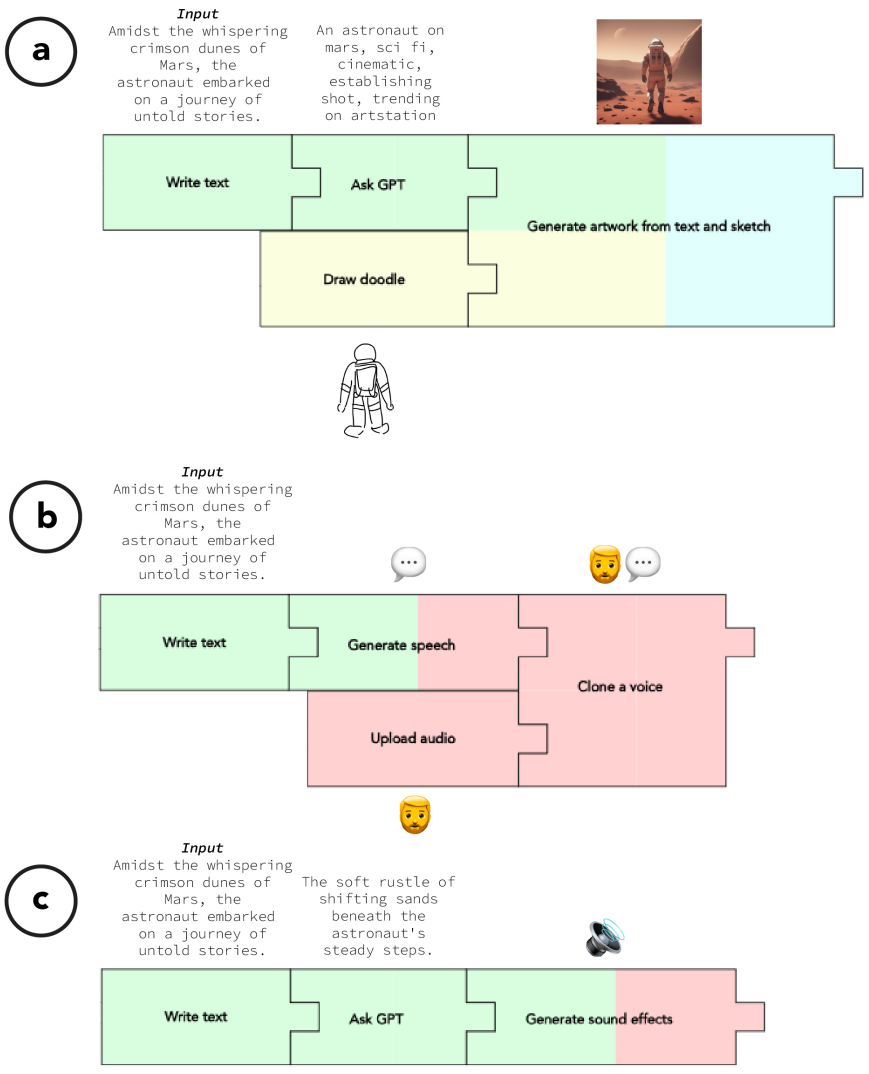


Figure 5.8. Model mosaic by P2, an illustrator, to create an audio-visual story. P2 uses Jigsaw to create an illustration based on a text description and a reference sketch (a), generate narrations through a cloned voice (b), and generate accompanying sound effects (c).

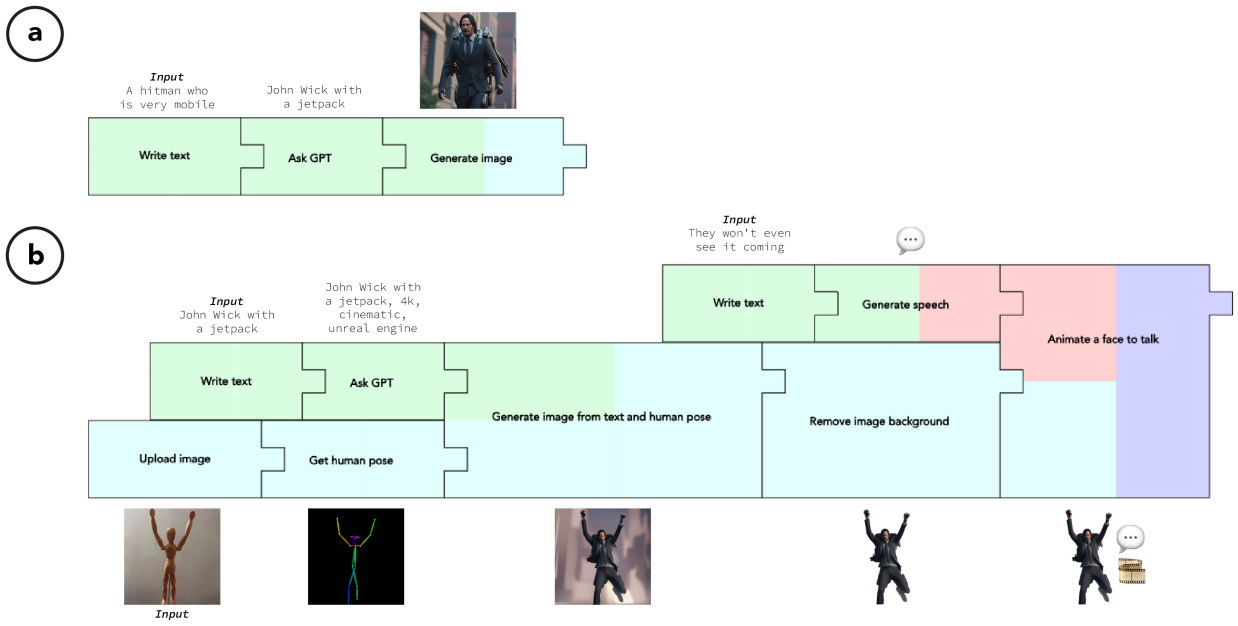


Figure 5.9. Model mosaic by P10, a game designer, to create a video game character. P10 uses Jigsaw to create a character concept and preview the character's visuals (a). P10 then specifies a pose for the character and generates the character's visuals in the specified pose (b). P10 then generates a line for the character to say and animates the character to deliver the line.

models continues to increase, we plan to expand our set of puzzle pieces for designers over time.

Participants commented that tooltips *"gave [them] a solid understanding of the capabilities of each of the puzzle pieces, like what can be expected as output and what types of inputs are suitable (P1)"*. In addition, participants expressed that the assisted prompting mechanism offered by the *translation* glue piece *"allowed [them] to achieve satisfying results without the need to laboriously rephrase and tweak prompts (P2)"*: *"Now, it's like I speak the AI's language. (P5)"*.

5.5.2.2 Supporting intuitive prototyping

Participants expressed that *"[they] enjoyed the idea of building with AI visually with tangible puzzle pieces (P10)"* and found it *"easy to pick up and start designing (P3)"*. In particular, participants appreciated the error-proof design: *"Knowing which puzzle pieces can be connected expedites my prototyping. I see the same colors and receive snapping feedback. I don't spend time building a workflow and then compile it to find compatibility errors (P3)"*. A contribution of this research is showing how the design benefits of block-based VPIs, commonly catered towards novice programming learners [**166**, **167**, **192**], can be effectively applied to the realm of design prototyping for non-technical designers to work with AI capabilities. An interesting extension of Jigsaw could be the implementation of a "tutorial mode" to teach novice designers. The system would disassemble a model mosaic created by an experienced designer into pieces, allowing a novice designer to recreate it and learn from the experienced designer's design process.

Furthermore, participants mentioned that the ability to near-instantaneously see intermediate outputs in a chain *"helped [them] to quickly test out ideas and make adjustments to individual steps as needed (P5)"*. This aligns with findings from prior research in interactive program debugging tools [**193**–**195**].

5.5.2.3 Serving as a brainstorming and documentation canvas

We observed participants making creative uses of the Assembly Panel, including using it to test different partial workflows before combining them into more complete workflows and using it to document their design explorations. Participants expressed that the Assembly Panel provides *"a playground to be messy and experimental (P3)"* and *"makes it easy to track the evolution of an idea (P4)."* Moreover, participants commented that the *ideation* glue piece was *"helpful for brainstorming concepts at the beginning [of the design process] (P2)"*. We observed that designers occasionally passed the outputs of the *ideation* glue directly into a generation model, as shown in Figure **5.8**c. In other instances, designers

maintained a shorter chain solely for concept generation. This is shown in the model mosaic created by P10, a game designer, in Figure 5.9, who created a video game character. He primarily used the short chain in Figure 5.9a to generate concepts for his character. We observed that designers frequently created multiple chains to organize different stages of their design process. Participants noted that since the canvas documents their creative process, it could serve as *"a boundary object for sharing and explaining ideas to clients (P5)"*. Moreover, participants commented that the Assembly Assistant was useful in *"generating an initial configuration of puzzle pieces to start working with (P1)"*. This aids in combating the *"blank canvas syndrome (P6)"*, a common occurrence at the onset of a creative activity [196]. In Figure 5.9b, P10 wanted his video game character to look like Superman flying. However, he initially struggled to come up with a method to accomplish this, so he sought assistance from the Assembly Assistant. The Assembly Assistant recommended a workflow of a reference pose image, a pose extraction model, and a ControlNet model that can be guided by pose. Given this workflow, P10 used a wooden mannequin to specify the pose for his character.

5.6 Limitations and Future Work

There are several avenues for improvement that we plan to address for future work. First, we currently implement one AI model per design task (e.g., Stable Diffusion for text-to-image). We plan to support switching between alternative models. We will surface tradeoffs (e.g., speed vs. quality) for both the user and Jigsaw's subcomponents (semantic search and Assembly Assistant), enable side-by-side comparison, and let users filter models by criteria like "text-to-image models with a typical runtime under 10 seconds." Second, we are interested in expanding Jigsaw to let designers define custom puzzle pieces, as suggested by P10, and logic operators such as if/else statements and loops, common in other VPIs [170]. Third, we currently use LLMs as glue, using the text modality for intermediate reasoning (Section 5.4.1.4). We anticipate extending the glue piece to incorporate newer research on multimodal LLMs (MLLMs), such as GPT-4V [197] and LLaVA [198], to add information from additional modalities. Fourth, we plan to expand the Input and Output panels to handle real-time video and audio streams, as suggested by P9. Finally, participants noted that the Assembly Assistant was less robust to ambiguous tasks or tasks that require very complicated mosaics. As the Assembly Assistant uses GPT, we anticipate improvements to the Assembly Assistant as stronger versions of GPT are released. This is also a common challenge recognized by recent machine learning works that also aim to *automatically* combine expert models to solve complex AI tasks [178, 199, 200]. A path forward, as suggested by

P7, could be to improve the Assembly Assistant to support *back-and-forth interactions* with the designer, becoming a design co-pilot that assists designers in creating complex workflows. As more designers use Jigsaw to create model mosaics, we plan to compile them into a template gallery that other designers can modify for their own use cases, as suggested by P8. We believe that the accumulated design templates can serve as a search space for the Assembly Assistant, enhancing its capabilities as a design search engine.

5.7 Conclusion

This research identifies the challenges designers face when using AI foundation models to support their work. The research prototype, Jigsaw, uses a puzzle piece metaphor to represent foundation models and allows the combination of models by assembling compatible pieces. Feedback from designers using Jigsaw demonstrated that designers discovered new AI capabilities, combined multiple AI capabilities across various modalities, and flexibly explored, prototyped, and documented AI-enabled design workflows. We are interested in extending Jigsaw with more capabilities and hope that this research can help inform future research on block-based prototyping systems for prototyping with AI foundation models.
Reading Jigsaw through the Human-AI-Interface Triangle: foundation models with typed inputs and outputs give a composable mechanism designers can chain end-to-end (controllability), drag-and-snap puzzle pieces support the moves composition requires (affordance), and the puzzle metaphor draws on a convention designers already understand, so type-coded ports and snapping behavior do what they look like they should do (intuitiveness). Composition assembles a pipeline from discrete pieces. The next chapter shifts to iteration, where the designer refines a single design through a tight feedback loop with the model; Inkspire pairs sketching with structural conditioning to keep that loop fast.

Chapter 6

Inkspire: Sketching as an Interface for Iterating with Structural Guidance

This chapter presents Inkspire, which addresses the *iterate* task. Designers iterate through sketching, but text-to-image tools break the iterative loop by treating each prompt as a discrete generation rather than a step in an ongoing conversation. Inkspire pairs sketching as the interface with ControlNet's structural conditioning, and converts each AI-generated design back into a low-fidelity sketch scaffold the designer can build on. Within the HAI Triangle, the design task is iterating through sketching, the AI mechanism is structural guidance via ControlNet, and the interface representation is the sketch.

This chapter was adapted from: Lin, D. C. E., Kang, H. B., Martelaro, N., Kittur, A., Chen, Y. Y., & Hong, M. K. (2025). Inkspire: Supporting design exploration with generative AI through analogical sketching. In Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems (pp. 1-18).

6.1 Introduction

We have seen significant progress in the capabilities of text-to-image (T2I) models, many of which are now able to generate realistic images using text [201]. These models not only accelerate the process of converting thoughts into visuals but can also potentially create serendipitous inspirations for users [202]. Recent research has also opened up new possibilities for translating one image representation into another, such as transforming a sketched drawing into detailed designs [4]. Consequently, many designers have begun embracing the use of T2I models to enhance their creative work.

However, despite the proposed benefits T2I models, integrating them into the designer's creative workflow can be challenging. In particular, recent works have observed that designers experience *a high level of fixation* [203] when using generative AI [204] leading them to

explore fewer novel ideas than may otherwise be beneficial for innovation. For instance, a designer might write a text prompt and hit generate. Upon viewing the result, they might adjust a few words in their prompt and hit generate again [205]—repeating this process akin to using a slot machine [206]. Their final prompt tends to be conceptually similar to their original one, often with small, incremental modifications aimed at getting the AI to create their *initial design intention*, instead of exploring new diverse design spaces [204]. As found in prior work, unsupported text-only prompting can be a limiting interface for generating good outputs from GenAI [186].

To understand how professional designers might experience these challenges with GenAI in their own processes, we conducted a comprehensive day-long exchange session with a team of professional designers from a large automotive company. The team outlined their design process, from conceptualizing a product design sketch based on a specific guiding theme to presenting final designs to stakeholders. From our discussions with the designers, we identified three key challenges in their use of GenAI:

C1 Designing through text prompts feels *unnatural* as compared to their traditional sketching and ideation process.

C2 T2I models struggle with generating inspiring designs from *abstract concepts* (e.g., "protective" car), a technique the team uses to create more novel designs.

C3 It is difficult to directly build on generated designs—they appear *"too complete"*, potentially leading to fixation.

To help designers working with GenAI avoid design fixation, we might take inspiration from design research working to break designer fixation. Two common strategies in line with what we learned from the professional design team include analogy-driven design [207–209], where concepts from outside the domain area are used to foster inspiration in the domain area (addressing C2) and providing multimodal interfaces that work visually and with lower-fidelity assets (addressing C1 & C3).

Several recent works in HCI have focused on developing new interfaces for GenAI to help designers explore a larger design space, including interactive prompting which aims help people who often have limited prompting ability [210], multimodal search [211], and visually navigating a 2D latent space interface [212]. Nonetheless, being able to help designers move beyond one thread of thinking and explore a wider design space while still taking inspiration and *building on* generated designs remains a significant limitation preventing GenAI to be fully integrated into designers' work. While the ML community offers methods to increase editing freedom of generated images, including local inpainting [213] and instruction-based

image editing [214], these methods remove designers away from their natural rhythm of interaction with ideas.

In this paper, we introduce a workflow of continuous exploration with T2I models, encouraging designers to adopt a mindset that facilitates more iterative and exploratory design generation. Specifically, we built Inkspire, a proof-of-concept tool that provides a more familiar interaction built around iterative sketching and by leveraging the concept of analogical design to facilitate inspiration around design ideas. Our tool integrates analogical inspiration to promote concept-level ideation from abstract themes, allowing designers to recognize creative possibilities without needing to come up with them and write prompts manually. This reduces cognitive friction, enabling fluid exploration of new ideas [215]. Additionally, inspired by prior works on drawing assistance systems that convert photographs into sketches to teach people how to draw [216], we introduce a new mechanism that converts high-fidelity AI designs into high-quality but low-resolution sketch scaffolds, directly underlaid on the designer's canvas. The scaffolding provides a transparent view into the current state of the AI and helps designers build on AI designs without being overly fixated on photorealistic renders. Finally, we enable a new design generation every time a new pen stroke is drawn (with near real-time performance), encouraging designers to consider small but meaningful changes in form and refinement with each pen stroke, thereby creating many more opportunities for designers to explore new directions. All of these components are designed to be seamlessly integrated into the sketch process familiar to designers.

To understand how designers use Inkspire, we invited both professional designers and novice users to design everyday products. We also asked them to use a baseline condition of a state-of-the-art ControlNet [4], which allows designers to use sketches and user-written prompts to guide generative design, but does not provide our proposed analogy inspirations or convert designs into lower resolution sketches, and requires the user to choose when they generate explicitly. The results show that users rated Inkspire as providing significantly more inspiration and exploration over baseline ControlNet. The interaction that users had with Inkspire was drastically different than using a baseline ControlNet—designers using Inkspire generated many more concept sketches and showed a process that appears more co-creative, whereas when using ControlNet they focused on manual sketching and refining prompts before handing off to the generative system. Inkspire enabled designers to more effectively co-create designs with T2I models with significantly increased partnership, controllability, communication, and sense of attribution over their creations.

This research thus contributes:

- **Inkspire**, a proof-of-concept, iterative sketching tool that helps ground abstract concepts into analogies and converts AI-generated images into abstracted sketch scaffolds

to support fluid design iteration and avoid fixation.

- A **within-subjects study** showing that Inkspire embodies a more iterative and exploratory workflow with T2I models, with designers rating Inkspire with significantly higher inspiration, exploration, and attributes of co-creation.

6.2 Related Work

This work builds on prior research in human-AI co-creativity for helping people generate ideas [217]. Specifically, we build upon prior works aiming to help designers create new visual concepts through text to image models and visual inputs [218–220]. We review how generative AI systems have been shown to increase idea generation, but counterintuitively can lead to more design fixation. We then review possible solutions for overcoming design fixation including analogy-driven design, computational sketching tools, and reducing the fidelity of generated images.

6.2.1 Generative AI and Design Fixation

One of the proposed benefits of generative AI is to help designers avoid *design fixation*, where designers remain stuck in a single line of thinking and a limited set of ideas, thus limiting the conceptual novelty of ideas and potential for innovation [203]. While many of the works suggest that generative AI can help people generate more ideas, potentially helping them to move into new conceptual spaces, recent experimental work has found contradictory results on generative AI's impact on design fixation. DesignAID [221] leverages large language models and image generators to help people explore a large, diverse space of ideas and was found to provide more inspiration than search-based tools. However, people rated the generative AI ideas as less valuable, with the paper authors suggesting that the ideas may not have been diverse enough, not well matched to the problem, or just not enough to break people design fixation. Bordas et al. [222] find that people using ChatGPT 3.5 to help generate ideas to protect an egg falling from 10 meters increased their idea generation but remained overly fixated on specific solutions. Wadinambiarachchi et al. [204] find that participants designing a chatbot avatar concept with a text-to-image generator showed significantly higher design fixation, with fewer ideas, less variety, and less originality than participants using an image search engine or working unassisted. The authors suggest that people's limited prompting ability, a known issue when lay people try to use LLMs [186], led them to simply copy keywords from the design brief, limiting the language used to generate the images. Davis et al. [212] and Zhang et al. [223] report similar findings, where designers

working with text-only image generators showed limited creative exploration, again due to people's limited abilities with text prompting.

To overcome the fixating issues of text-based generative systems, Davis et al. [212] developed a generative system taking example images of dresses and provided graphical buttons and sliders to generate more realistic vs. creative ideas and alter shape, color and texture. This visual generative system was preferred by designers and lead to more creative idea explorations. However, the high-fidelity images presented by many image generation tools, which Wadinambiarachchi et al. [204] suggest could lead to more fixation based on prior design research showing that high-fidelity images lead to more fixation over rougher sketches [224, 225].

In our work, we explore how to overcome the fixating issues that many generative AI systems have today. First, we look to help designers explore more novel ideas by providing interfaces that overcome their limited prompting abilities and help them generate more novel ideas. For this, we look to analogy-driven design as a potential solution. Second, we move away from text-only generation and provide more visual interfaces for design exploration (C1), helping better match how designers come up with ideas. We explore how iterative, computational sketching can be used as an visual input to image generation. Third, we break from showing only *high-fidelity* images which could lead to design fixation and explore how AI generated images can be altered and presented to designers in *lower resolution* forms to see if such representations may scaffold new ideas.

6.2.2 Analogy-Driven Design

Analogy-driven design, or design-by-analogy, is an approach to drawing inspiration from a known domain, including concepts and products, to find novel solutions to a target domain. Many text-based research systems and prototypes, such as DANE [226], Idea-Inspire 4.0 [227], and BioTRIZ [228], support design ideation by retrieving analogies and mapping them onto a target problem based on inferred similarities.

Recent proliferation of text and image data repositories combined with advances in vision and language models gave rise to new multimodal approaches that expand analogy-driven design to the visual domain [209]. For instance, Kwon et al. developed an approach that leverages visual similarity to discover visual analogies for generating new ideas [229]. In addition, Zhang and Jin proposed an unsupervised deep learning model, Sketch-pix2seq, to extract shape features from Quickdraw sketches, creating a latent space that enables defining visual similarities and searching for analogical sketches [230]. Jiang et al. developed a CNN-based model to create feature vectors representing patent images, which combine visual and technological information to enhance visual stimuli retrieval [231]. While these models

present promising methods to support image-based analogy-driven design, the specialized nature of these models reduces their practical appeal to designers seeking exposure to out-of-distribution inspirations.

Large pre-trained models provide exciting opportunities to support domain-agnostic analogy based image retrieval. While T2I models alone struggle to generate images from an abstract concept such as "mystery" [232], machine learning research has demonstrated the use of LLMs to convert abstract concepts into semantically meaningful physical representations, thereby streamlining the process for downstream T2I generation tasks [233–235]. For instance, Fan et al. introduced an approach that extends an abstract concept such as peace with concrete objects (e.g., white doves, olive branches), then rewriting the original prompt to incorporate the objects in a scene [234]. Liao et al. proposed the Text-to-Image generation for Abstract Concepts (TIAC) framework that builds on a three-layer artwork theory to clarify the *intent* of the abstract concept with a detailed definition, then transforming it into semantically-related physical *objects* and concept-dependent *form* [233]. However, much of this work focused on scene illustration with multiple objects. Moreover, techniques that rely on automated prompt enrichment reduce the steerability of T2I models. In Inkspire, we apply a similar process through the use of LLMs and and analogical reasoning to convert an abstract concept into individual physical objects from across multiple domains. We investigate how providing users with a menu of analogical concepts can help the user rapidly generate diverse analogical-grounded designs with T2I models, to overcome the current challenges of creating inspiring designs from abstract design prompts (C2) in ways that also offer them more control over design space exploration.

6.2.3 Guided Sketching

Research has explored various computational techniques that provide sketch guidance to aid in several use cases including skill building, serving as reference points, and encouraging creative exploration.

Many existing systems have explored guided sketching tools that help novices learn how to sketch. ShadowDraw [216] offers real-time shadow-based feedback, while systems such as The Drawing Assistant [236] and Painting with Bob [237] focus on translating photographs to sketches. Several works have explored crowdsourcing-based approaches. Systems like Limpaecher et al. [238] and Sketchy [239] leverage collective human knowledge to provide guidance. These systems excel at teaching specific drawing techniques but are less suited for open-ended creative exploration. Similarly, portrait-specific systems such as DualFace [240], which employs a two-stage drawing guidance for freehand portraits, and PortraitSketch [241], which provides face sketching assistance, demonstrate the value of domain-specific guidance

but are constrained to a narrow use case.

Most relevant to Inkspire is Creative Sketching Partner [242], which retrieves sketches that are visually and conceptually similar to the user's sketches as a means to stimulate exploration and inspire new designs. This system demonstrates the potential of computational guided sketching systems to inspire new designs. However, it has the limitation of relying on existing sketch databases, whereas in this work, we leverage the generative capabilities of AI.

In our work, we draw inspiration from empirical research by Williford et al. [243], whose analysis of 240 concept sketches revealed that ambiguous sketch underlays could reduce fixation on conventional forms and promote divergent thinking, guided sketching techniques such as ShadowDraw [216], and existing design practices of sketch scaffolding [244]. With Inkspire, we propose a novel computational pipeline for converting GenAI designs into abstracted, yet high-quality sketching scaffolds that underlay the user's canvas. To the best of our knowledge, this work is the first to support the full closed-loop-cycle of sketching GenAI designs and abstracting GenAI designs into sketches. Through this technique, we address the challenge of building on GenAI images that are "too complete" (C3) by drawing designers' attention away from the high-fidelity generated image and inspire designers to iterate on top of the silhouette of GenAI designs with low friction.

6.3 Formative Session with Design Professionals

To understand how professional designers use Text-to-Image (T2I) models in their work, we conducted a day-long exchange session with a team of seven professional product designers from a large automotive company. The designers in our exchange session work at a top 5 automotive manufacturing company and cover multiple disciplines of training spanning a wide range of roles in the company, including creative director, modeling lead, interior designer, exterior designer, artistic creator, conceptual lead creator, UX/UI and strategy. There is significant cross-collaboration across departments globally and with other companies in the industry. The designers also have experience in using T2I tools such as Midjourney [160] and Vizcom [245]. The designers showed our team their design processes in presentations with specific examples from their past work. This included sketches, concept boards, and various documentations of collaborative meetings for a wide range of mobility concepts. The designers also showed their process of using current T2I tools, and we discussed new ideas on how to design tools to support them. From our interaction with the designers, we identified key challenges they face when using T2I models and summarized them into three design goals to inform the development of Inkspire.

6.3.1 Design Goals

Design Goal 1. Sketching as a Natural Method of Interaction.
Designers emphasized that prompting is an unnatural approach to designing. They expressed difficulty in effectively conveying design ideas through language [186]. They often felt constrained by the need to craft comprehensive prompts, which limited their ability to explore a wider range of ideas. In contrast, designers expressed a preference for approaching design tasks through *sketching* [6], often beginning with just a simple line or silhouette. Therefore, our first design goal is to allow designers to interact with AI via sketching as a natural method of interaction. We aim to support designers in starting with simple abstract lines and assist them in progressing towards complete sketches.

Design Goal 2. Visually-Concrete Inspirations.
Designers mentioned that design briefs are typically inherently abstract, for example, "design a vehicle that conveys a sense of protectiveness". However, they found T2I models to generally produce poor and generic results when prompted with such abstract terms [246]. Even when resorting to prompt engineering tricks, they find it challenging to visualize abstract concepts in concrete forms. Therefore, our second design goal is to assist designers in visualizing abstract design themes through visually-concrete inspirations. Inspired by the way designers may draw inspiration from nature [247], we aim to recommend analogical inspirations [248] to designers and make it easy for them to quickly visualize a variety of diverse inspirations.

Design Goal 3. Complete the Feedback Loop.
Designers expressed difficulty in iterating on AI-generated designs and they described the process of using T2I models as a one-way process. They feel that the generated designs look "too complete", making it difficult for them to envision new ways to build on them. Often, designers find themselves in a position where they can either choose to use a design or discard it entirely. Therefore, we aim to bridge the gap in the feedback loop – while T2I models transform ideas into images, our goal is to *transform images back into abstractions (i.e., sketches), to allow designers to continue the iteration process.*

6.4 Inkspire

We first illustrate how a user would use Inkspire through a concept car design example. We then describe the technical implementation of Inkspire, which consists of two primary components: Sketch2Design (generating AI designs from sketches and analogies) and Design2Sketch (converting AI designs into lower fidelity sketch scaffolding).

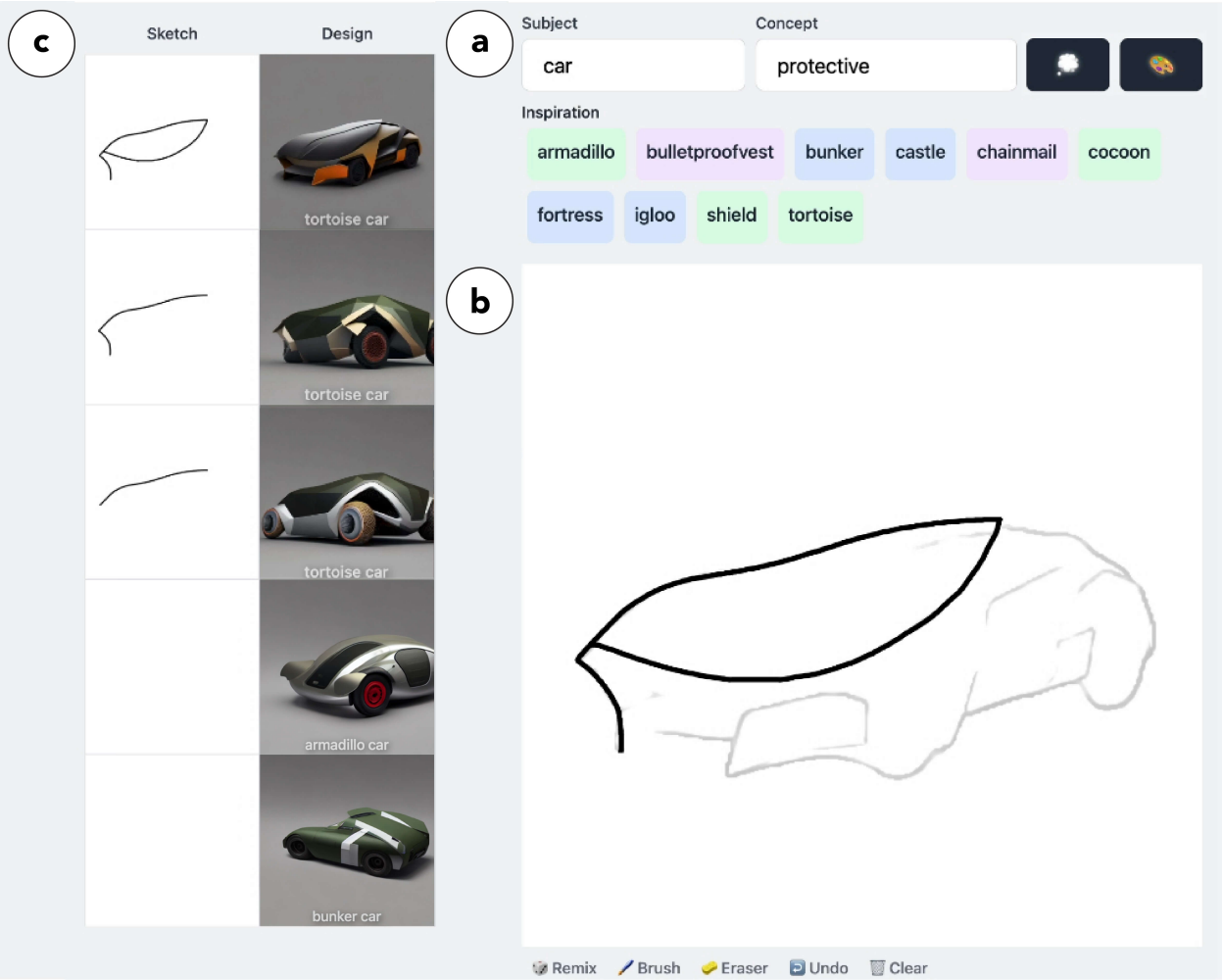


Figure 6.1. The Inkspire interface. The designer may use the Analogical Panel (a) to ideate analogical inspirations for abstract concepts (e.g., "protective car" → "tortoise car"). The designer may sketch on the Sketching Panel (b) to iteratively guide AI design generations. For each iteration, we display a sketch scaffolding under the canvas. This scaffolding is created through abstracting AI designs into lower fidelity. Finally, the designer may view the history of iterations on the Evolution Panel (c).

6.4.1 System Walkthrough

DeLorean is an automotive designer tasked with creating a concept car design that embodies a sense of "protectiveness" (Figure 6.1).

Ideating a Design Concept.

To begin, DeLorean uses the *Analogy Panel* (Figure 6.1a) to generate visually-concrete inspirations for the abstract concept (Design Goal 2). He uses "car" as the subject and "protective" as the abstract concept, then clicks on the inspiration button. He is presented with a selection of inspirations, color-coded by categories: nature, fashion, and architecture. DeLorean experiments with several inspirations (such as bunker, armadillo, tortoise) by clicking on them. For each inspiration he selects, the AI generates a design in the *Evolution Panel* (Figure 6.1c). He can manually edit inspirations in the concept box and manually click on the generate button. DeLorean decides to use tortoise as his inspiration.

Iterating on Designs through Sketching.

DeLorean now iteratively guides the AI through sketching in the *Sketching Panel* (Figure 6.1b) (Design Goal 1). He starts off with a simple silhouette line, and the AI generates an initial design in the Evolution Panel. In the Sketching Panel, DeLorean sees a *scaffolding* abstracted from the initial AI design (Design Goal 3). This allows him to take inspiration from the AI designs without being overly fixated on photorealistic renders. He is drawn to the bold curve of the windshield area shown in the scaffolding and loosely traces this part to add to his sketch. The AI then generates a new design and scaffolding.

DeLorean repeats this back-and-forth process with the AI (Design Goal 1), continuing to iterate on his designs through sketching until he achieves a design that satisfies him. To navigate between iterations, restart a sketch, or refine parts of the sketch, he uses the *Undo*, *Clear*, and *Eraser* tools. To explore design variations using the same input (the same inspiration and sketches), he uses the *Remix* tool.

6.4.2 Sketch2Design

The *Sketch2Design* component helps users brainstorm design concepts and generate product designs through sketching (Figure 6.2). First, the user specifies the subject that they are designing for (e.g., car) and an initial abstract concept (e.g., protective). To brainstorm more concrete design ideas for the abstract concept (Design Goal 2), we leverage Large Language Models (LLMs) (GPT-4) [154] to generate analogical inspirations (Figure 6.2a). We leverage prior techniques in chain-of-thought reasoning [249] to break down the problem of creating analogies based on the given concept word. We take a two-step prompting approach shown in the listings below. We first prompt the LLM to detail the design principles for the given

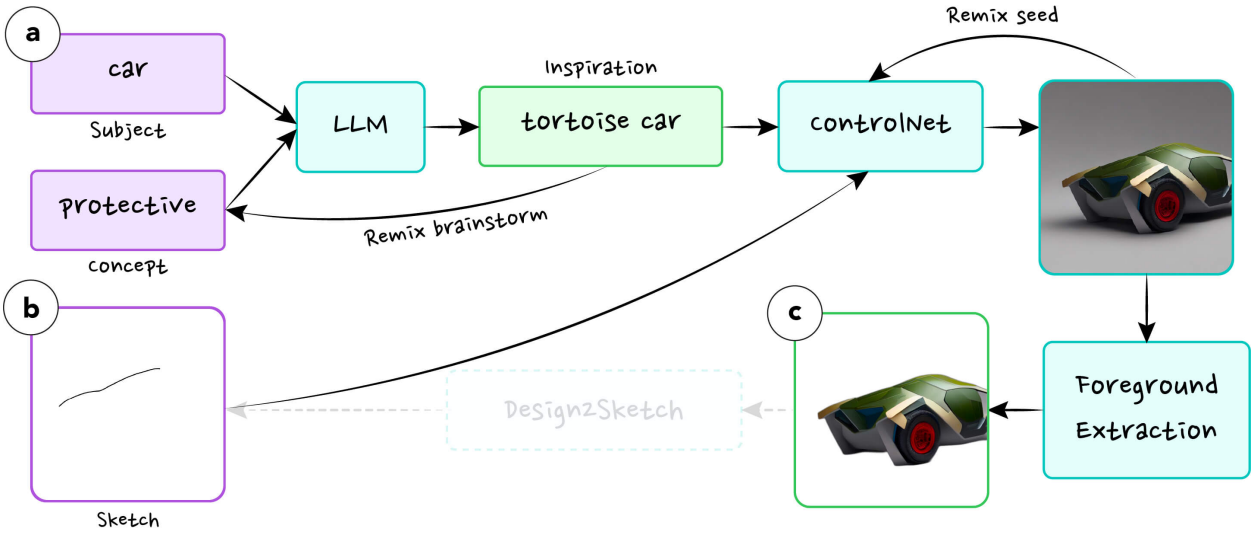


Figure 6.2. Sketch2Design pipeline, including (a) inspiration generation with LLMs, (b) sketch-guided design generation, and (c) foreground extraction.

subject (e.g., car design).

> `Describe the key design principles in <subject> design in one short paragraph.`

Such design principles can be useful as context [250] to ground the LLM to ideate inspirations that are more suitable for the specific product domain. An example intermediary result from this step for the domain of car design could be:

> `Key design principles for car design include aerodynamics exteriors for fuel efficiency and performance...`

Given the design principles, we then prompt the LLM to generate analogical inspirations. Our definition of analogy draws on the work of Gentner [251]. This definition involves identifying parallel relations from a source domain to apply to a target domain even when their surface features differ. We have structured our prompt using this definition of analogies by finding visually concrete objects from three source domains (nature, architecture, fashion) that convey concepts of the target domain (abstract concept).

> `You are a <subject> designer. The design principles in <subject> design are as follows: <design principles from Step 1>. Brainstorm analogical inspirations for <subject> design to convey a sense of <concept> from one of the following domains: nature, architecture, or fashion. Answer`

```
in a bullet-point list of 10 items (item1\nitem2...\nitem3) using visually-concrete
objects not adjectives and don't repeat.
```

We empirically found that prompting specifically for the domains of nature, architecture, and fashion leads to particularly interesting inspirations. Furthermore, these domains are outside of the primary product design domain and are common sources of inspiration for designers. The LLM then provides single noun-phrases as results (e.g., `protectiveness` → `tortoise, armadillo, armor`).

The user may select a recommended inspiration and continue branching out to explore further inspirations. For example, selecting tortoise and rerunning the analogy inspiration chain could result in new analogies such as `tortoise` → `tank, backpack, treasure chest`. The user may also freely change their concept (e.g., changing *protectiveness* to *freedom*, resulting in a new set of analogical inspirations). In our current implementation, users cannot return to previous inspirations or explore multiple inspiration branches in parallel. When a user selects an inspiration, it serves as a base to generate another set of inspirations. Furthermore, the generated inspirations remain independent of what the user sketches on the canvas. These features suggest potential areas for future work.

After selecting an analogical design inspiration, the user may create product designs by sketching on a canvas. Our conversations with the professional design team revealed that they often start a design with a single silhouette line (Design Goal 1). In Inkspire, the user may start generating images with as little as a single stroke (Figure **6.2**b). Using ControlNet [**4**] to guide Stable Diffusion [**10**], we generate a product design guided by the initial stroke. The user may continue adding additional strokes. Each time a stroke is drawn, we generate a new design, making the creation process iterative and implicitly encouraging users to focus on sketching instead of engineering text prompts (i.e., the current paradigm of working with T2I models).

ControlNet does not support per-stroke interaction out-of-the-box as it struggles with incomplete sketches, which is especially problematic during the initial stages of user sketching. Thus, we adapted the ControlNet model with a dynamic guidance scale to enable our desired per-stroke interaction (Equation **6.1**). The guidance scale is a parameter for controlling how closely the model adheres to user input. We initialize with a low guidance scale to handle incomplete sketches. We progressively increase the guidance scale as the designer adds more ink to the sketch, pushing the generations to become more sensitive to the user's sketch over time.

$$G(n) = 7 - 4 \cdot 0.5^{\frac{n}{3}} \tag{6.1}$$

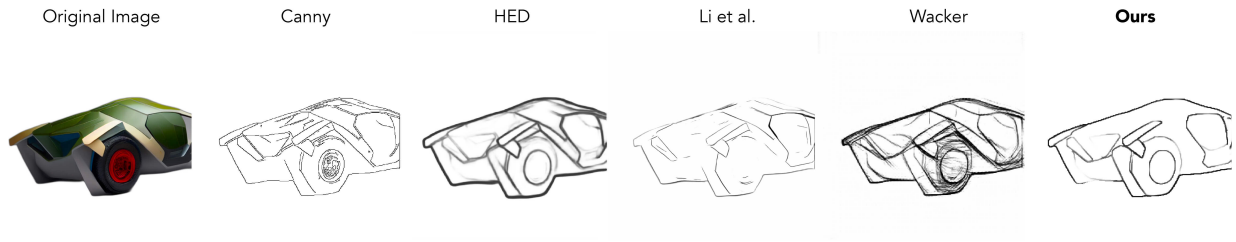


Figure 6.3. Comparison of our Design2Sketch method with potential alternative methods, such as edge detection, manga line extraction models, and models trained explicitly on pairs of sketches and images.

where $G(n)$ is the guidance scale at n number of strokes by the user. The guidance scale starts at 3 when n=0 and approaches a maximum of 7 as the sketch becomes more complete (n≈10). The decay term $0.5^{\frac{n}{3}}$ creates a sharp logarithmic growth in guidance scale that converges gradually at the maximum value.

By gradually increasing the guidance scale, Inkspire allows the user to more precisely guide the generations as their sketches become more complete and well-defined. We maintain the same initial seed (seed specifies the random noise used to initialize image generations) between generations to maintain *consistency* between iterations and for *near-real-time* generations. The user may click on the "remix" button to change to a different seed and generate more diverse designs that break from the current thread that the designer is exploring. Finally, we remove unnecessary backgrounds from the generated design using a foreground extraction method [252] (Figure 6.2c).

6.4.3 Design2Sketch

The *Design2Sketch* component helps users build on top of previously generated designs by converting them into *scaffoldings* by abstracting a design into a reduced-fidelity sketch-style with the aim of reducing design fixation on the high-fidelity image (Figure 6.4). The scaffolding appears as a transparent underlay beneath the user's canvas, functioning like tracing paper [253] that updates in real-time as they sketch. This enables the user to draw inspiration from aspects of the previously generated designs and also helps them overcome the challenge of starting with a blank canvas [196], especially during the early stages of sketching. The user can continue iterating through sketching, completing the sketch-to-design-to-sketch feedback loop (Design Goal 3).

While there are many methods for reducing high fidelity images into lower-resolution, we introduce a novel approach for converting designs to sketch scaffolds. We initially tested existing methods including Canny edge detection [254], HED soft edge extraction [255], a

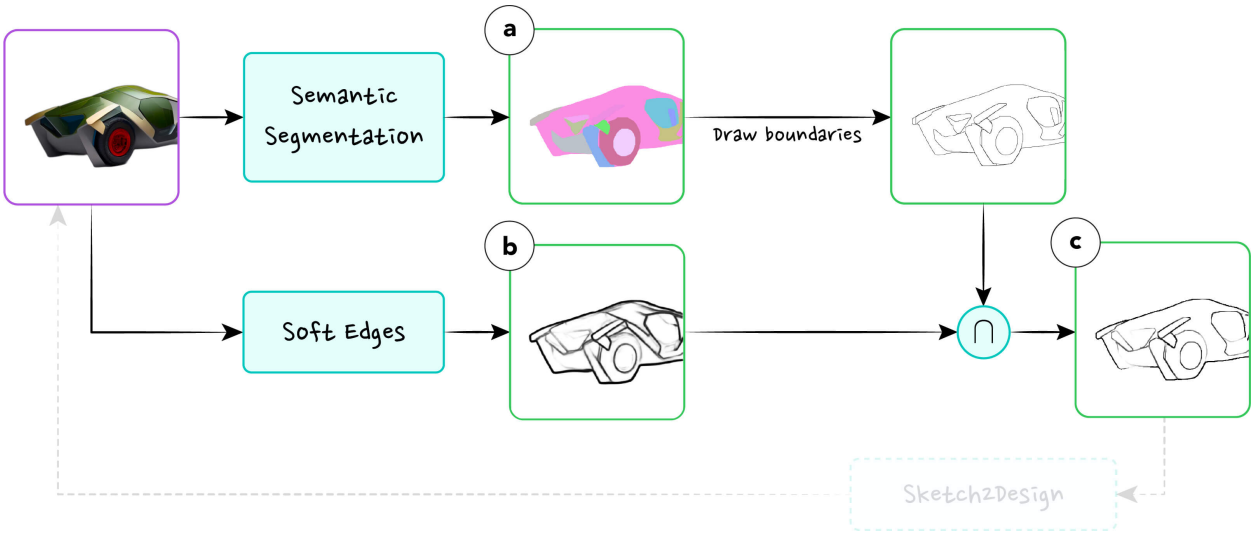


Figure 6.4. Design2Sketch pipeline, including (a) semantic segmentation, (b) soft edge extraction, and (c) computing an intersection.

state-of-the-art method for extracting main lines from manga illustrations [256], and a neural network method explicitly trained on pairs of sketches and images. (see Figure 6.3). We observed that edge extraction methods including Canny edge detection and HED soft edge extraction frequently produce unwanted lines caused by the texture of designs. Furthermore, we observed that manga line extraction methods, trained primarily on cartoon illustrations, can lead to a loss of key lines or produce broken lines. Finally, neural networks explicitly trained on pairs of sketches and images on the task of translating images to sketches can create artifacts of excessive sketch stylization, such as shading effects.

In our approach, we combine semantic segmentation and edge extraction. First, we perform semantic segmentation on the design [155] to create a segmentation map that color-codes a design into distinct semantic regions (Figure 6.4a). Given the segmentation map, we draw the boundaries between the different regions to create an image of semantic boundary lines. Second, we extract soft edges from the design [255] (Figure 6.4b). These soft edges include varying thickness and line weight, simulating a sketch-like look, though often with many redundant lines caused by texture. Finally, we take a pixel-wise intersection between the segmentation map boundary lines from the first step and the extracted soft edges from the second step as the final scaffolding (Figure 6.4c):

$$Scaffolding = Boundary(Seg(D)) \cap SoftEdge(D) \tag{6.2}$$

where D is the generated design. Through this approach, we are able to acheve the best of both worlds – creating a sketch scaffolding that achieves a natural sketch look while focusing

only on the design's key structural lines, filtered through the boundary lines from semantic segmentation step.

6.5 User Study

We conducted a within-subjects study to understand how Inkspire could address designers' pain points in working with T2I models, its potential to be integrated into design workflows, and identify areas for improvement. We compared Inkspire against a baseline condition using a typical ControlNet [4] setup consisting of a prompt box and a sketching canvas, adopting a similar interface layout as Inkspire (see Figure **B.2**).

6.5.1 Participants

We invited twelve participants (P1-P12, 10 male and 2 female) to participate in a one-hour user study. Among the participants, six are professional designers who perform product design activities daily or weekly (self-rated confidence in product design μ=6.17, σ=0.75; self-rated confidence in drawing μ=6.00, σ=1.27; 7-point Likert scale) and six are novices who have moderate drawing experience (self-rated confidence in drawing μ=4.17, σ=1.33; 7-point Likert scale) but do not actively engage in product design. The participants were recruited through known contacts and Upwork [**145**], a platform for hiring freelancers. They were not exposed to the Inkspire system or concept prior to the user study. Participants accessed Inkspire and the baseline tool through a web browser.

6.5.2 Measures

For both conditions, we asked participants to complete questionnaires to capture their perspectives on using both Inkspire and ControlNet. We assess creativity using the Creativity Support Index [**1**], measuring exploration, inspiration, engagement, expressiveness, tool transparency, and effort/reward tradeoff. We assess designers sense of the human-AI collaboration using questions from [**2**] measuring controllability, communication, harmony, partnership, attribution, and ownership. We also asked participants to rate their experiences of sketching by referencing the sketching principles from Bill Buxton's Sketching User Experiences [**6**], measuring how quick and timely, inexpensive and disposable, and loose and abstract sketching with each tool felt. We asked designers to rate the quality of the final design created with Inkspire and ControlNet and their overall experience satisfaction using each tool. All questionnaire questions were rated on a 7-point Likert scale (7=highly agree, 1-highly disagree). We compared the questionnaire measures using parametric paired t-tests.

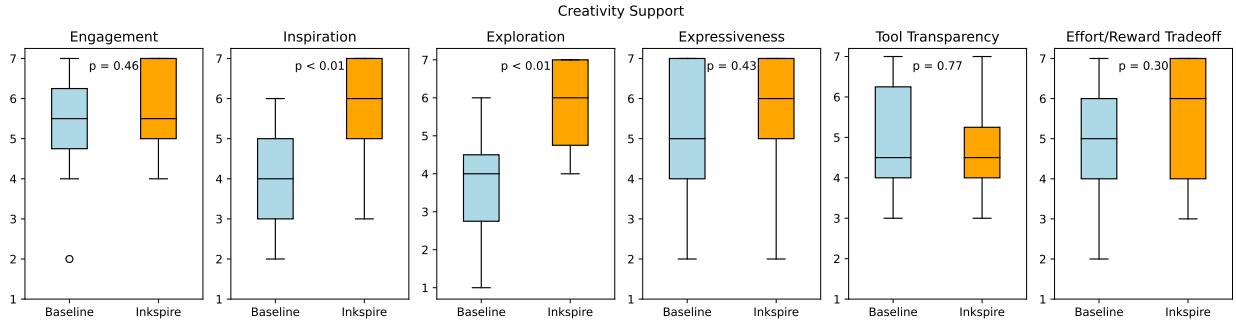


Figure 6.5. Results on creativity measured with the Creativity Support Index (CSI) [1] (7-point Likert scale, higher is better).

In addition, we log user interaction data, such as when participants draw a new sketch stroke, edit a prompt, and generate a new design. (Figures **B.1**).

6.5.3 Procedure

Introduction (5 minutes).
Participants provided informed consent and were given an overview of the study procedures. In addition, we briefly explain how components of the system, such as the analogy generation and sketching and scaffolding interaction, works.

Design Tasks (45 minutes).
Participants completed a design task with Inkspire (Figure **6.1**) and another design task with the baseline ControlNet tool (Figure **B.2**). The two design tasks are "design a lamp with the theme of serenity" and "design a chair with the theme of fluidity." We counterbalance both the order of the tools and the order of the design tasks. After each condition, the participants completed the questionnaires.

Post-Study (10 minutes).
Participants gave feedback during a short interview as well as through a free response questionnaire on the individual subcomponents of Inkspire, their overall experience of using Inkspire, whether they could see Inkspire being integrated into their design workflow, and areas for improving the tool. We reviewed these qualitative data to support the quantitative results.

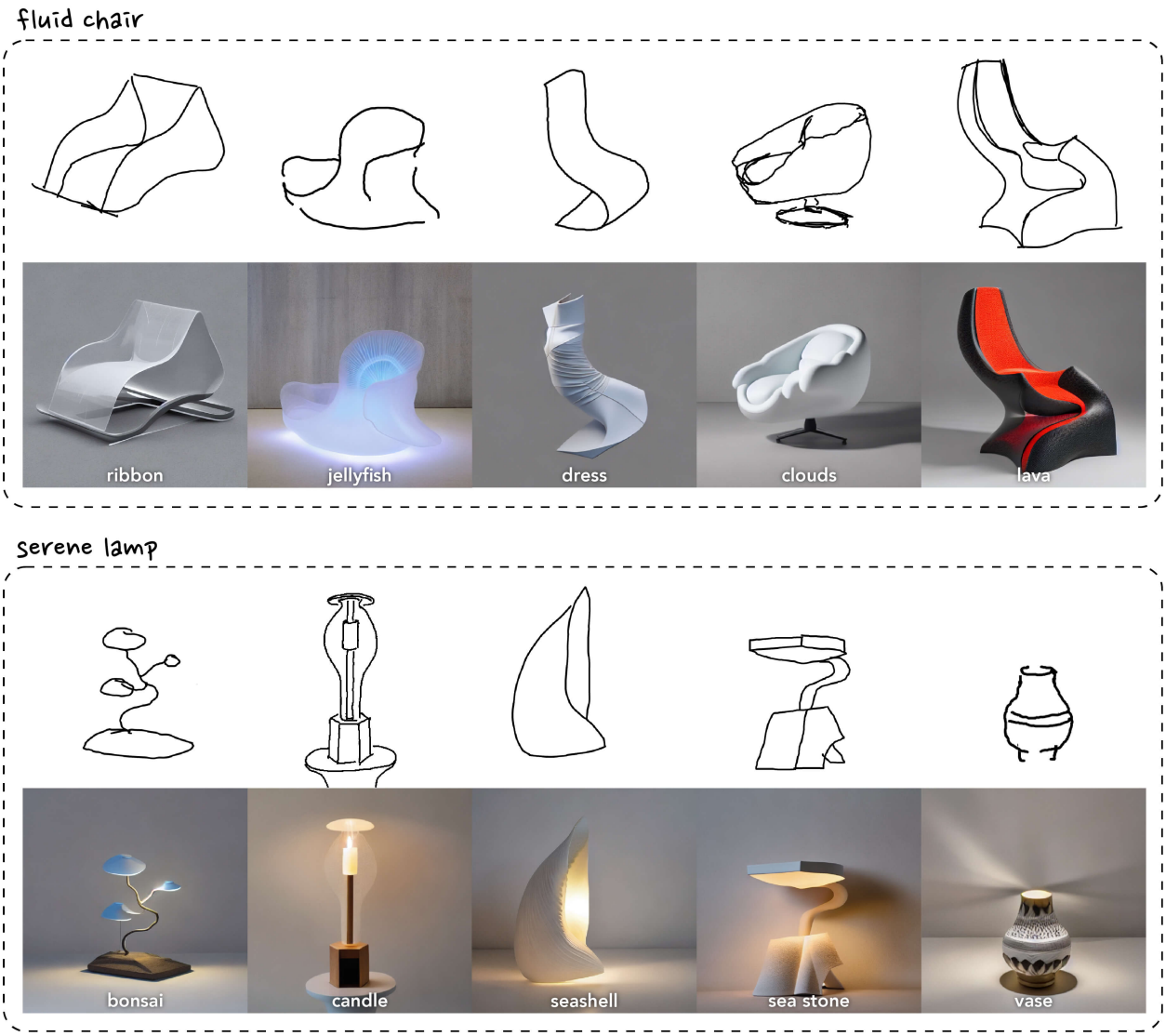


Figure 6.6. Example designs created by participants using Inkspire for the design tasks of designing a fluid chair and a serene lamp. The final participant-generated sketch is shown on the top, and the generated T2I image is shown on the bottom, along with the selected analogy word chosen by the participant.

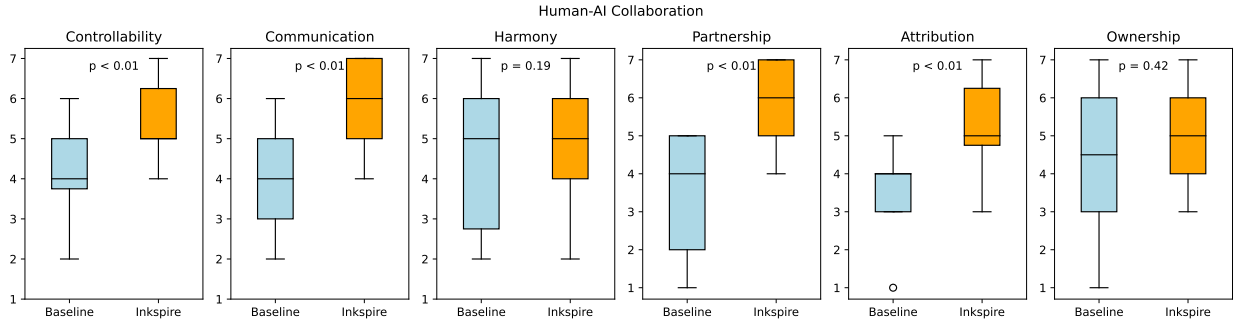


Figure 6.7. Results on human-AI collaboration measured with Human-Machine Collaboration Questions from [2] (7-point Likert scale, higher is better).

6.6 Results

6.6.1 Creativity

Participants felt that Inkspire improved support for creativity across some attributes of the Creativity Support Index (CSI), shown in Figure 6.5. Notably, participants reported significantly higher exploration with Inkspire (μ=5.83, σ=1.27) as compared to the baseline (μ=3.83, σ=1.64), ($t(11)$=3.94, p¡0.01, r=0.77, d_s=1.13). Participants also reported significantly higher inspiration (μ=5.92, σ=1.24) as compared to the baseline (μ=4.00, σ=1.41), ($t(11)$=3.44, p¡0.01, r=0.72, d_s=0.99).

From our interviews, participants noted that the [analogy] inspirations feature is "*helpful while doing design ideations*" (P12) and "*a good tool to brainstorm in the early stage of design*"(P5). Participants also found Inkspire to effectively support the exploration of multiple ideas, such as a "*variety of forms, styles, patterns, and proportions*"(P5). P11 explained that they could explore different design directions by manipulating high-level concepts ("*I could just make a basic shape, and change a few keywords and the entire look and feel would change and present me with some great concepts*"). In contrast, we observed that in the baseline condition, many participants focused on sketching extensively on a single idea, a potential sign of the "sunk-cost effect" [257] (the more time spent in a given direction, the harder it is to move to a different one).

We also note that the other attributes of the CSI, Engagement, Expressiveness, Tool Transparency, and Effort/Reward Tradeoff, were not significantly different between Inkspire and the ControlNet baseline. While Inkspire did not improve these aspects of creativity over using ControlNet it did not appear to degrade them either. Overall, we find that Inkspire improved exploration and inspiration, supporting our original design goals and working toward reducing design fixation.

6.6.2 Human-AI Collaboration

Designer self-ratings of Human-AI Collaboration.

Participants felt that Inkspire improved the experience of collaborating with the AI, across dimensions of Human-Machine Collaboration questions from [2] rated on 7-point Likert scales, shown in (Figure 6.7). Participants reported significantly higher controllability when using Inkspire (μ=5.58, σ=1.00) as compared to the baseline (μ=4.17, σ=1.19), ($t(11)$=3.56, p¡0.01, r=0.73, d_s=1.03). Participants felt that they had significantly higher communication with Inkspire (μ=5.75, σ=1.14) as compared to the baseline (μ=3.92, σ=1.38), ($t(11)$=4.52, p¡0.01, r=0.81, d_s=1.31). In addition, participants rated having a significantly higher degree of partnership with the AI when using Inkspire (μ=5.83, σ=1.03) as compared to the ControlNet baseline (μ=3.42, σ=1.56), ($t(11)$=4.57, p¡0.01, r=0.81, d_s=1.32). Lastly, participants rated having significantly more of their own attribution in the designs when using Inkspire (μ=5.25, σ=1.29) as compared to the baseline (μ=3.67, σ=1.07),($t(11)$=3.51, p¡0.01, r=0.73, d_s=1.01). Participants did not report significant differences in their feelings of harmony with the AI or ownership over the designs between Inkspire and baseline ControlNet.

From our interviews and open-response questions, many participants commented that they found the scaffolding helpful in being able to *understand the current state of the AI* and plan subsequent sketches ("[The scaffolding] *was useful because it let me know where the current iteration was so I could tell where I'd like to move it next*" (P11)). With scaffolding, participants felt that they could *steer* the direction of the design by building on previous generations ("[scaffolding] *helped with building upon the previous AI-generated design and gave me a direction for what to adjust*" (P6), "*the AI-generated drawing overlay help[ed] me to draw my next line*"(P3), "*having the [scaffolding] as reference helped me [to] combine and remix [old designs] in my [new] sketches* "(P6)). These observations and self-reported results suggest that Inkspire improves human-AI collaboration by increasing controllability, communication, and partnership.

Prompting Behavior.

As expected, we observed that participants generally did less prompt engineering when using Inkspire (μ=8.50, σ=6.31, number of prompts) than when using the baseline (μ=11.9, σ=10.6, number of prompts). When using the baseline ControlNet system, participants often relied on manipulating the prompt to change their design whereas they used more diverse analogical inspirations as prompts when using Inkspire. Analyzing the semantic similarity between user prompts, we observed a much lower semantic similarity in the Inkspire condition (μ=0.51, σ=0.08) compared to the baseline (μ=0.76, σ=0.13), measured with BERTScore [258].

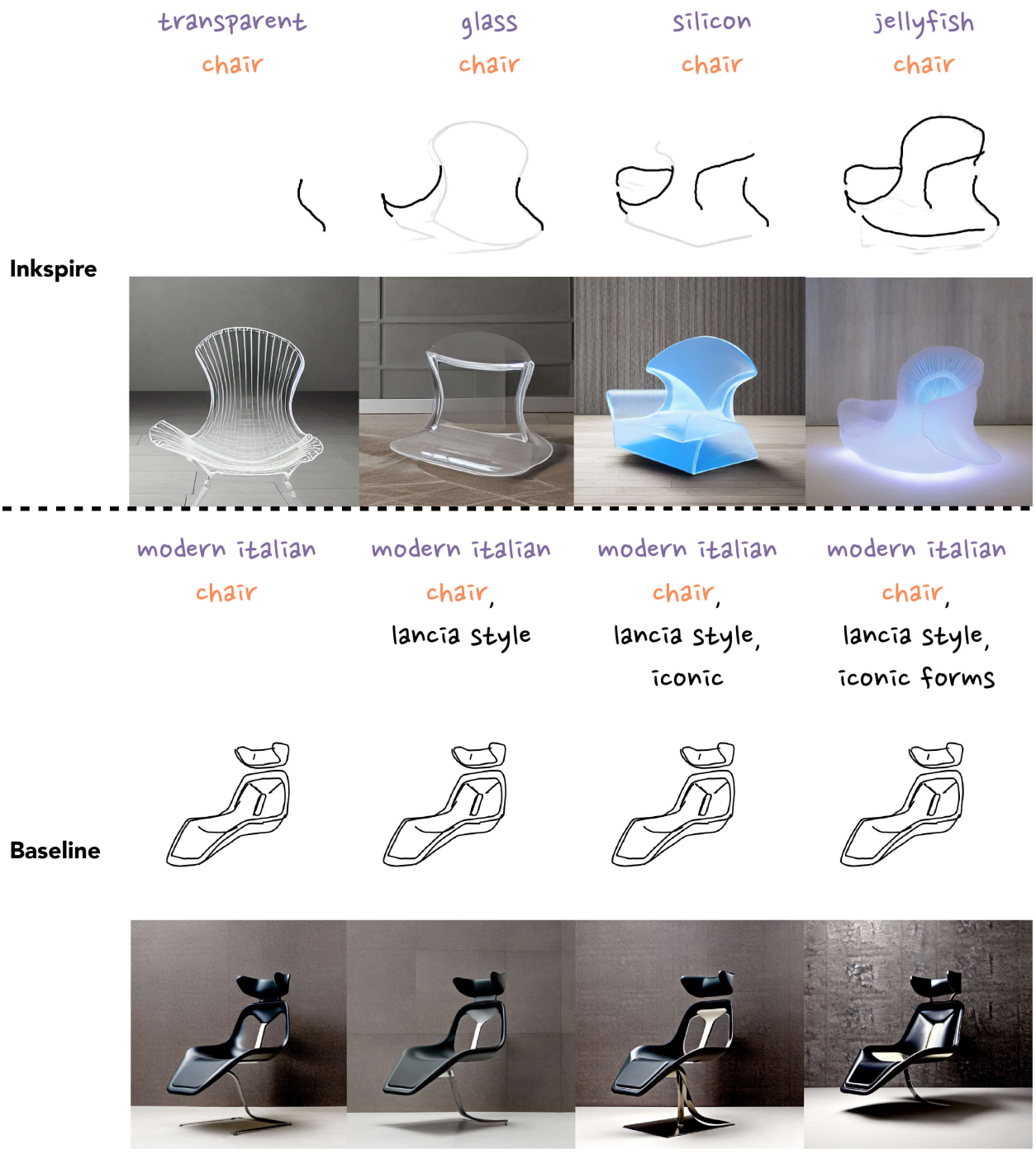


Figure 6.8. Example user iterations with Inkspire (top section) vs. the baseline condition (bottom section). For each section, the top row shows the prompts, the middle row shows the sketches and scaffolds, and the bottom row shows the generated designs. Using Inkspire, users create diverse designs via analogies. In addition, users start with a single sketch line and continuously build on their sketch, with the guidance of scaffolding. In contrast, using the baseline, users typically draw a full sketch and update their prompt with incremental modifications. This may lead to a higher degree of fixation and a smaller design space explored.

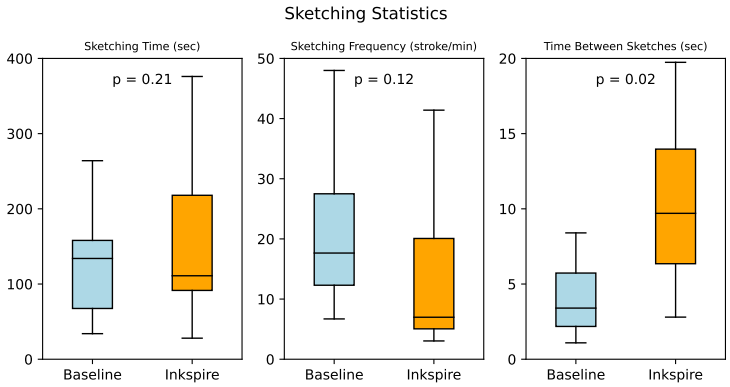


Figure 6.9. Results on sketching statistics, including total sketching time, sketching frequency, and time between sketches.

Looking at the logs of prompts designers used, we observed that participants using ControlNet often stuck to their original prompt, making small edits to make the prompt more and more detailed (see Figure **6.8**). For example, P3 started with the prompt `serenity lamp` (i.e., explicitly prompting the AI with the abstract design task), and then expanded it with additions like `serenity lamp with clear glass and black base`, `serenity lamp with clear glass and black pedestal`, and so on. This result echoes the prompt fixation results of [**204**] and adds further evidence to the challenges designer have in knowing how to prompt [**186**]. Using Inkspire, participants made fewer manual prompt edits and frequently utilized the recommended analogical inspirations to *guide* their prompting direction. For example, the same participant above (P3) used various analogical inspirations for creating a *fluid* chair such as `silk`, `river`, and `waterfall`.

Sketching Behavior.

We observed that participants drew fewer total sketch strokes using Inkspire (μ=17.3, σ=7.40, number of strokes) as compared to using the baseline (μ=59.8, σ=40.5, number of strokes). When using Inkspire, participants also had a lower sketching frequency (μ=12.5, σ=12.5, strokes/min) as compared to the baseline (μ=20.1, σ=12.6, strokes/min), though this was not significant ($t(11)$=-1.67, p=0.12, r=0.45, d_s=0.48). Participants also spent more time between strokes when using Inkspire (μ=13.8s, σ=12.1s) as compared to the baseline (μ=4.44s, σ=2.94s), ($t(11)$=2.66, p=0.02, r=0.63, d_s=0.77).

Despite participants sketching less and taking longer between adding more ink to their drawing, participants rated that sketch strokes felt significantly more inexpensive when using Inkspire (μ=6.08, σ=0.90, 7-point Likert scale, higher is better, based on Buxton's sketching principles [**6**]) as compared to using the baseline (μ=3.75, σ=1.60, 7-point Likert scale, higher is better), ($t(11)$=5.01, p¡0.01, r=0.83, d_s=1.45). Participants also rated

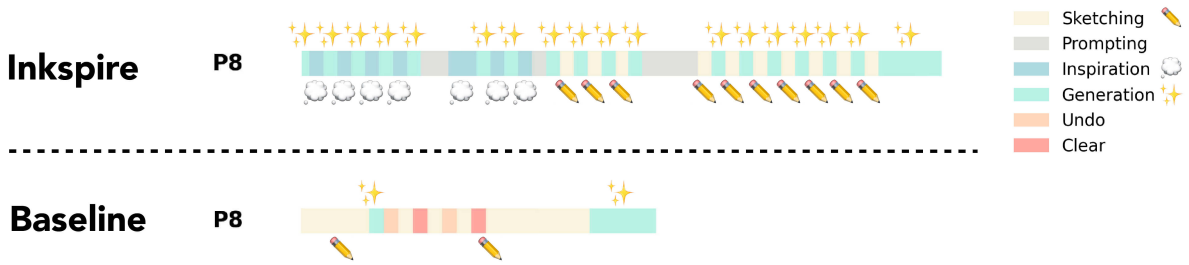


Figure 6.10. Example user interaction log. We observe that, using Inkspire, the user started by ideating several analogical inspirations (thought bubble). Next, they performed sketching in a highly iterative manner (pencil+sparkles). Overall, the user generated a significant amount of new designs (sparkles). In contrast, using the baseline condition, the user sketched in large stretches with infrequent new design generations. We see that this pattern generally holds true across participants (please see Figure **B.1** for data on all participants).

that the sketching was more abstract when using Inkspire (μ=5.75, σ=1.14, 7-point Likert scale, higher is better, based on Buxton's sketching principles [6]) as compared to the baseline (μ=3.83, σ=1.47, 7-point Likert scale, higher is better), ($t(11)$=4.24, p¡0.01, r=0.79, d_s=1.23). Overall, Inkspire appears to have provided designers with an interface for sketching ideas collaboratively with AI that improves over the more stilted sketching and image generation experience of other text-to-image systems.

Analogy Inspirations.

We observed that participants generally explored multiple analogical inspirations (μ=4.58, σ=2.43) distributed across the three categories (Nature μ=2.08, σ=1.56; Architecture μ=1.58, σ=1.3; Fashion μ=0.917, σ=0.716) (Table **B.1**). Architecture was the most common final choice (n=6 instances across participants), followed by Nature (n=4) and Fashion (n=2), though participants explored Nature-based inspirations more frequently on average. We also observed that participants frequently switched between analogy categories. For instance, P12 quickly switched between all three categories. Among the switches, we observed that Nature→Architecture (n=6) and Nature→Fashion (n=5) were common transitions. This may suggest that Nature can serve as a common "bridge" category. Many participants' final category deviated from their initial category after exploration. For instance, P2 started with Nature and selected Architecture after exploring 14 different inspirations across all categories. Overall, participants drew inspiration from all categories, with Nature playing a central role, though each category proved to be a valuable source for inspiration.

Overall Usage Patterns.

By analyzing our logged interaction data, we observed common usage patterns for Inkspire vs. the baseline ControlNet workflow (please see Figure **6.10** for an illustrative example and Figure **B.1** for the full data). First, participants using Inkspire often start the initial ideation phase by experimenting with many analogical inspirations to explore the design space (shown as many thought bubbles during the beginning, Figure **6.10**). This early divergent exploration behavior may suggest that designers were able to consider a wide range of inspirational ideas at the start of their process. Subsequently, participants using Inkspire engaged in a highly iterative manner of sketching and interacted with scaffolding, often drawing one or a small number of strokes, seeing a generation and scaffolding, then pausing a bit to consider where to move next (shown as many pencils interleaved with sparkles, Figure **6.10**). In contrast, when participants used the baseline system, they often did a complete full sketch (shown as large stretches of a single pencil, Figure **6.10**) before ever hitting the generate button. Designers made more "Undo" actions, removing strokes that they had made (as shown in orange blocks). Overall, participants generally created more new designs with Inkspire than with the baseline condition (shown as significantly more sparkles in Inkspire than the baseline, Figure **6.10**). Together, these patterns show how when participants used ControlNet, they focused more on crafting a whole sketch then handing it off to the generative AI rather than working collaboratively back and forth with the AI.

6.6.3 Final Design Quality

Participants rated their final creations as having higher design quality when using Inkspire (μ=5.92, σ=1.08) as compared to the baseline (μ=4.58, σ=1.93) on a 7-point Likert scale, shown in Figure **6.11** (left). Though this difference in design quality ratings is not statistically significant, the ratings showed less spread than when using the baseline ($t(11)$=2.11, p=0.06, r=0.54, d_s=0.61).

In Figure **6.6**, we show several example creations from participants. While the quality of output in general appears relatively high, we do observe that the final results produced with Inkspire do appear to be more diverse than the results produced with ControlNet. Looking at the trend of designs created during a user session, we see that the thread of designs created using ControlNet often do not change tracks, whereas those done with Inkspire show more diversity of conceptual exploration. An example showing chair designs from two different users illustrates this in Figure **6.8**. The participant using Inkspire (top) shows a wide exploration of concepts for a `transparent` chair including `glass`, `silicon`, and `jellyfish`. Alternatively, a participant using ControlNet creates a `modern Italian` chair with a high quality initial sketch and iterations on the prompt from `lancia style` to `lancia`

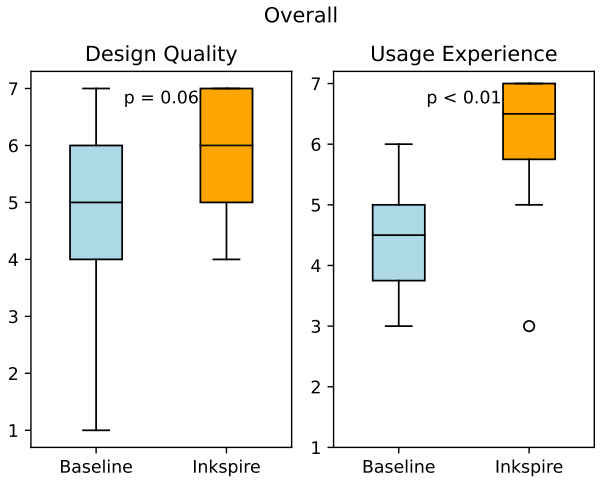


Figure 6.11. Results on design quality (left) and usage experience (right) (7-point Likert scale, higher is better).

`style, iconic` to `lancia style, iconic forms`. While both final result are generally of high quality, the final output from the participant using Inkspire is derived from a more exploratory process over design analogies over the more fixated design process from the designer working with ControlNet.

6.6.4 Designer Ratings of Usage Experience Satisfaction

Participants rated significantly higher satisfaction with their usage experience when using Inkspire (μ=6.08, σ=1.24) as compared to the baseline (μ=4.33, σ=0.98) when rated on a 7-point Likert scale ($t(11)$=3.54, p¡0.01, r=0.73, d_s=1.02), shown in Figure **6.11** (right). Based on the interaction logs, Inkspire may improve participants' experiences of sketching with AI by supporting low cost experimentation and helping participants draw abstract sketches that focus on the big picture rather than being bogged down by small details. In the baseline, we observed that participants drew a number of large strokes in succession *before* generating anything, with the majority being "filler" lines (Figure **B.1**). This generally led to designers spending more time between trial-and-error attempts to achieve their intended result, possibly reducing their satisfaction when using ControlNet.

6.7 Discussion

We introduced Inkspire, a prototype system to explore new ways for designers to leverage generative AI while avoiding design fixation and embedded into their existing practice of sketching new concepts. We focused on a challenge identified in design research literature

that exploring a wider range of ideas can lead to improved design outcomes, though designers often find it difficult to naturally engage in diverse design exploration [37, 259]. Inkspire helped designers explore a more diverse design space by recommending them with diverse analogical anchors based on their initial design concept (e.g., turtle, shield, or bunker for embodying the concept of "protective"). These analogical concepts were used as input along with the sketch from the designer to generate new images. The designer could then continue to iterate on the designs collaboratively with the AI by leveraging stroke-by-stroke generation and building on low-resolution sketch scaffolds of high fidelity renders. Through a user study, we found that participants using Inkspire explored a wider design space and qualitatively changed their ideation process to be more conceptually iterative and collaborative with the AI. In contrast, participants in a control condition, in which they used a state-of-the-art ControlNet, primarily focused their efforts on crafting full sketches and making small changes to prompt inputs before having the AI render an image.

A core challenge we grappled with in this work was helping designers to leverage generative AI while avoiding fixation on the outputs of the AI. We believed this to be especially likely in the context of creative design and sketching because of several factors derived from our formative conversations with professional designers and prior research: text prompting leading to what designers called "unnatural" interaction, text-to-image models struggling with generating inspiring designs from abstract concepts, both often due to designer's limited abilities to prompt well [186, 204, 212, 223] and the AI generating outputs that often look overly "finished", a well-known cause of design fixation [204, 224, 225].

Our results suggest a fundamental issue with the structure of prompt- or sketch-focused generative image interfaces, which is that current controls through prompts or sketches can engender an affordance of needing to feel relatively complete in order to convey the designer's intention. The results from when our participants used ControlNet are similar to results from Wadinambiarachchi et al. [204] where designers focus most of their effort on minor iteration of prompts. Needing to provide a rich enough prompt or a full enough sketch may explain why our participants felt less control over the ideation and less communication and partnership with the AI model.

In contrast, with Inkspire the design problem prompt and analogical keyword concepts provide a scaffolding in the design space allowing a single stroke to have expressive meaning while also generating an output that is relevant to the user's goals. Relatedly, our work also explores ways for designers to have a shared mental model of what they are trying to accomplish versus what the AI is trying to do. For example, an interesting point we noticed from our participants was that they found the rough sketches inferred from the AI generation to be useful as control points that showed what the AI was doing and how the user could

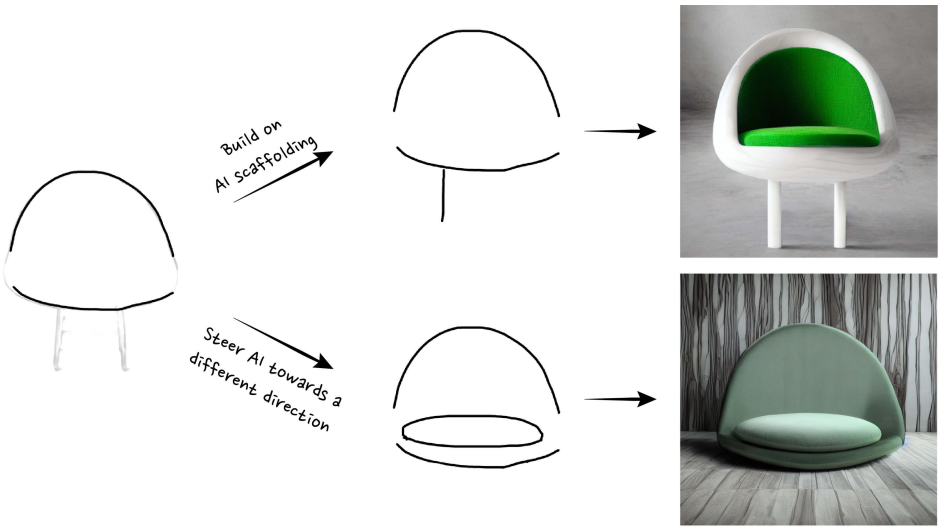


Figure 6.12. Given a scaffold, the user may choose to build on it (by tracing it) or steer the AI towards a different direction.

emphasize or change that, for example by either drawing over or changing a stroke on the roughed sketch, respectively (please see Figure **6.12**). In this way the rough sketch serves not only as a scaffold for the designer's iterative sketching but as a communication tool and shared mental model with a generative AI partner. The continual updating of both sketch and analogical concept ultimately leads to a more iterative and collaborative process where the designer works in turns with the AI. Other generative AI tools could potentially build on this pattern of rapid turning taking to help promote more co-creativity for their creative task.

Overall, the interactions we designed into Inkspire appear to have worked in helping our participants avoid design fixation. As commented on by P6 "*the AI was helpful in providing alternative designs I did not consider previously, especially by adopting a new semantic understanding of the sketch that differed from my initial intention.*" However, even though working with Inkspire pushed the designers into a space they had not originally considered and even resulted in them sketching less in total, they found that we also succeeded in overcoming issues where designers feel less agency when working with generative AI. That the participants felt they had more agency and more ownership of the final design outputs suggests the analogies, stroke-based interaction pattern, and scaffolds led to more fluid interaction through a design space. Again quoting from P6, Inkspire had "*an interesting workflow that I think helps make the tool disappear more in comparison to a prompt-based drawing tool. It allowed me to focus more on sketching than prompt engineering.*"

While our overall results show that Inkspire achieved our design goals, there are some potential issues with Inkspire that could arise and could be considered in future co-creative systems. First, although we see more evidence of designers moving to more distinct ideas throughout a session, they still work along a single thread. Prior research has shown that designers often prototype ideas in parallel and that such parallel exploration can lead to better design outcomes [37]. One potential way to enable parallel explorations in Inkspire is to generate numerous analogies for a concept and generate an image for each during each sketch step. Second, as Inkspire was primarily designed for early explorations, it may not serve a designer as well during later stages of the design when they have a clear idea in mind that they are trying to render. In this case, a designer may prefer using a tool such as ControlNet where they can focus on the details they want before handing it to an AI. Lee et al. similarly proposed an adaptive multimodal T2I system that initially supports early ideation with a prompt-guided (e.g., protective chair) and sketch-supported (e.g., simple sketch) system, which gradually evolves into sketch-guided (e.g., detailed sketch) and prompt-supported (e.g., "more curved back") system to help refine the concept in later stages of the creative process [260]. This suggests potential opportunities for generative systems to provide controls for how much turn taking the AI should aim to have with the designer. Third, Inkspire and other generative AI approaches may help individual designers break their fixation on specific ideas, however, if common generative models are used, it may promote collective design fixation, where everyone's images begin to look the same as they have a common source [261]. This being said, Inkspire may work to counteract such collective fixation by promoting increased designer engagement by focusing more on suggestions of designs though lower-resolution sketch underlays rather than pixel-level in-painting. Future systems could explore other techniques for avoiding design fixation, such as showing partial photographs [225] instead of sketch underlays. Other ideas for reducing fidelity such as blurring or filtering could also promote designers to fill in details on their own, potentially leading to designs less similar to common output from AI models.

6.7.1 Future Improvements to Inkspire

There are several avenues for future work on improving Inkspire collected from participants' comments. First, we could improve the sketching canvas with more advanced sketching features such as line weight control and shading tools, which could allow users to sketch with greater detail. Second, participants suggested adding more fine-grained region-based control to the Sketch2Design pipeline. For example, a user could specify different material types for different regions of the sketch. Third, we could extend the analogical panel to include the capability to go back to a previous inspiration and perform multiple branches

of inspiration explorations in parallel. This could help users compare vastly different design directions. Fourth, we currently adopt a two-step prompting mechanism for generating analogical inspirations: defining the design principles for the target domain, then generating visually-concrete object inspirations for the target domain using nature, architecture, and fashion as source domains. Future work could explore more complex prompting structures, such as traversing hierarchical tree structures [248], that might lead to better analogical inspirations. Fifth, Inkspire contains several features that could all affect a user's behavior, such as automatic per-stroke generation and sketch scaffolding. For future work, we could conduct additional ablation studies. For example, adding dynamic guidance scale to enable automatic per-stroke generation for the baseline condition or hiding sketch scaffolding for Inkspire. Sixth, we adopted a within-subjects design with a relatively small sample size (n=12), which could be improved through larger-scale studies. Seventh, participants suggested the possibility of adding additional guidance for the AI, such as mechanical and material constraints, though such a feature may require more research into how to better connect analogically created T2I design concepts with their feasibility to be manufactured. Finally, in this paper, we introduce a sketch-generation-scaffold interaction with generated analogies for abstract concepts through the application of product sketching. We think the sketching strategy could generalize to other forms of drawing, though the analogies might be better suited for product design. For example, very recent work has explored a similar strategy but using an image underlay instead of sketch scaffolding [262]. Future work could explore extending the system to additional design domains and incorporating background elements into designs for mockup or decoration.

6.8 Conclusion

This research identifies the challenges designers face when using Text-to-Image (T2I) models in design and proposes a set of design guidelines for creating new interfaces for interacting with T2I systems. The research prototype, Inkspire, supports designers in prototyping product design concepts with analogical inspirations and through a complete sketch-to-design-to-sketch feedback loop. Feedback from designers using Inkspire suggests that it could encourage more inspiration and exploration of novel design ideas while allowing designers to focus on iterative sketching collaboratively with the T2I model rather than prompt engineering. These analogy and sketch scaffolding features allowed designers to generate initial ideas for a design concept from a single abstract concept and single stroke of the pen to expand them into multiple alternatives. The iterative interaction and turn-taking between the designer and the AI helped the designer guide the models toward novel design intentions with the

potential to overcome design fixation. We are interested in extending Inkspire with more capabilities and hope that this research can help inform future research on sketch-driven tools for co-creating with Generative AI models.

Reading Inkspire through the Human-AI-Interface Triangle: structural conditioning via ControlNet gives a handle that responds to spatial input, and the sketch scaffolds make that interpretation visible step by step (controllability), pen strokes support the moves sketching requires (affordance), and the sketch is the designer's native medium for iteration, so each stroke produces the structural change it implies and the model fills in the rest (intuitiveness). Iteration converges on an intended design. The next chapter takes the inverse approach: instead of refining toward a target, designers diverge through variation. GENErator treats style adapters as genes that can be crossed, mutated, and evolved.

Chapter 7

GENErator: Evolution as an Interface for Mixing Style Models

This chapter presents GENErator, which addresses the *remix* task. Style exploration is hard to specify in words. A designer may know they want something between Art Deco and Brutalist without being able to name the result. And tens of thousands of style adapters now make the space too large to browse by keyword. To address these challenges, we developed GENErator, a proof-of-concept tool that treats each visual style as a "gene" in a navigable semantic style space. Designers cross, mutate, and evolve combinations through an evolution tree that keeps each design's lineage visible. Within the HAI Triangle, the design task is discovering styles a designer could not name in advance, the AI mechanism is mixing style models in a semantic space, and the interface representation is evolution.

This chapter was adapted from a paper conditionally accepted to UIST '26: Lin, D. C. E., & Martelaro, N. GENErator: Supporting Design Style Exploration with Generative AI through Genetic Mixing.

7.1 Introduction

Generative AI models can now be fine-tuned into thousands of style-specific adapters, each encoding a distinct visual aesthetic. Low-Rank Adaptation (LoRA) models [263], in particular, have become a popular way for creators to capture and share visual styles (from Art Deco to Brutalist to Vaporwave), with platforms like CivitAI hosting over 100,000 such models. Recent advances in multi-LoRA composition have shown that these adapters can be smoothly blended [264] and combined [265], creating a combinatorial design space of significant scale. Product designers have begun experimenting with these adapters for concept ideation, surface material exploration, and moodboard creation [266, 267], applying visual styles to objects like sneakers, handbags, furniture, and vehicles.

Despite these possibilities, designers face a vast and largely opaque design space. With tens of thousands of style adapters available, and the combinatorial possibilities of blending them, the space of achievable styles far exceeds what any designer can explore manually. Yet current interfaces offer little support for navigation. The typical workflow involves either browsing large lists of style adapters or searching for style adapters by keyword and selecting them by name. Designers then adjust numerical weight sliders, generate an image, and repeat. The style adapter search process is vocabulary-bound: designers can only find styles they already know how to name. It is also opaque: slider values bear little meaningful relationship to visual outcomes, and it is challenging to see how styles relate to one another. Without visibility into the structure of the style space, exploration defaults to trial-and-error, a pattern well-documented to cause design fixation [268]. In our pilot sessions with professional designers from a major automotive company, we observed this firsthand: designers reported that with existing tools and interfaces, they '*can't remember how [they] got there*' when a good combination emerged, and that without structured operations, '*you're not going to use 99% of the library.*' Recent empirical findings confirm that text-to-image tools can actually increase rather than decrease fixation through prompt anchoring and output attachment [269], and that text-based prompting restricts creative exploration, especially for novices [270].

Prior work has explored ways to address AI design fixation through methods such as keyword recombination [271], sketch-based interaction [272], cross-domain analogy [273], and graphical style manipulation [270]. However, these tools address fixation at the level of input modality or ideation process. There remains limited support for exploring the *style space itself*: discovering which styles are available, understanding how they relate, and combining them into new directions. Interactive evolutionary computation (IEC) offered an early vision for this kind of exploration [274], allowing users to guide aesthetic search by selecting preferred candidates rather than describing them. But random mutation causes user fatigue, as most variations bear little meaningful relationship to the selected parent [275, 276]. Meanwhile, LoRA composition has made multi-style blending technically feasible [264, 265, 277, 278], but the interfaces (CivitAI, Automatic1111) still present designers with sliders and text fields, leaving the style space a black box with little visibility into how styles relate or where a combination sits in the space.

To address these challenges, we developed GENErator, a proof-of-concept tool that treats each visual style as a "gene" in a navigable semantic style space. Our key insight is that by creating embeddings of each style across diverse object contexts and subtracting the object contribution, we can isolate a content-independent style direction, effectively giving every gene a semantic address. This facilitates genetic operations that are semantically

Table 7.1. Comparison of tools for AI-assisted style exploration.

Tool	Exploration strategy	What is varied	Semantic guidance	Multi-style composition	Lineage tracking	Style search
GENErator	**Genetic operations**	**Style models (LoRA)**	**Yes (style embeddings)**	**Yes (crossover, mutation, evolution)**	**Yes (evolution tree)**	**Semantic + reverse image**
Picbreeder [279]	Collaborative IEC	CPPN genotypes	No	Yes (branching)	Yes (phylogenetic tree)	Keyword
Artbreeder [280]	User-guided IEC	GAN latent vectors	No	Yes (weighted crossover)	Yes (genealogy view)	No
Hall & Yaman [281]	IEC + aesthetic scoring	CAN latent vectors	Partial (aesthetic fitness)	Yes (crossover)	No	No
DesignWeaver [282]	Dimension variation	Prompts	Partial (named dims)	Partial (multi-tag)	No	No
POET [283]	Auto-diversification	Model outputs	No (automatic)	No	No	No
Luminate [284]	Dimension variation	LLM outputs	Partial (interp. dims)	No	No	Keyword + filter
CreativeConnect [271]	Reference recombination	Text keywords	No	Partial (keyword merge)	No	No
StyleFactory [285]	Strength calibration	Single LoRA weight	No	No	No	No
CivitAI/Vizcom/A1111 [286–288]	Slider adjustment	LoRA weights	No	Yes (manual sliders)	No	Keyword

guided rather than random. Designers can discover styles by describing aesthetics, breeding crossover styles to produce hybrid offspring, mutating styles towards similar or maximally different regions of the space, and performing natural selection to automatically evolve a population toward a target aesthetic. An evolution tree interface makes the full lineage of every style combination transparent and traceable. Rather than adjusting opaque parameters, designers discover, combine, mutate, and evolve styles through operations informed by how styles relate to each other in the style embedding space. In a within-subjects study (N=12), we compared GENErator to a standard multi-style slider-based interface where designers select styles by name and adjust numerical weight sliders. Participants using GENErator explored 2.2 times more unique styles, reported significantly improved understanding of style relationships (d_s=1.66), and shifted from linear trial-and-error toward structured branching exploration.

This research thus contributes:

- A **content-independent style embedding technique** that isolates each style's semantic contribution by encoding it across diverse object contexts and subtracting the object signal, positioning all styles in a shared space to enable guided genetic operations.
- **GENErator**, a proof-of-concept tool that supports diverse style exploration through genetic operations, including crossover, guided mutation (complement and contrast), natural selection, and an evolution tree that visualizes the full compositional lineage of every design.
- A **within-subjects evaluation** (N=12) comparing GENErator to a standard multi-style slider-based interface, showing significantly broader exploration (2.2 times more unique styles), improved understanding of how styles can be combined, and supported a behavioral shift from blind parameter tweaking toward more structured, additive exploration.

7.2 Related Work

GENErator builds on prior research in design fixation in AI-assisted design, style representation and composition in generative AI, and interactive evolutionary computation. Table 7.1 compares GENErator against the most closely related tools for AI-assisted style exploration.

7.2.1 Design Fixation in AI-Assisted Design

Design fixation, the tendency to remain stuck exploring a limited set of ideas [268], has become a growing concern as generative AI tools enter design practice. Wadinambiarachchi et al. [269] found that text-to-image tools can increase fixation through anchoring to effective prompts and fixating on AI outputs rather than exploring alternatives. Doshi and Hauser [289] report that while AI improves individual creativity, it reduces collective diversity. Davis et al. [270] found that graphical interfaces outperform text prompting for creative exploration.

Several recent systems have proposed approaches for reducing fixation. Some change the input modality. For example, Inkspire [272] replaces text prompting with a sketch-to-design-to-sketch feedback loop and analogical inspiration. Others restructure the exploration process. Luminate [284] lets users vary LLM outputs along interpretable dimensions, CreativeConnect [271] supports visual reference recombination through keyword extraction, and DesignWeaver [282] scaffolds T2I product design with named dimensions extracted from generated designs. Concurrently, POET [283] addresses output homogeneity in text-to-image generation by automatically discovering and expanding dimensions of variation. Each of these tools addresses fixation at the level of input modality, ideation process, or output diversification. In this work, we aim to operate directly on the style space itself, making its structure visible and trackable, and allowing designers to compose, mutate, and evolve styles within it.

A complementary line of work has explored how making generative spaces visible and navigable supports creative exploration. Hartmann et al. [290] established that creating multiple alternatives in parallel supports design exploration. More recently, InterFaceGAN [291] identified interpretable semantic directions in GAN latent spaces, GANCollage [292] presented a mood board interface for composing GAN-generated elements, and Browsing the Latent Space [293] let users walk through a generative space to discover unexpected outputs. A growing body of systems have extended this approach to specific creative domains, including video editing [294], analogical design [273], art browsing [295], spatial ideation [296, 297], and iterative concept exploration [298–300].

These systems establish that visibility supports creative exploration, but most focus on navigation and selection within existing outputs rather than creating new points in the space. In our work, we apply genetic diversity operators directly to the style representation space. Where prior tools help designers express their intent more effectively or navigate existing outputs, GENErator lets designers create new points through crossover, move toward semantically relevant neighbors through mutation, and converge toward targets through natural selection.

7.2.2 Style Representation and Composition in Generative AI

Over the past year, the technical foundations for multi-style composition have advanced rapidly. B-LoRA [277] showed that style and content can be implicitly separated by training LoRA on specific transformer blocks, indicating that style is a manipulable parameter rather than a side effect of generation. K-LoRA [265] achieved training-free multi-LoRA fusion, demonstrating that multiple styles can be combined without additional optimization. DiffMorpher [264] achieved smooth image morphing by interpolating between LoRA parameters, showing that the LoRA parameter space supports smooth semantic transitions. CLoRA [278] introduced contrastive test-time attention masking to reduce semantic conflict when composing multiple LoRAs, Zhong et al. [301] proposed two training-free methods (LoRA Switch and LoRA Composite) for multi-LoRA composition with the ComposLoRA benchmark, and CMLoRA [302] used frequency-domain analysis to optimize LoRA integration order. In the text-embedding domain, EvoPrompt [303] applied genetic operators to discrete text prompts using LLMs, showing that evolutionary strategies can optimize prompts for downstream tasks, but for automatic prompt optimization rather than style exploration.

Many HCI systems focus on helping designers control individual styles. StyleFactory [285] supports style-strength ranking and alignment visualization for calibrating a single style's intensity. DesignPrompt [304] combines images, colors, and text into multimodal prompts for individual generations, and Generative.fashion [270] offers graphical style exploration using GANs. Misty [305] supports conceptual blending of UI design examples into prototypes. While some of these systems support compositional mechanics, few provide ways to navigate or search across a large combinatorial style space. In current practice, platforms like CivitAI [286], Vizcom [287], and Automatic1111 [288] present multi-LoRA composition through weight sliders and text fields, with no semantic search, relationship visibility, or lineage tracking. In our work, we build on LoRA composition and semantic embeddings to create a style exploration layer where genetic operations guide designers across the style space through semantically meaningful composition, rather than manual slider adjustment.

7.2.3 Interactive Evolutionary Computation

Interactive evolutionary computation (IEC) is a paradigm in which users guide an evolutionary process by selecting preferred candidates from a generated population, with the system producing new candidates through genetic operators such as crossover and mutation [275]. A persistent challenge is user fatigue: because mutations are random, most generated candidates bear little relationship to the user's aesthetic direction, requiring many evaluation rounds to make progress [275]. Dawkins [306] introduced the concept with biomorphs, simple shapes evolved through user selection. Sims [274] brought IEC to computer graphics at SIGGRAPH, establishing that selection is a natural interface for aesthetic preference: designers pick what they like rather than describing it. Picbreeder [279] extended this to collaborative online evolution where users could branch from each other's evolved images, and Artbreeder [280] brought IEC to millions of users through GAN-based image breeding. However, Artbreeder's mutations remain at the level of random perturbations without reference to named styles or aesthetic concepts, and it provides no lineage tracking across evolved images.

More recent systems have combined IEC with learned latent spaces. Bontrager et al. [307] placed GAN latent vectors under evolutionary control, showing that deep generative models provide compact representations for IEC. Schrum et al. [308] extended this to game level generation, StyleIE [309] introduced crossover in StyleGAN's style space, and Hall and Yaman [281] combined IEC with creative adversarial networks, proposing an "intelligent mutation" operator based on aesthetic scoring. This last system comes closest to addressing user fatigue but still lacks semantic guidance: mutations are drawn from a local neighborhood in latent space without reference to named concepts or user intent.

Despite two decades of proposed mitigations [276], the root cause largely remains that random mutation produces candidates that are mostly irrelevant to the designer's direction. In our work, we aim to address this by making mutation semantically guided: complement mutations select from the most similar styles in the semantic style space, contrast mutations from the most distant, and natural selection progressively evolves styles towards a user's natural language descriptions.

7.3 GENErator

We first illustrate how a designer would use GENErator through a lounge chair design example. We then describe the technical implementation, covering the latent style space, gene library and style search, genetic operations, natural selection, and the evolution tree. We detail technical validations of the system's core mechanisms in Appendix C.1.

7.3.1 System Walkthrough

Zaha is a furniture designer exploring styles for a new lounge chair concept. Teaser figure, Figure 7.2, and Figure 7.4 show example interfaces of this walkthrough.

Zaha first types "fluffy" into the *semantic search* bar. GENErator ranks all genes in the database by similarity to her query and surfaces `Plush` (67% similarity score), `Wooly` (55%), and `Cotton Candy` (42%) (Figure 7.2). She drags `Cotton Candy` onto the canvas, types "lounge chair" as the prompt, and the system generates a playful, cloud-like chair in soft pastel hues.

Zaha also has a photo of a jellyfish that captures a luminous quality she wants. She uploads it using *reverse en-GENE-ering*, and the system identifies which genes best match: 42% `Glow`, 28% `Bioluminescent`, 18% `Ethereal`. She drags `Glow` onto the canvas, over `Cotton Candy` to create a crossover. GENErator blends the two style genes, producing an offspring that merges the softness of `Cotton Candy` with the luminescence of `Glow`. The offspring appears as a child node in the evolution tree.

Wanting to push the hybrid further without losing its character, Zaha right-clicks the hybrid gene and selects *complement* mutation. GENErator randomly selects from the top 10% most semantically similar genes and picks `Opalescent`, creating a smoother glow. Curious about an unexpected direction, she tries *contrast* mutation, which randomly draws from the bottom 10% most dissimilar genes. GENErator adds `Geode`, producing a crystalline offspring.

Finally, Zaha wants to refine towards a specific aesthetic direction using *natural selection*. She types "luxurious" as a natural selection target. GENErator uses her hybrid gene as the initial seed and automatically evolves populations of styles toward this aesthetic over three generations, blending in genes like `Posh` and `Diamond` that score highest against the target (Figure 7.4). The evolution tree now shows her full exploration lineage: from initial search, through crossover and mutation, to automated evolution. She can click any node to backtrack and branch off in a new direction without losing her current path.

7.3.2 Latent Style Space

For genetic operations to be semantically meaningful, the system needs to know where each style lives in a shared space. The challenge is that styles are entangled with objects: encoding "a chair in Chrome" captures both the chrome material *and* the chair shape. Our approach isolates the style contribution by subtracting out the object. For each style gene s, we encode N=30 prompts with the style applied ("a chair in Chrome," "a sneaker in Chrome," etc.) and the same N prompts without ("a chair," "a sneaker," etc.), take the difference for each pair, and average: We then take the difference for each pair and average:

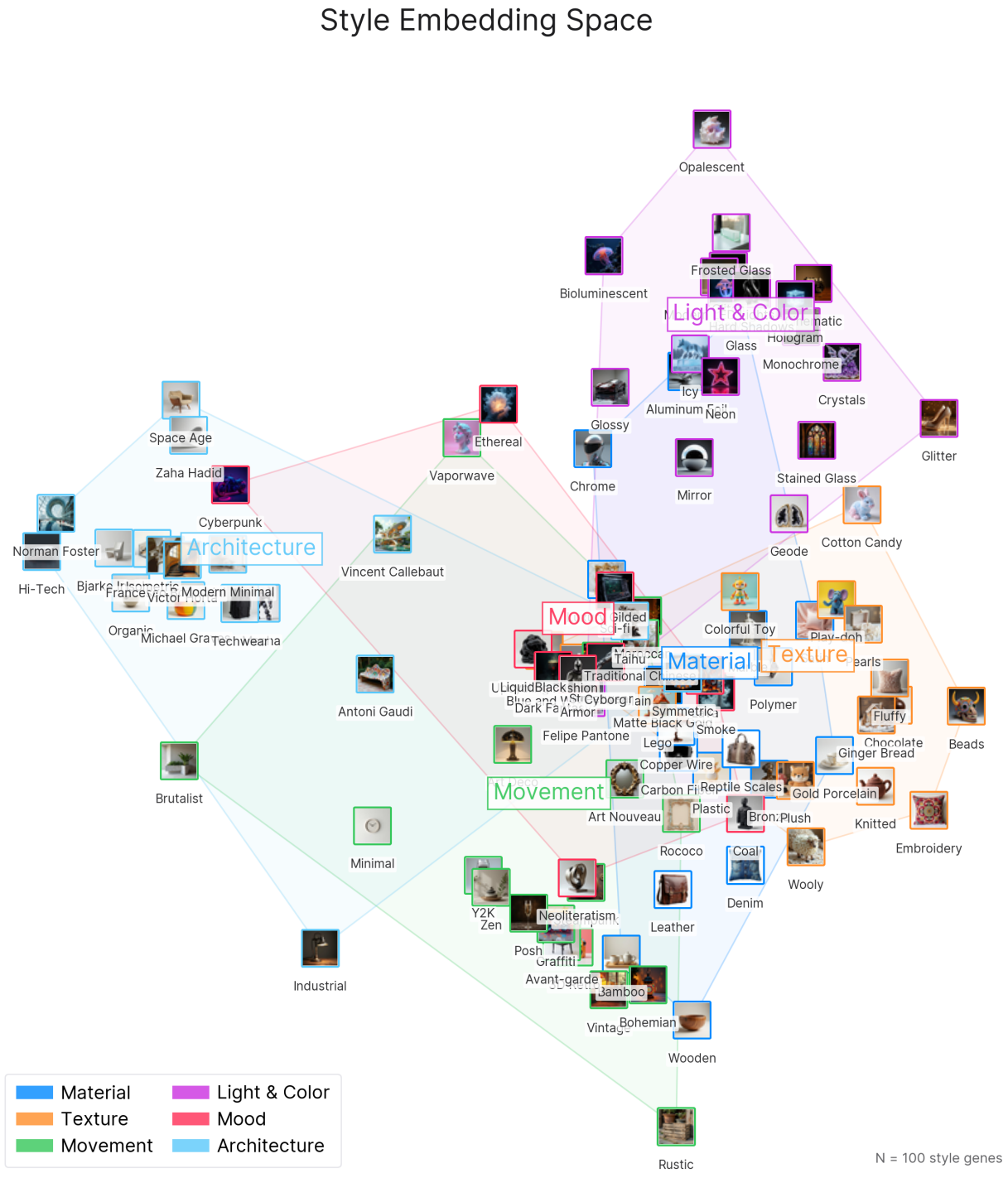


Figure 7.1. The 100-gene style space projected via PCA, with convex hulls drawn around each of the six categories. Semantically related styles cluster without explicit supervision: e.g., architectural styles (upper left), luminous effects (top), reflective materials (center), and tactile textures (right).

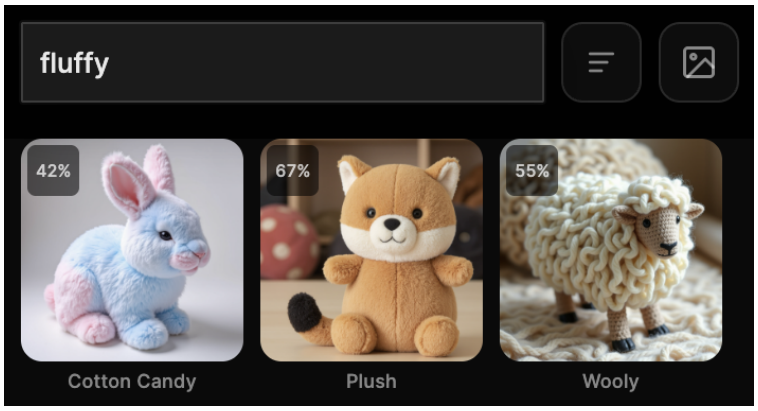


Figure 7.2. Semantic search in the gene library. Querying "fluffy" ranks all genes by semantic similarity, surfacing Plush (67%), Wooly (55%), and Cotton Candy (42%).

$$e_s = \frac{1}{N}\sum_{i=1}^{N}\Big[\text{CLIP}(\text{“a } obj_i \text{ in } style_s\text{”}) - \text{CLIP}(\text{“a } obj_i\text{”})\Big] \tag{7.1}$$

Each subtraction removes the object's contribution, and averaging across diverse objects (N categories: chair, sneaker, lamp, handbag, car, etc.) cancels remaining object bias. The result is a 768-dimensional vector that captures what the style *adds* independent of any particular object. For example, `Chrome` and `Glossy` end up nearby because they add similar reflective qualities regardless of whether they are applied to a chair or a sneaker. Figure 7.1 shows the resulting 100-gene style space projected via PCA: semantically related styles cluster together without any explicit category supervision. This differenced averaging technique is what makes downstream operations work. Complement mutation draws from the nearest 10% of genes, so tighter clustering in the embedding space means more aesthetically coherent suggestions. We validate this in Section C.1.

7.3.3 Gene Library and Style Search

GENErator's gene library contains 100 curated LoRA [263] style adapters sourced from CivitAI, HuggingFace, and our custom-trained models (Figure 7.1). Genes span six categories (material, texture, movement, light & color, mood, architecture) and are organized with filterable tags for era, fidelity, and color tone (Figure 7.2).

Designers find genes through two search modes. *Semantic search* lets designers type a natural language description (e.g., "dark and moody") and the system ranks all genes by cosine similarity between the query's CLIP embedding and each gene's style embedding. Because the embeddings capture semantic meaning rather than keyword overlap, "dark and

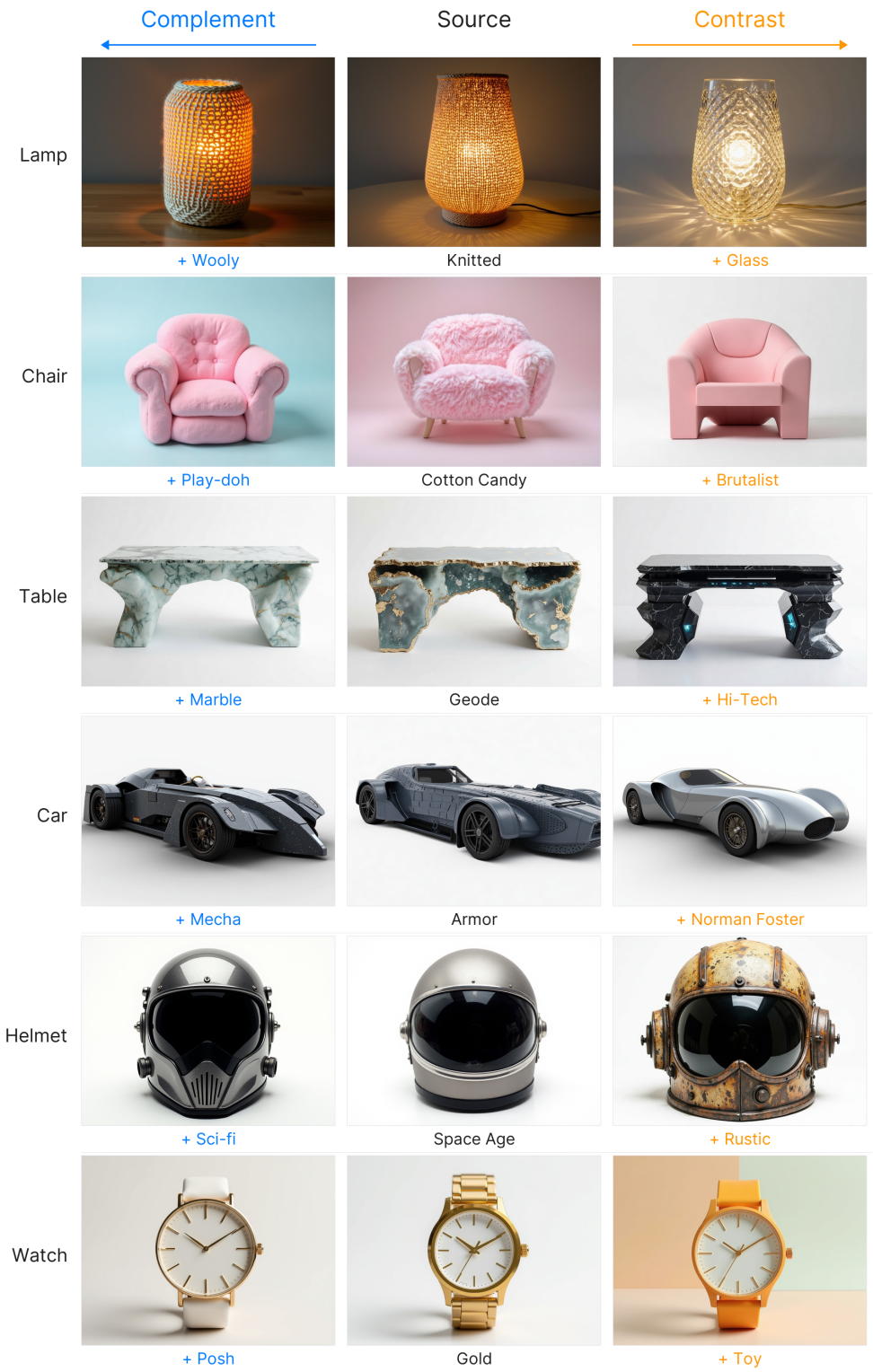


Figure 7.3. Complement vs. contrast mutation across six products. Each row shows a source gene (center) with its complement (left) and contrast (right).

moody" surfaces `Coal`, `Dark Fantasy`, `Grunge`, and `Brutalist` even though "moody" does not appear in any trigger word. *Reverse en-GENE-ering* lets designers upload any reference image and the system identifies which genes best match its aesthetic, using the CLIP image encoder in place of the text encoder. Results appear as a ranked list with match scores (e.g., "42% `Glow`, 28% `Bioluminescent`, 18% `Ethereal`"), and designers can drag any identified gene directly onto the canvas.

7.3.4 Genetic Operations

7.3.4.1 Crossover

combines two parent genes into an offspring by blending their LoRA adapters. The offspring inherits both parents' adapter sets, each scaled by a blend ratio $\alpha \in [0, 1]$ (default 0.5, adjustable via slider). Each offspring is added as a child node in the evolution tree, connected to both parents, building a branching exploration structure.

Mutation. introduces new genetic material by blending a source gene with a second gene selected from the library. GENErator offers three mutation modes that control how far from the source the offspring lands. *Complement* draws from the top 10% most similar genes in the style space (cosine similarity $\geq \tau_{90}$, approximately 0.66), yielding a pool of $\sim$10 candidate genes that refine within the same aesthetic family. *Contrast* draws from the bottom 10% most dissimilar genes (cosine similarity $\leq \tau_{10}$), introducing a different aesthetic direction. *Random* selects uniformly from the full library. The distinction between modes is not how blending occurs (all use the same weighted adapter combination as crossover) but *where* the second parent comes from in the style space (Figure 7.3).

7.3.5 Natural Selection

While crossover and mutation let designers manually guide exploration one step at a time, natural selection automates the process: the designer describes *where* they want to go, and the system evolves a population of styles toward that target.

The mechanism adapts the Simple Genetic Algorithm (SGA) [310] for semantic style exploration. In standard SGA, mutation is random and fitness is defined externally (e.g., manually by the user). GENErator makes two changes: mutation is semantically-guided (drawn from genes semantically related to the target rather than random noise), and fitness is computed automatically by comparing generated images to the designer's target description.

Given a target description t (e.g., "luxurious"), the system initializes a population of P individuals (optionally seeded from the canvas), then iterates. First, each individual is **evaluated** by generating an image and scoring how well it matches the target:

$$f(g, t) = \cos\big(\text{CLIP}_{\text{img}}(\text{Generate}(g)),\ \text{CLIP}_{\text{text}}(t)\big) \tag{7.2}$$

where Generate(g) renders a product image using gene g's LoRA adapters, and the cosine similarity between the image embedding and the target text embedding serves as the fitness score. The system then **selects** the top performers weighted by fitness, and **mutates** each by blending it with a gene drawn from the top 10% most similar to the *target description* (not the parent, as in complement mutation). The loop runs for 3–5 generations (defined by the user). The result is a grid of evolved designs that progressively embody the target aesthetic (Figure 7.4).

7.3.6 Evolution Tree

As designers crossover, mutate, and evolve genes, GENErator builds an evolution tree that tracks the full lineage of every style explored. Parent genes appear above their offspring, with connection lines showing which operation produced each child (crossover, complement, contrast, or evolution). Line thickness indicates the contribution of each parent: a thicker branch means that parent contributed more to the offspring's blend.

Each offspring receives a creative name generated by an LLM (e.g., crossing `Cotton Candy` with `Glow` might produce `Cotton Glow`). Hovering over any gene reveals a composition tooltip showing a pie chart of its genetic makeup, e.g., "45% `Cotton Candy`, 30% `Glow`, 25% `Geode`". Designers can click any node to backtrack, branching off from earlier points in the tree to explore alternative directions without losing their current branch. They can also bookmark favorite genes, save custom hybrids back to the library for future use, and export the full canvas as a visual record of their exploration.

Magic Layout. As designers add, crossover, and mutate genes, the canvas can become cluttered. Magic Layout reorganizes genes so their spatial positions on the canvas reflect their positions in the style embedding space. Library genes are projected via PCA from their precomputed embeddings; offspring are positioned as weighted averages of their parents' positions (matching the blend ratio α), with force-directed repulsion to prevent overlap. The result: after clicking Magic Layout, the canvas becomes a navigable map of the explored style territory, showing at a glance which regions have been explored, where clusters of related experiments formed, and where unexplored gaps remain. Designers can manually drag genes to override positions at any time.

We implement GENErator as a web application using an infinite canvas interface with D3.js [311] visualizations of evolution trees. Image generation uses the FLUX.1-dev model [312] with multi-LoRA support, producing each image in approximately 2 seconds. An LLM recommends an appropriate camera angle for each product type to keep style comparisons

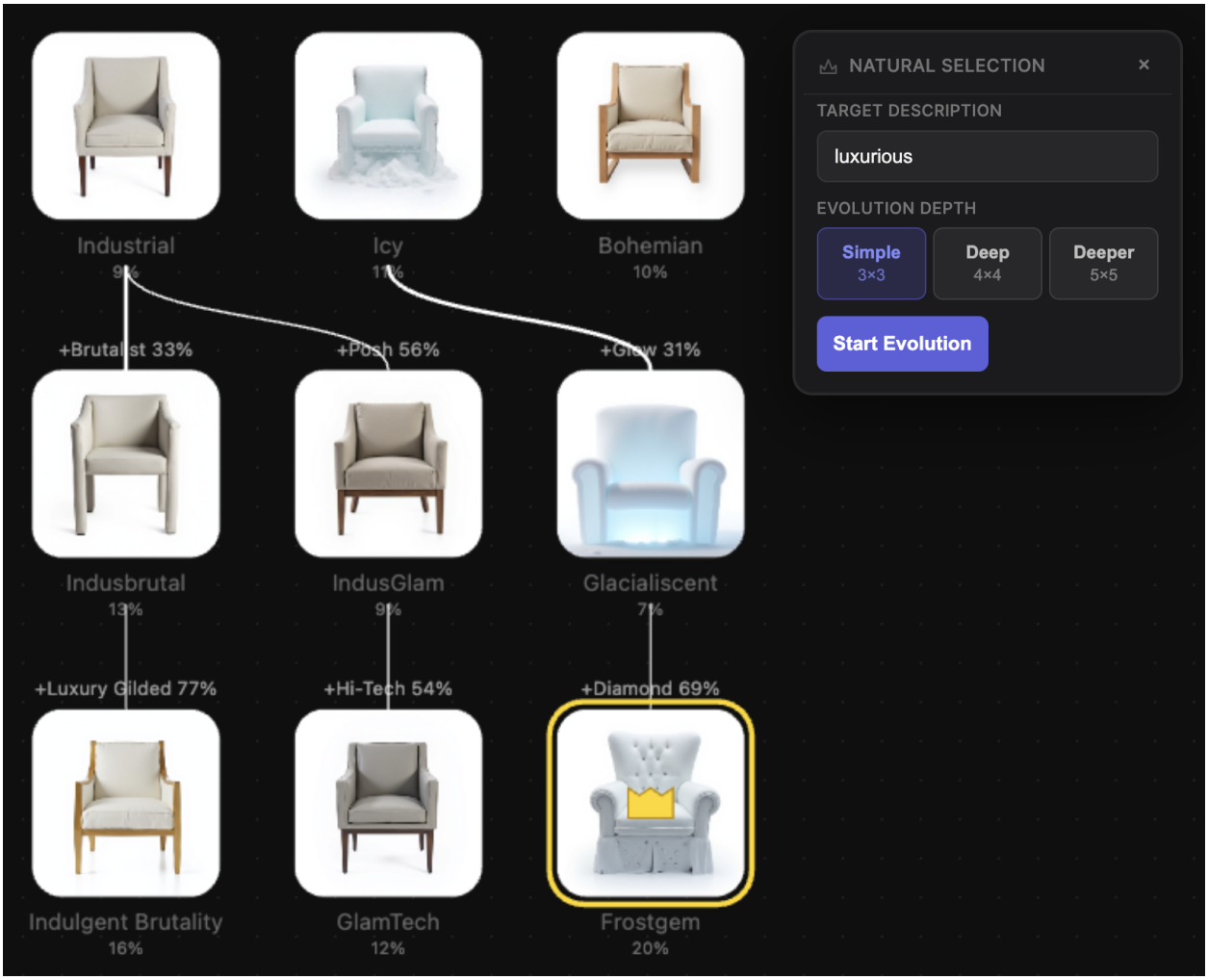


Figure 7.4. Natural selection evolving a population of three styles toward "luxurious" over three generations. In each generation, the system scores every candidate among the population by similarity to the target, selects the fittest, and mutates them using semantically related genes to produce the next generation. At the final generation, the winner is selected (gold crown).

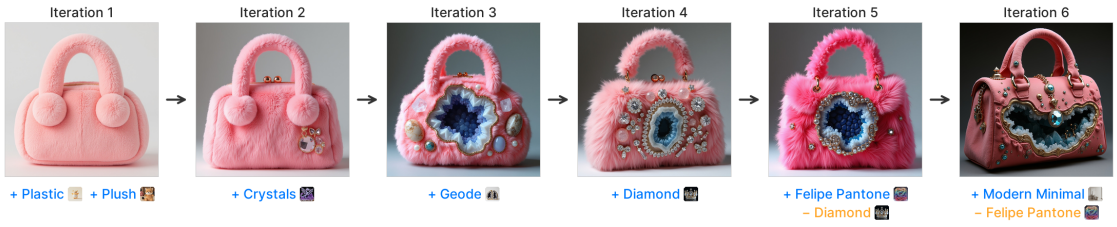


Figure 7.5. Example iteration sequence for a handbag. Starting from Plastic + Plush, the designer adds or removes genes across six iterations, each building on the previous offspring.

visually consistent (e.g., a 3/4 front view for a sneaker). Style embeddings are encoded with the CLIP ViT-L/14 model [313]. All interaction events (generations, gene additions, crossovers, mutations, evolutions, searches, bookmarks, and weight adjustments) are logged in a database for post hoc analysis.

We validate three quantitative properties of the embedding and genetic operations in Appendix C.1. Our embedding method achieves 83% tighter within-group clustering than straightforward text encoding ($d_s = 2.32$). Complement targets land 24% closer to the source than random ($d_s = 3.22$), contrast targets 27% farther ($d_s = 4.76$). Figure 7.3 shows the visual effect across six product types. Natural selection converges monotonically across all 10 tested aesthetics (Figure C.1).

7.4 User Study

We conducted a within-subjects study to understand how GENErator could support designers in exploring visual styles, its potential to change how designers interact with style spaces, and to identify areas for improvement. We compared GENErator against a baseline condition representing a standard interface for multi-style AI image generation tools used by designers today. The following outlines our participants, baseline, procedure, and measures.

7.4.1 Participants

We invited twelve participants (P1–P12, 8 male and 4 female, aged 18–53, μ=27.8, σ=9.0) to participate in the study. Participants were recruited from design programs and professional practice through known contacts and word of mouth, with the majority coming from one of the top five industrial design schools in the United States. Design experience ranged from 1 to 26 years (μ=6.0, σ=6.8), spanning product design, automotive design, CMF (color, material, finish), furniture design, architecture, and interaction design. Student participants typically had professional-level experience through a combination of industry-sponsored internships and co-op programs. Participants reported moderate familiarity with AI image generation (μ=2.5, σ=1.1, 4-point scale) and low familiarity with LoRA-based style generation specifically (μ=2.0, σ=1.0, 4-point scale). They were not exposed to GENErator prior to the study. Participants accessed both tools through a web browser. We compensated participants $40 USD for approximately 90 minutes.

7.4.2 Baseline

The baseline represents a standard interaction model for multi-style AI image generation tools, similar to widely used platforms such as Automatic1111 WebUI [288], CivitAI's on-

line generator [286], and Vizcom [287]. The baseline shares the same gene library (all 100 styles) and the same generation pipeline (same model, resolution, and quality). The difference is how designers explore. In the baseline, designers select styles from a browsable grid, adjust individual weight sliders (0–100%) to control each style's influence, and click Generate. A generation history grid displays all previous outputs. Clicking any previous result restores its selected styles and weights, enabling one-click backtracking. This is stronger than most existing tools, which require designers to manually recreate previous configurations (Figure C.4).

7.4.3 Procedure

7.4.3.1 Introduction (10 minutes)

Participants provided informed consent, completed a background questionnaire, and were given an overview of both tools.

7.4.3.2 Comparison Tasks (50 minutes)

Participants completed a design task with GENErator and another design task with the baseline. The two design tasks are "design a futuristic sneaker" and "design a luxurious handbag." We counterbalanced both the order of the conditions and the assignment of design tasks to conditions. Each task lasted 20 minutes. Participants bookmarked their favorite outputs at the end of each session and completed questionnaires about the tool they just used. Throughout the comparison tasks, participants shared their screen and verbally explained their actions and thinking (think-aloud). The researcher observed and took notes.

7.4.3.3 Free Exploration and Post-Study (30 minutes)

Participants used GENErator to design any product of their choice with full creative freedom (20 minutes). Participants designed water bottles, kitchen tables, flying cars, gaming mice, and cat food dispensers. Participants then gave feedback through open-ended questions asking them to compare the two tools, describe moments of surprise or discovery, and discuss how they might integrate each into their workflow (10 minutes).

7.4.4 Measures

For both conditions, we asked participants to complete questionnaires to capture their experience using GENErator and the baseline. We assess creativity and exploration using 5-point Likert questions adapted from the Creativity Support Index (CSI) [3]. We included

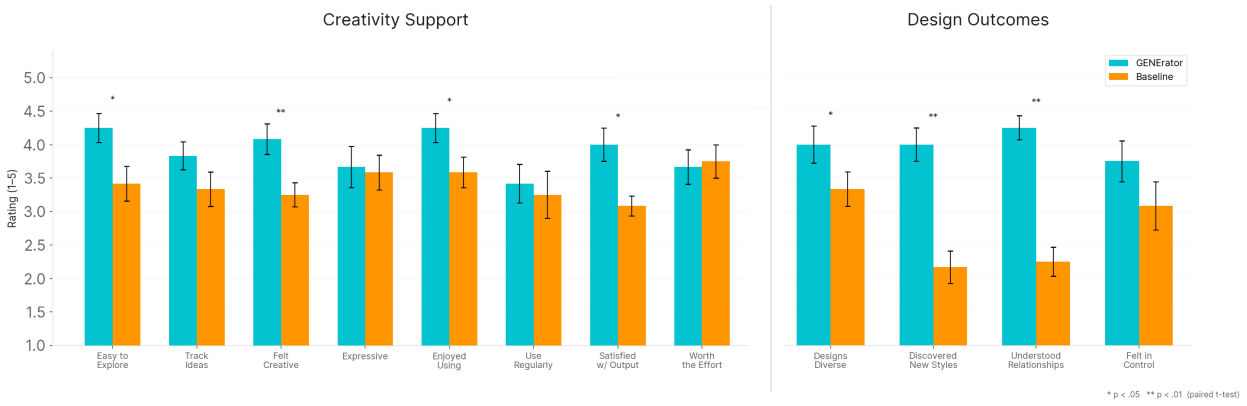


Figure 7.6. Likert results across all subjective measures. *Left:* Creativity and exploration (adapted from the Creativity Support Index [3]). GENErator rated significantly higher on ease of exploration, creativity, enjoyment, and satisfaction. *Right:* Style-specific design outcomes. GENErator rated significantly higher on diversity, unexpected discovery, and understanding of style relationships. Error bars show standard deviation.

4 custom questions targeting style-specific dimensions not covered by CSI: diversity of output, unexpected style discovery, understanding of style relationships, and sense of control. In addition, we log all interaction events to a database with timestamps, including every generation, crossover, mutation, evolution run, semantic search, bookmark, and weight adjustment. We computed post-hoc semantic similarity metrics from user-generated designs, including style space coverage (convex hull area in PCA-projected embedding space, see Figure 7.7) and mean pairwise CLIP [313] distance between outputs. We compared all questionnaire and behavioral measures using parametric paired t-tests. We report effect sizes (d_s) throughout. We reviewed qualitative data from open-ended responses, think-aloud transcripts, and researcher observation notes to support the quantitative results.

7.5 Results

All twelve participants completed both comparison tasks and the free exploration session. GENErator improved exploration breadth, style understanding, and creative ownership across both subjective (Figure 7.6) and behavioral measures (Figures 7.7, 7.8).

7.5.1 Creativity and Exploration

GENErator enables broader exploration and stronger creative ownership. Participants reported significantly higher ease of exploration when using GENErator (μ=4.25, σ=0.75) as compared to the baseline (μ=3.42, σ=0.90) ($t(11)$=2.59, p¡.05, d_s=0.75). Par-

ticipants' sense of creativity showed the largest effect among the CSI items (μ=4.08 vs. μ=3.25, $t(11)$=3.46, p¡.01, d_s=1.00). Participants also reported higher enjoyment (μ=4.25 vs. μ=3.58, p¡.05) and satisfaction with their designs (μ=4.00 vs. μ=3.08, p¡.05). As P4 put it, using GENErator felt like '*I made this*' while the baseline felt like '*the AI made this*' (P4).

This was supported by behavioral data: participants explored 2.2 times more unique base styles with GENErator (μ=16.8, σ=4.5) as compared to the baseline (μ=7.5, σ=2.1) ($t(11)$=6.26, p¡.01, d_s=2.62), and bookmarked 5.2 times more outputs (μ=9.2, σ=2.2 vs. μ=1.8, σ=1.4, p¡.01, d_s=3.94). Participants also rated their outputs as significantly more diverse (μ=4.00 vs. μ=3.33, p¡.05, d_s=0.86). Notably, the mean pairwise cosine distance between explored styles was comparable between conditions (GENErator μ=0.46 vs. baseline μ=0.47, p=.13), suggesting that GENErator's advantage comes from visiting more *regions* of the style space rather than sampling more distant individual styles (Figure **7.7**). Figure **C.3** shows P12's trajectories as a representative example.

The remaining CSI items (idea tracking, intent expression, regular use, effort-reward, and control) did not differ significantly between conditions. GENErator did not degrade these aspects relative to the baseline.

7.5.2 Style Understanding and Discovery

GENErator makes style relationships understandable and leads to more serendipitous discoveries. The ability to understand the relationships between styles showed the largest subjective effect (d_s=1.66). Participants rated significantly higher understanding when using GENErator (μ=4.25, σ=0.62) as compared to the baseline (μ=2.25, σ=0.75) ($t(11)$=5.74, p¡.01). The evolution tree interface was noted as a key mechanism: P6 commented '*you can see the whole tree of styles I explored and eventually see how the end product was created*' (P6), and P3 described '*the canvas tool (GENErator) feels more reliable because I can understand the generation process and adjust it with clearer direction*' (P3). In the baseline, the same participants called it '*all blind box behind me*' (P3), and P5 noted '*the name of the style doesn't mean anything because the outcome is very different from the style*' (P5).

Participants also reported significantly more unexpected discoveries with GENErator (μ=4.00 vs. μ=2.17, $t(11)$=7.61, p¡.01, d_s=2.20). Contrast mutation was a key driver. P7 applied contrast to a mirrored metal gene, expecting something dull. Instead, GENErator surfaced a neon light gene: '*I expected something more bland... like the opposite of mirror*' (P7). The result was not a literal opposite but an unexpected aesthetic direction. Later, exploring a flying car concept, P7 observed: '*Guess it associates flying with translucency. That's inter-*

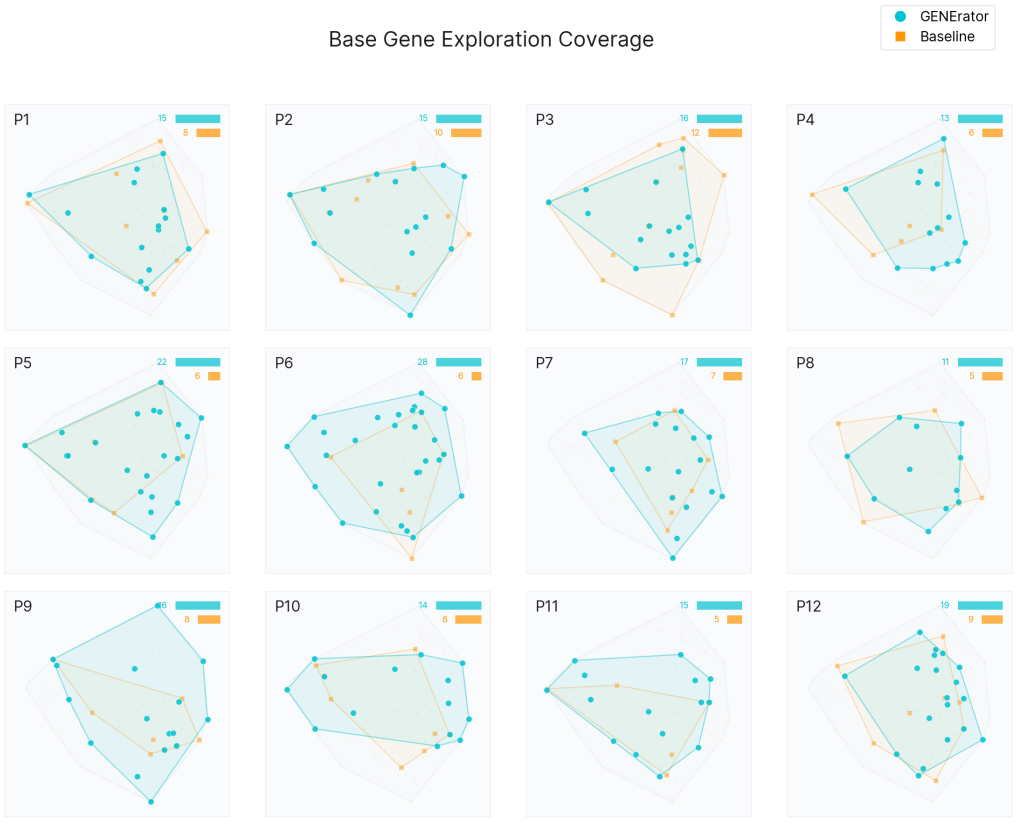


Figure 7.7. Per-participant style space coverage in PCA-projected CLIP space. Convex hulls show GENErator (teal) vs. baseline (orange). The faded gray convex hull shows the complete style space (the same one as Figure 7.1). GENErator participants generally covered more of the style space.

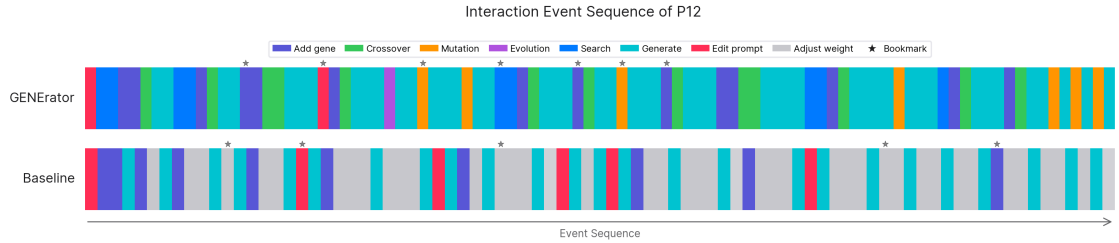


Figure 7.8. Interaction event sequence for P12. The baseline (bottom) is dominated by weight adjustments (gray). GENErator (top) shows crossover (green), mutation (orange), evolution (purple), and search (blue), with more bookmarks (stars). See Figure C.6 for all 12 participants.

esting' (P7). Not all contrast suggestions were immediately accepted, but participants valued that surprises were interpretable. Not all contrast suggestions were immediately accepted, but participants valued that surprises were interpretable: P6 described an unexpected glass texture from crossover as '*not exactly like the [parent] styles but also something that came out of merging them*' (P6). Figure C.2 visualizes this pattern across all participants: complement mutations (blue) generally stay closer to the source gene and contrast mutations (orange) jump to more distant regions.

7.5.3 Interaction Patterns

GENErator shifts designers from repeated parameter tweaking to structured, orchestrated exploration. Interaction logs revealed a clear shift in how designers explored the style space (Figure 7.8). With the baseline, participants followed a linear cycle: search the library, select styles, adjust weight sliders, generate, then start over. Each generation largely started from a new slider configuration rather than building on previous results. The dominant action was weight adjustment (μ=47.8, σ=17.0 per session), accounting for nearly half of all interactions – yet adjusting a slider from 70% to 80% gives little indication of how the output will change. P9 commented '*I just keep changing sliders randomly. It's hard to understand what's happening.*' (P9).

With GENErator, participants built on previous outputs through crossover, mutation, and evolution rather than reconfiguring each generation independently. Weight adjustments dropped significantly to μ=1.3 with GENErator (p¡.01, d_s=3.87), with participants leveraging crossover (μ=12.3, σ=5.9), mutation (μ=7.1, σ=1.1), and evolution (μ=1.6, σ=1.3). Prompt editing also dropped significantly (μ=1.8 vs. μ=5.1, p¡.01, d_s=2.56), suggesting that genetic operations substantially reduced the role of prompt engineering (Figure C.5). When P9 ran natural selection (evolution) to evolve `Lego` toward a futuristic aesthetic, the

progression was visible: '*The Lego started blocky and then generation by generation it got sleeker*' (P9).
P10, with over 25 years of design experience, described GENErator like orchestrating a design team: '*Your team of designers is constantly generating ideas... and you are combining and you say, how about a little bit of this one with this part of that one*' (P10). Several participants placed GENErator within their broader workflow: P10 saw it as '*intended for the mood board phase*' (P10).

7.6 Discussion

A core challenge we grappled with in this work was helping designers explore diverse styles without resorting to random trial-and-error. Current tools mainly sit at two extremes: fixation, where vocabulary-bounded search keeps designers circling the same aesthetic territory, or noise, where random mutation in traditional IEC produces user fatigue [275, 276]. Our results suggest GENErator enables a middle ground we call *guided divergence*: exploration that is simultaneously broader and more intentional than either extreme. Each genetic operation plays a distinct role: complement refines within an aesthetic family, contrast leaps to distant territory, crossover blends two directions, and natural selection converges toward a target. That participants used both complement and contrast throughout their sessions (Section 7.5.3) suggests they valued surprise as much as refinement. As P12 put it, GENErator allowed them to '*explore without having to know what [they are] looking for.*' The content-independent style embedding is what makes this possible: our validation shows that without it, within-group similarity drops by 83%, which would make complement less coherent and contrast less distinct (Section C.1). This principle may generalize to other domains where AI models can be embedded in a meaningful space (e.g., fonts, music, or motion styles).
Our results also suggest a structural limitation of slider-based interfaces: weight sliders offer parametric control without semantic meaning. Genetic operations replace opaque parameter tuning with semantically interpretable composition. Understanding style relationships and how styles can be combined was the strongest subjective finding (d_s=1.66), and the evolution tree was cited as a primary mechanism. Participants who could see how designs derived from parent styles reported feeling more confident branching into unfamiliar territory, suggesting that transparency may enable bolder exploration. Where the baseline's history grid shows *what was generated*, the evolution tree shows *how designs relate to each other*. Participants consistently described a shift from '*I found this*' to '*I designed this*' (Section 7.5.3). A key enabler was the gene abstraction itself: where the baseline required designers to express intent through arbitrary numerical weights, GENErator let them express intent through

aesthetic relationships: blend these two, surprise me, take this somewhere luxurious.

7.6.1 Limitations and Future Work

There are several avenues for future work collected from participants' feedback and our observations. First, GENErator operates at the level of overall visual style. Several participants expressed interest in controlling specific visual elements, such as mutating only the color palette while keeping the form, or crossing over the texture of one design with the silhouette of another. Second, GENErator currently exists as a standalone exploration tool. P10 described it as '*for the mood board phase*' (P10), suggesting it fits early in the design process. Exporting material parameters or color palettes to downstream tools such as Figma, Rhino, or KeyShot could bridge the gap between exploration and production. Third, our study compares the full GENErator system to the baseline. Isolating the contribution of each genetic operation (crossover, mutation, evolution) could be a direction for a future ablation study. Fourth, we measured exploration breadth and subjective experience but did not independently evaluate output quality. Compared to the baseline, participants bookmarked 5.2 times more outputs (p¡.01), a behavioral indicator of perceived design value taken during the task rather than a post-hoc rating. Participants also rated significantly higher satisfaction (p¡.05, d_s=0.75). Additional expert quality ratings could complement these self-report measures. Fifth, natural selection uses CLIP similarity as the fitness function. While convergence is consistent across 10 tested aesthetics (Figure C.1), CLIP was not trained for aesthetic judgment specifically. Future work could explore aesthetic-specific metrics. Sixth, while no participant reported fatigue in our sessions, longer-horizon studies with explicit fatigue measures would test whether semantic guidance sustains its advantage as exploration scales up. Finally, our library of 100 genes was curated by the research team. The embedding technique itself scales (computing additional embeddings is inexpensive), but deployment with larger ecosystems such as CivitAI's [286] 100K+ styles would test whether the experience remains effective at scale.

7.7 Conclusion

This research identifies the challenges designers face when exploring visual styles with AI generation tools and proposes a content-independent style embedding technique for making style spaces navigable. The research prototype, GENErator, treats each visual style as a gene in a semantic space, enabling designers to crossover, mutate, and evolve styles through genetic operations grounded in these embeddings. In a within-subjects study, participants using GENErator explored 2.2 times more styles, reported improved understanding of how styles

relate to one another, and adopted structured, branching exploration patterns rather than blind trial-and-error. Complement and contrast mutations surfaced styles that participants had not previously encountered, and the evolution tree made the full compositional history of every design traceable. More broadly, GENErator demonstrates that genetic abstractions over semantic embedding spaces offer a viable path for navigating large style libraries. As generative AI makes the space of possible designs vastly larger than any designer can explore alone, we hope this work informs future tools that make these latent spaces not just visible, but operable.

Reading GENErator through the Human-AI-Interface Triangle: style adapters embedded in a content-independent semantic space give compositional handles over aesthetic direction (controllability), genetic operations (crossover, complement and contrast mutation, natural selection) support the moves remixing requires (affordance), and the evolution tree draws on a biological metaphor designers can read, so each operation produces the categorical effect it implies, as the technical validation that follows demonstrates (intuitiveness). The four systems in this thesis together cover the navigate, compose, iterate, and remix tasks. The next chapter synthesizes the patterns that emerge across the four cases.

Chapter 8

Discussion and Conclusion

A core challenge I grappled with in this work was how to give designers control over AI without forcing them to learn the internals of each model. The four systems took different routes through that problem. However, a number of patterns emerged across all of them. In this chapter, I examine those patterns, distill a set of design guidelines for AI-powered design exploration, discuss the limitations of this work, and outline directions for future research.

8.1 Research Contributions

In this thesis, I develop a framework for AI steering interfaces that support design exploration. My core contribution is to show how AI may be made steerable through interactions that designers already understand, such as navigating a map, assembling pieces, sketching, and mixing styles.

The Human-AI-Interface (HAI) Triangle. I identify three components that must align in an effective AI steering interface: design tasks (what designers want to do), AI mechanisms (what models can actually control), and interface representations (what designers manipulate). Alignment is defined through three conditions, one per vertex: *controllability* for the AI (the mechanism exposes handles a designer can act on), *affordance* for the Human (the controls match the moves the task requires), and *intuitiveness* for the Interface (the representation draws on conventions designers already work with, so they can anticipate what each control will do). When the representation bridges the human task and the AI mechanism, steering becomes a form of direct manipulation. For example, a map may let designers navigate a latent space, and puzzle pieces may let them compose model capabilities. However, when the three conditions do not align, designers often struggle even with a powerful model underneath. The triangle therefore serves two purposes: diagnosing an existing tool by identifying which condition fails, and designing a new one by working backward from the task and the mechanism to the representation that connects them.

A design space of AI steering tasks. I map four modes of steering AI for design exploration, namely navigate, compose, iterate, and remix, each aligned with a distinct AI mechanism and interface representation (Table 1.1). We implemented each alignment as a working system and evaluated it in a user study (N=10–14), with the per-system findings summarized in Table 8.1. Across the four studies, participants were generally able to accomplish work that the baselines did not enable.

Design guidelines for AI steering interfaces. Several patterns extend beyond any single system, and four of them recurred across all four. First, AI models encode rich possibility spaces that remain hidden without the right interface, and making this knowledge visible and manipulable is the core design challenge. Second, the exploration process itself often creates value, and not only the final output. Third, familiar interaction metaphors such as maps, puzzles, sketches, and style mixing may lower barriers by letting users draw on skills they already have. Finally, fast feedback appears to support branching exploration rather than early commitment to a single path. Each of these is developed as a guideline in Section 8.2.

8.2 Design Guidelines for AI Steering Interfaces

While the HAI Triangle is a framework for *analyzing* AI steering interfaces, the guidelines below are heuristics for *building* them. I distilled them from what worked across the four systems, what participants struggled with, and what appeared to distinguish effective steering from frustration.

8.2.1 Make the Hidden Visible and Manipulable

Modern AI encodes large possibility spaces that stay hidden without the right interface. Much of what AI offers lies not in any single output but in the spaces it can move through. These spaces, which include latent representations, model compositions, conditioning pathways, and style embeddings, stay hidden from users without an interface that brings them within reach.

Each system made a different aspect of AI capability visible and manipulable:

In each case, making the capability visible appeared to lead to specific behaviors that the baseline did not produce. VideoMap editors used latent-space proximity to surface match-cut candidates across visually similar but semantically distant clips, which a filename search cannot reach. Jigsaw participants used the catalog to find unfamiliar models. As P9 put it, they could "*discover new AI abilities I was previously unaware of.*" They then connected the models into pipelines, testing feasibility before committing. Inkspire participants treated the scaffolds as a way to read the AI's interpretation of their sketch before deciding the next

Table 8.1. How each system surfaced latent AI capabilities through interface representations.

System	What Was Latent	How It Was Surfaced
VideoMap	Semantic/visual relationships between frames	Maps
Jigsaw	Composable model capabilities	Puzzle pieces
Inkspire	AI's interpretation of structure	Sketching
GENErator	Mixable style-specific models	Evolution

stroke. P11 described this as a way to "*know where the current iteration was so I could tell where I'd like to move it next.*" GENErator participants used complement and contrast mutations to encounter styles they had not previously known existed, sometimes discovering them as offspring of unrelated parents.

A design heuristic follows from this pattern. When building an AI tool, we may first ask *"what does this model know that users cannot currently see or manipulate?"* The answer often points toward the interface representation needed to complete the triangle.

8.2.2 Design for Exploration

All four systems supported broader exploration than their baselines, and the magnitude of the difference is what makes this guideline concrete rather than rhetorical. Inkspire participants scored significantly higher on the Creativity Support Index's exploration dimension (μ=5.83 vs. 3.83, d_s=1.13) and inspiration dimension (μ=5.92 vs. 4.00, d_s=0.99) than they did with ControlNet. GENErator participants explored 2.2 times more styles and bookmarked 5.2 times more outputs than the baseline. Jigsaw participants who started from the same set of puzzle pieces ended with pipelines they did not plan in advance, often discovering AI capabilities (P9: "*discover new AI abilities I was previously unaware of*") in the act of assembly. VideoMap editors found visual connections such as match cuts and semantic neighbors that depended on the map view, with no equivalent action in a timeline editor.

We designed these systems so that the exploration process produces value that the final artifact alone does not capture. For example, with VideoMap much of the work is the act of moving through the map, and the final edit is only one possible product of that exploration. Jigsaw's pieces are similarly designed to be tried in combinations, and users often learn what their tools can do by snapping them together in ways they had not planned. Inkspire treats each sketch stroke as part of an ongoing conversation with the model, and GENErator's selection and crossover steps are themselves the design activity, in that by the time a user

has bred a final image they have effectively explored a region of the style space.

Two design implications follow from this observation. The first is that the right baseline for an AI exploration tool may not be whether it can produce a usable artifact, but whether the path to that artifact lets the designer cover more of the space than the alternative does. The second is that designers of such systems should be wary of optimization targets that compress the path. A one-shot tool that produces a perfect output on the first try removes exactly the work that this guideline suggests creates value. The four systems trade some efficiency on a single output for substantially more coverage of the surrounding space, and our studies suggest that designers may want this trade.

8.2.3 Familiar Metaphors Lower Barriers

Resnick and colleagues articulated a principle for creative tools: "low floors, wide walls, high ceilings" [166], meaning that a tool should be easy to start with, supportive of diverse approaches, and capable of sophisticated outcomes. I found that the triangle framework helps achieve all three of these properties.

The interface representation provides the low floor, in that maps, puzzles, sketches, and evolution trees are all things people already understand. Participants in our studies were generally able to begin exploring immediately because the metaphors were familiar to them: they already knew how to navigate a map, snap puzzle pieces together, sketch on a canvas, or mix and evolve styles, and they did not need to learn a new interaction paradigm before they could start working.

The AI mechanism provides the high ceiling, in that the metaphors sit on top of mechanisms that encode relationships designers could not specify by hand: CLIP's semantic organization of visual content, foundation models' multimodal chaining, ControlNet's structural conditioning, LoRA's style composition. The ceiling is set by what these mechanisms can do, not by what the metaphor itself can express.

The alignment between the interface representation and the AI mechanism provides the wide walls, by which I mean multiple valid paths through the space. For example, in VideoMap some participants found match cuts through the shape lens while others followed narrative connections through the semantic lens. Across Jigsaw sessions, participants who started from the same set of pieces often ended with very different pipelines, and personal sketching styles developed over the course of the Inkspire studies that the system was able to interpret. With GENErator, mutation and crossover surfaced style combinations that participants had not planned to explore. The interface supported exploration rather than prescribing a single correct workflow.

8.2.4 Immediate Feedback Enables Branching

These systems lean toward *play* rather than *work*, in the sense that work tends to push reward to the end of a task while play keeps users engaged throughout. For example, Inkspire generates on every stroke with a feedback loop measured in milliseconds, a crossover in GENErator produces visible offspring right away, and dragging in VideoMap surfaces nearby frames as the cursor moves.

Fast feedback reshapes exploration in a specific way: when designers see results in milliseconds, they try more variations, backtrack more freely, and stumble onto unexpected directions. Slow feedback pushes them to commit early; fast feedback gives them room to branch.

Software development calls this principle hot reloading: when code changes take effect without a restart, programmers stop batching changes and start exploring [314]. We apply the same principle to AI generation in the systems presented here. A sketch stroke reruns ControlNet, a crossover reblends LoRAs, and dragging on the map re-projects the latent space. Each gesture causes the model to update in place. A representation that needs a Generate button to refresh treats the AI output as a destination, whereas a representation that updates as the designer acts treats it as part of the work surface. Jigsaw is slower on purpose. This is because composing a pipeline is a different kind of exploration than steering a single generation. Feedback does not have to be instantaneous, but its rhythm should match the steering work the representation is doing.

I did not manipulate feedback latency as an independent variable in any of the four studies, so this guideline is a cross-system observation rather than a tested causal claim. It is consistent with the participant behavior described above and with prior work on responsive interfaces [314], but a controlled study that compared the same representation under different feedback rhythms would be needed to establish the mechanism cleanly.

8.2.5 Match Interface Complexity to AI Role

Across the four systems, the AI's role varies along a spectrum from stimuli to collaborator. When AI performance is well below human level, its outputs generally serve as external stimuli, which is to say raw material for humans to process, filter, and refine. However, as AI approaches and exceeds human-level performance on a specific task, it may take initiative, offer alternatives, and adapt to the user's direction, and in doing so becomes a collaborator rather than a supplier of raw material.

The systems in this thesis sit at different points along this spectrum:

Table 8.2. How interface requirements vary with the AI's role in the human-AI relationship.

Position	System(s)	AI's Role	Interface Needs
Navigation	VideoMap	Organizes space for human to explore	Spatial overview, smooth traversal
Orchestration	Jigsaw	Executes what human assembles	Visible connections, testable sub-pipelines
Collaboration	Inkspire, GENErator	Interprets input / generates variation	Fast feedback, visible AI state

Inkspire and GENErator are both collaborative but in different ways: Inkspire interprets user input and shows what it "understood" (the scaffold), while GENErator generates variation that may surprise the user with possibilities they could not have imagined.

As the AI moves along this spectrum from stimuli to collaborator, the interface must do more. Stimuli may be presented simply as a feed to scroll through, whereas collaboration typically requires richer protocols, since the interface has to carry the designer's intent to the AI and the AI's state back to the designer: what it understood, what it can do, where the exploration currently sits. As AI capability rises in a given domain, the corresponding interfaces likely need to shift along this spectrum, though the specific representation that fits the collaborative end is an open design question, not one this thesis resolves.

8.2.6 Tensions Between Guidelines

The five guidelines are not jointly optimizable, and several of them pull against each other, so any working tool has to choose which to prioritize.

Controllability pulls against simplicity of representation, because exposing more handles gives designers more to manipulate, but the most intuitive representations lower barriers precisely by keeping much of the mechanism behind a recognizable shape. GENErator's evolution tree exposes many handles and takes longer to learn; VideoMap's map exposes fewer and is easier to pick up.

Exploration pulls against immediacy, since some moments call for deliberation, such as comparing options or backtracking, while others call for fast feedback, such as sketching or mutating. A rule of "fast feedback always" would conflict with a rule of "let the designer pause and think."

Familiarity pulls against collaboration, in that more capable AI needs richer protocols for negotiating direction with the designer. Those protocols may not yet have familiar metaphors, because the underlying interaction patterns are new.

The framework does not resolve these tensions, but it does give a vocabulary for naming them and a way to reason about which alignment matters most for the specific task the tool is built around.

8.3 Future Directions

Having built the framework and demonstrated it through four working systems, I next want to push it into new territory along two axes derived from its structure. Three directions extend the triangle's vertices: applying the framework to new task domains (§**8.3.1**), designing new representations for emerging AI mechanisms (§**8.3.2**), and pushing the conditions of controllability, affordance, and intuitiveness upstream into how models are built (§**8.3.3**). Three more directions extend the triangle's edges, one per edge: AI steering interfaces along the Human → AI edge (§**8.3.4**), generative interfaces where AI produces the steering surface along the AI → Interface edge (§**8.3.5**), and adaptive interfaces where the representation shifts with each user along the Interface → Human edge (§**8.3.6**).

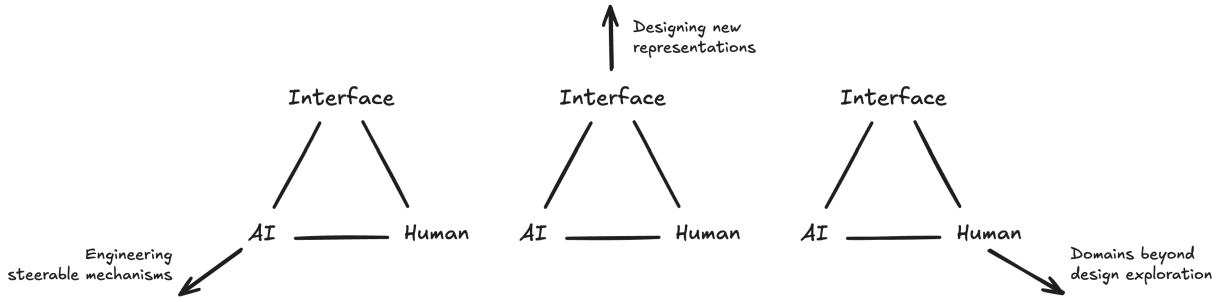


Figure 8.1. Three future directions, each extending one vertex of the HAI Triangle. Engineering steerable mechanisms extends the AI vertex, designing models for controllability from the start. Designing new representations extends the Interface vertex, creating surfaces for emerging AI mechanisms. Domains beyond design exploration extends the Human vertex, testing which representations transfer to new tasks.

8.3.1 Beyond Design Exploration

The four design tasks generalize beyond creative design. However, I have only tested the framework within it. For example, a researcher traversing AI-embedded literature faces the navigate task, as does an analyst exploring customer segments in high-dimensional data or a writer surveying narrative directions. Composition fits developers assembling AI coding agents and scientists building analysis pipelines. Iteration appears wherever a designer

partially specifies and the AI completes: animation, music arrangement, architectural floor planning. Evolution applies to preference-led work without clear targets: musicians breeding beats, fashion designers evolving garments, game designers exploring character variations. In each case, the framework predicts that representations similar to maps, puzzle pieces, sketches, and evolution trees should support effective steering, but the framework's three conditions must hold under new mechanisms: controllability through embedding-based retrieval rather than CLIP, intuitiveness under conventions specific to each domain.

Some domains will resist transfer altogether. High-stakes fields such as medicine and law require accountability alongside control: representations must expose audit trails, constrain certain steering paths, and require confirmation before consequential decisions. Domains with long time horizons, like training a personal assistant over months or curating a recommendation feed over years, need representations that surface patterns across sessions rather than within one. Domains with multiple stakeholders raise coordination requirements: whose intent is currently active, how conflicts get resolved, what prior decisions led to the current state.

My immediate next step here is to apply the framework to a domain where the task structure matches but the AI mechanism differs from those in this thesis. The test is whether the framework's predictions hold across mechanism shifts: embedding-based retrieval for text rather than CLIP for images, or differentiable rendering rather than diffusion conditioning. Building and evaluating such a tool would directly address the framework's central limitation: it has so far described its own systems, not been used to design new ones.

8.3.2 Designing New Representations

Every new AI mechanism creates a representation design problem: the mechanism exists, the tasks are real, but no representation yet makes the mechanism steerable. This thesis demonstrated four representations for four mechanisms, and new mechanisms will keep arriving. Consider chain-of-thought reasoning, where models can now show their intermediate reasoning steps but users have no way to redirect flawed logic in mid-stream. A reasoning tree representation could let users trace AI thinking step-by-step, branch at decision points, prune unproductive paths, and edit intermediate conclusions before the model continues. With such a representation, chain-of-thought stops being a transparency feature and starts working as a steering mechanism. In-context learning is similar in shape but different in what it needs: an example workbench could let users organize examples, weight their importance, test combinations, and observe how each example shapes model behavior, teaching AI by showing rather than describing. Attention manipulation extends naturally: an interactive attention editor could overlay attention weights on inputs and let users boost attention to

overlooked regions, suppress it on distracting elements, and regenerate with the modified pattern. Memory mechanisms in newer architectures need similar treatment: a memory browser to inspect, edit, and reinforce what the AI has retained across sessions.

In each of these cases, a new mechanism appears first and the design problem is to surface it. However, the relationship between mechanisms and representations runs in both directions. Sometimes the representation comes first, and the question becomes what mechanism could support it, which is the subject of §**8.3.3**. My research agenda develops both sides: designing interfaces for mechanisms that already exist, and shaping mechanisms to fit interfaces that designers need.

8.3.3 Engineering Steerable Mechanisms

ControlNet and GLIGEN were engineered for steering from the start. ControlNet exposes structural conditioning because its designers recognized that spatial control matters for creative work, and GLIGEN exposes layout control because object placement needs direct manipulation. However, most AI mechanisms were not engineered this way. They typically emerged from optimization objectives unrelated to human steering, leaving HCI researchers to design interfaces around whatever happened to be exposed.

What makes an AI mechanism controllable? Four properties stand out across the four systems in this thesis. *Interpretable dimensions*: the mechanism's parameters should map to concepts users already understand, as CLIP's joint image-text space gave VideoMap and GENErator semantic axes designers could name. *Smooth interpolation*: small input changes should produce small output changes, as ControlNet's structural conditioning gave Inkspire anticipable updates per stroke. *Anticipable responses*: users should be able to predict how their steering will affect the output, the property that distinguishes the four systems from raw-parameter sliders. *Composability*: mechanisms should combine cleanly, as foundation model chaining made Jigsaw's puzzle metaphor tractable.

These properties may be designed into models rather than discovered after the fact. By specifying what alignment requires, the framework therefore becomes a target for model architecture rather than a vocabulary for describing finished tools. My research agenda pursues this from both sides: designing interfaces that surface new mechanisms (§**8.3.2**) and shaping new mechanisms to fit the interfaces designers need. This dual involvement is already evident in the technical contributions of this thesis. GENErator's content-independent style embedding (Chapter **7**), Inkspire's sketch scaffolding pipeline (Chapter **6**), and VideoMap's multi-lens latent space encoding (Chapter **4**) each required substantive engineering on the mechanism side, not only interface design. A concrete first project would be to train a model whose latent space is structured for controllability from the start, with axes that align with

task affordances and a structure that supports an intuitive representation, and compare it against a conventionally-designed equivalent on a specific design exploration task. If this works, the framework may become useful as a specification for model architecture, and not only as a post-hoc analysis tool.

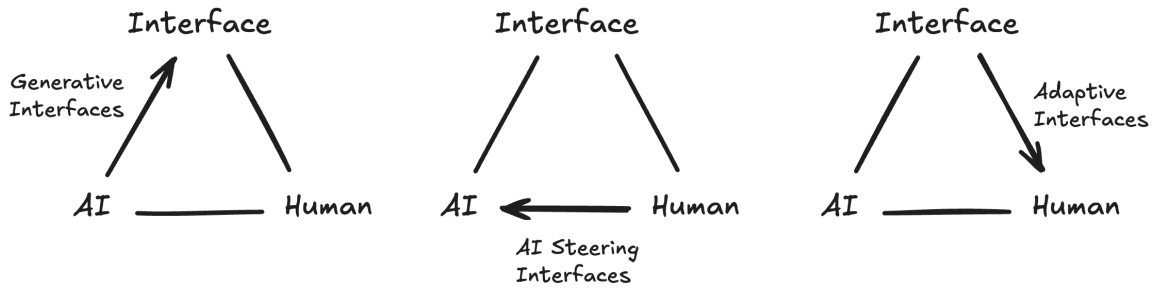


Figure 8.2. Three future directions, each along an edge of the triangle. Generative interfaces extend the AI → Interface edge, with the AI producing the steering surface itself given a task and an available mechanism. AI steering interfaces extend the Human → AI edge, continuing the thesis's central direction with new tasks, mechanisms, and contexts. Adaptive interfaces extend the Interface → Human edge, with the representation shifting to each user's abilities and preferences.

The three directions above extend each vertex of the triangle. The next three extend the edges: each one-way flow between human, AI, and interface is its own research direction. None has been tested in this thesis, and each connects to a substantial literature I cannot do full justice to here.

8.3.4 AI Steering Interfaces

The Human → AI edge is the center of my research agenda. The four systems in this thesis demonstrate AI steering interfaces for one designer, one design task, and one session at a time, and they are starting points rather than end points.

The agenda is to build many more: for emerging AI mechanisms, new design domains, and design tasks beyond the four developed here. The vertex extensions above (§8.3.1–§8.3.3) describe what each vertex needs from the framework. The work at this edge is therefore the construction itself, carried out system by system.

A near-term direction is integration across tasks, since the thesis treats navigate, compose, iterate, and remix as separate tools but design exploration in practice moves between them. Consider a designer working on a short video, who might survey footage in a VideoMap-like

view, pipe selected clips through a Jigsaw-style composition with audio and effects, iterate on the result through Inkspire-style sketch overlays, and remix the style with GENErator-like genetic operations, all in one session and in one tool. A concrete project would be a unified design environment where the four tasks share state. The research question is whether the HAI Triangle's three conditions hold across task boundaries: does controllability transfer when the AI mechanism shifts from latent space to model chaining, and does intuitiveness survive when designers carry one representation's conventions into another?

8.3.5 Generative Interfaces

If AI can generate content, it can also generate the interfaces that steer that content. The HAI Triangle makes this concrete: given a task and an available mechanism, the system must produce a representation that aligns them. The hard parts of doing so are inference and usability. Inference is difficult because the system must guess the user's current task without being able to ask, and that task may shift mid-interaction. Usability is difficult because a generated representation that satisfies the framework is not automatically usable, which is to say that alignment is necessary but not sufficient.

A concrete project would be a generative interface system that, given an AI mechanism and a task category from the four developed here, outputs a working steering interface aligned through the framework. Success can be evaluated against hand-designed equivalents: do designers using the generated interfaces reach comparable task outcomes? If the framework can drive automatic interface generation, interface design may then scale to every new AI mechanism without manual work.

8.3.6 Adaptive Interfaces

AI steering interfaces today are predominantly visual and text-driven, leaving out users who cannot see screens, manipulate fine pointers, or articulate intent in writing. Adaptive accessibility, by which I mean the problem of shifting the representation to fit the user's abilities, is among the more pressing open problems in HCI. Wobbrock's ability-based design [315] provides a foundation here. For example, we might consider a sonified VideoMap for users with low vision, or a voice-driven Jigsaw for users with motor impairments. The framework's three conditions still apply, but each must hold under a different modality: controllability through speech-driven handles, affordance through non-visual moves, intuitiveness through conventions designers already use within that modality.

A concrete project would be to build accessibility-preserving variants of all four systems and evaluate whether the framework's alignment conditions hold when the representation shifts

modality. The deeper research question is whether the HAI Triangle can generate accessibility variants systematically, predicting which controllability, affordance, and intuitiveness moves transfer across modalities, or whether each requires bespoke design.

An honest tension remains here, in that user-fit and task-fit can pull apart. An interface that adapts strongly to a user's habits may end up reinforcing suboptimal patterns rather than challenging them. Adaptive steering interfaces must be transparent about which fit they optimize for, and let users see the tradeoff when the two conflict.

8.4 Conclusion

> "Now that we can do anything, what will we do?" — Bill Buxton

In this thesis, I present a framework and a set of systems for steering modern AI through interfaces beyond the prompt. AI can now produce images, compose music, write code, and simulate worlds. However, capability alone is not enough. People need interfaces through which they may access that capability, direct it, and understand what is possible.

The four systems each expose a different latent aspect of AI through a direct representation: maps for navigating latent space, puzzle pieces for composing capabilities, sketches for steering structure, and evolution for mixing styles. Across the studies, designers navigated, composed, iterated, and remixed AI capabilities through these representations. They found video connections invisible to timeline search, assembled multimodal pipelines without writing code, sketched while the AI followed their structural intent, and bred style combinations they could not have specified in advance.

In design exploration, the interface representation largely shapes what designers are able to do with AI. Designing an AI tool is therefore as much a problem of representation as of model capability, and the relationship between machine learning and HCI runs in both directions. ML produces mechanisms and HCI designs interfaces that surface them, but the relationship also runs the other way, in that HCI can specify what mechanisms should look like, and ML can build toward those specifications. The technical contributions of this thesis already span both sides: each system required engineering on the AI mechanism alongside interface design. I plan to extend this work in both directions: designing interfaces for newly emerging mechanisms, and shaping mechanisms to fit the interfaces designers need.

Appendix A
Jigsaw: Supplementary Materials

A.1 List of Implemented Models

Table A.1. Full list of implemented models in Jigsaw.

Puzzle piece name	**Model name**	**Input**	**Output**
Ask GPT	GPT-4	Text	Text
Generate image	Stable Diffusion	Text	Image
Generate video	AnimateDiff	Text	Video
Generate 3D model	Shap-E	Text	3D
Generate sound effects	AudioGen	Text	Audio (Sound Effects)
Generate music	MusicGen	Text	Audio (Music)
Generate speech	Bark	Text	Audio (Speech)
Describe image	BLIP	Image	Text
Tag image	Segment Anything	Image	Text
Extract text in image	PyTesseract	Image	Text
Classify emotion from face	ResidualMaskingNetwork	Image	Text
Increase image resolution	Real-ESRGAN	Image	Image
Restore distorted face	GFPGAN	Image	Image
Grayscale to Color	BigColor	Image	Image
Remove image background	Rembg	Image	Image
Remove people	EdgeConnect	Image	Image
Get human pose	OpenPose	Image	Image (Pose)
Get segmentation map	Segment Anything	Image	Image (Segmentation)
Get depth map	MiDaS	Image	Image (Depth)
Get normal map	MiDaS	Image	Image (Normal)
Get edge map	HED	Image	Image (Edge)
Generate 3D model from image	One-2-3-45	Image	3D
Classify video	X-CLIP	Video	Text
Remove video background	Robust Video Matting	Video	Video
Increase video resolution	RealBasicVSR	Video	Video
Increase video frame rate	RIFE	Video	Video
Classify music genre	MusicClassification	Audio	Text
Transcribe speech	Whisper	Audio	Text
Generate image from text and driving image	Instruct-Pix2Pix	Image + Text	Image
Edit face with text	StyleCLIP	Image + Text	Image
Generate image from text and human pose	ControlNet	Image (Pose) + Text	Image
Generate image from text and segmentation map	ControlNet	Image (Segmentation) + Text	Image
Generate image from text and depth map	ControlNet	Image (Depth) + Text	Image
Generate image from text and normal map	ControlNet	Image (Normal) + Text	Image
Generate image from text and edge map	ControlNet	Image (Edge) + Text	Image
Animate a face to talk	SadTalker	Image + Audio	Video
Clone a voice	FreeVC	Audio + Audio	Audio
Generate image from text and sketch	ControlNet	Sketch + Text	Image
Generate artwork from text and sketch	Scribble Stories	Sketch + Text	Image

A.2 Participant Model Mosaics

Table A.2. Model mosaics created by participants during the free creation task.

Participant	**Free creation design**	**Names of puzzle pieces used**	**Number of puzzle pieces used**
P1	Designing a formal dress	Ask GPT (custom, ideation, translation), Tag image, Get human pose, Generate image from text and pose, Remove background, Generate image from text and image	8
P2	Creating an audio-visual story	Ask GPT (ideation, translation), Generate artwork from text and sketch, Generate speech, Clone a voice, Generate sound effects	6
P3	Creating a music video	Ask GPT (custom, ideation, translation), Generate speech, Clone a voice, Classify music genre, Generate music, Generate video, Increase video frame rate	9
P4	Creating multiple movie endings	Ask GPT (custom, ideation, translation), Grayscale to Color, Restore distorted face, Generate image, Generate video, Generate music, Increase video resolution, Increase video frame rate	10
P5	Designing a living room	Ask GPT (ideation, translation), Describe image, Get segmentation map, Get depth map, Generate image from text and segmentation map, Generate image from text and depth map, Remove background, Remove people	9
P6	Designing a tote bag	Ask GPT (ideation, translation), Generate image, Remove background, Generate image from text and image, Get edge map, Generate image from text and edge map, Generate artwork from text and sketch	8
P7	Designing a logo	Ask GPT (custom, ideation, translation), Tag image, Generate image, Generate image from text and sketch, Generate artwork from text and sketch	7
P8	Enhancing a story for the visually impaired	Ask GPT (ideation, translation), Extract text in image, Describe image, Classify emotion from face, Generate speech, Generate sound effects, Generate music	8
P9	Designing a 3D model of a vase	Ask GPT (ideation, translation), Generate 3D model, Generate image, Remove background, Generate 3D model from image	6
P10	Designing a video game character	Ask GPT (ideation, translation), Generate image, Get human pose, Generate image from text and human pose, Remove image background, Generate speech, Animate a face to talk	8

A.3 Semantic Search and Assembly Assistant Examples

Table A.3. Example search queries and returned relevant puzzle pieces using the Catalog Panel's semantic search.

Semantic search query	**Returned relevant puzzle pieces**
Improve content quality	Increase image resolution, Restore distorted face, Increase video resolution, Increase video frame rate
Describe the image	Describe image, Tag image, Extract text in image, Classify emotion from face
Remove unwanted things	Remove image background, Remove people, Remove video background
Restore an old photo	Grayscale to Color, Restore distorted face, Edit face with text
Get the structure of an image	Get depth map, Get edge map, Get normal map, Get segmentation map
Generate an animation	Generate video, Animate a face to talk
Generate spoken words	Generate speech, Clone a voice
Generate content with only text	Ask GPT, Generate image, Generate video, Generate 3D model, Generate sound effects, Generate music, Generate speech
Generate content with an image reference	Generate image from text and driving image, Generate 3D model from image, Edit face with text, Animate a face to talk

Table A.4. Example task queries and recommended combination of puzzle pieces using the Assembly Assistant.

Assembly Assistant task query	**Recommended combination of puzzle pieces**
Design clothes for a model	Get human pose, Ask GPT, Generate image from text and human pose
Create an animated cartoon character	Ask GPT, Generate image, Generate speech, Animate a face to talk
Make a recipe for chocolate chip cookies	Ask GPT, Generate speech
Add sound for an illustration book	Describe image, Ask GPT, Generate sound effects
Create a podcast episode on finance	Ask GPT, Generate speech, Clone a voice
Remodel my home interior	Tag image, Ask GPT, Get depth map, Generate image from text and depth map
Create a short sci-fi film	Ask GPT, Generate video, Generate speech, Generate sound effects
Plan a trip to San Francisco	Ask GPT, Generate speech
Turn my sketch into an artwork	Ask GPT, Generate artwork from text and sketch
Give me gardening advice for my plants	Tag image, Ask GPT

Appendix B

Inkspire: Supplementary Materials

B.1 Participant Usage Logs

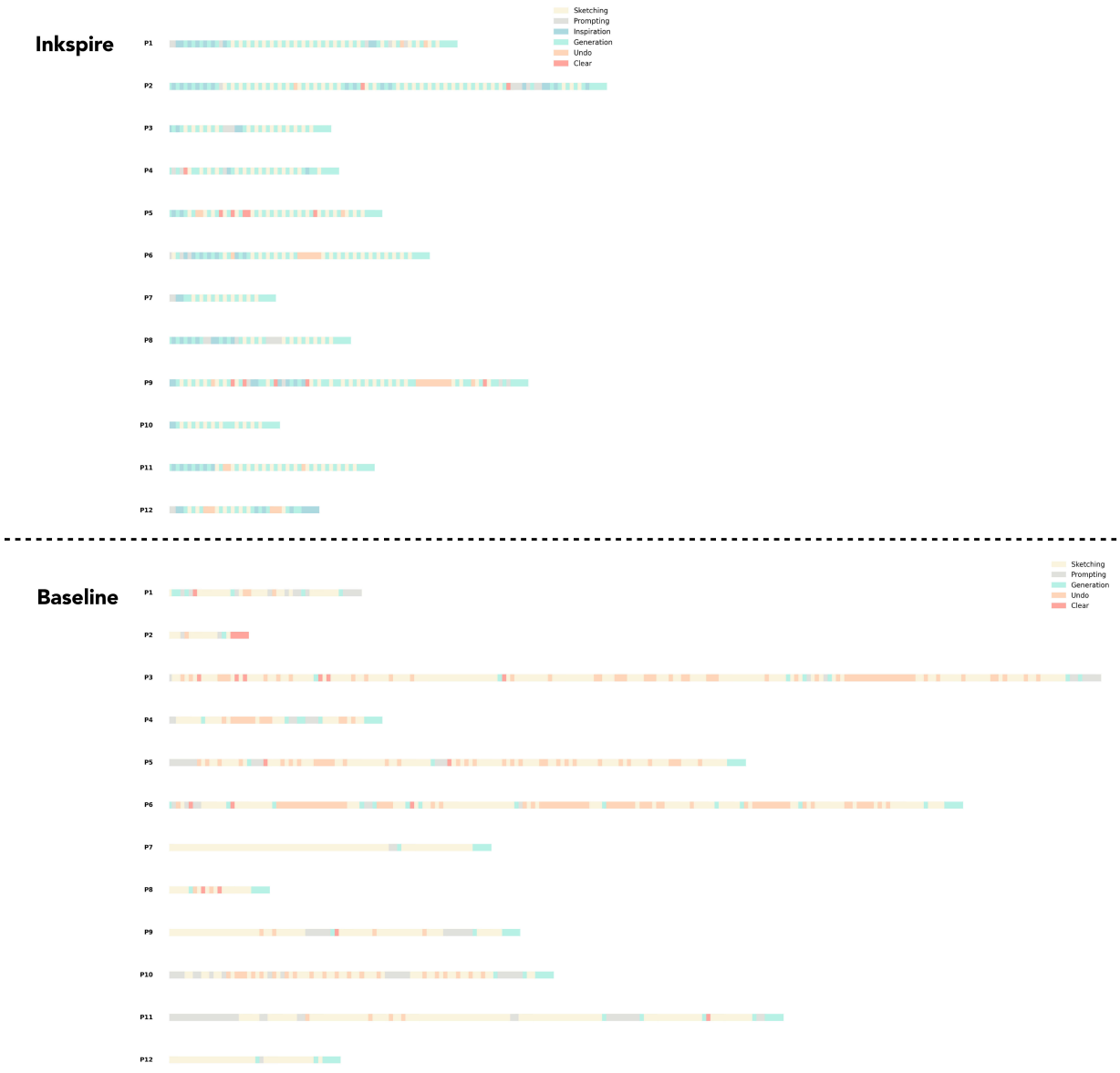


Figure B.1. Complete participant usage logs when using Inkspire vs. the baseline. Please see Figure **6.10** for an annotated illustrative example.

B.2 Baseline Interface

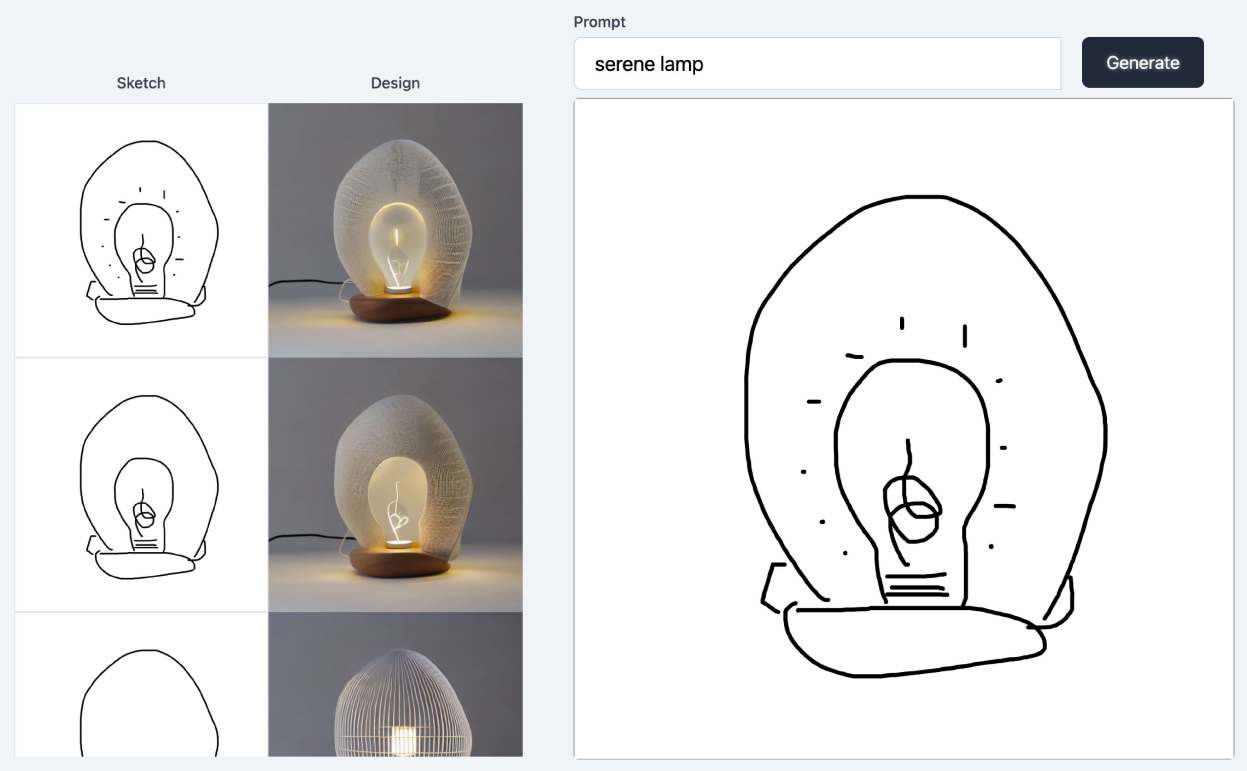


Figure B.2. The baseline interface adopts a similar layout as Inkspire, but without analogical inspirations, scaffolding sketch underlays, and stroke-by-stroke generations. The participant would typically manually specify a prompt and draw a complete sketch for the T2I model – a standard workflow of how designers work with T2I models with additional sketch control [4].

B.3 Analogical Inspirations by Participant

Table B.1. Analogical inspirations explored by participants and statistics on inspiration categories.

ID	Analogies	Category Count				Final
		Total	Nature	Architecture	Fashion	Category
P1	running cheetah, pleated dress, catwalk, skyscraper, spiral staircase, pleated dress, jellyfish	6	2	2	2	Nature
P2	waves, waterfall, silk train, flowing dress, flowing dress, waves, running river, running river, waves, willow tree, infinity pool, concrete, silhouettes of brutalist buildings, rubble	10	4	4	2	Architecture
P3	silk, river, clouds, waterfall	4	3	0	1	Nature
P4	zen garden, moonlight, candle, minimalist church	4	1	2	1	Architecture
P5	bamboo, bonsai, japanese tea house, zen garden	4	2	2	0	Architecture
P6	bay window, japanese tea house, zen garden, lotus flower, zen garden, japanese tea house	4	1	3	0	Architecture
P7	lotus	1	1	0	0	Nature
P8	bamboo grove, moonlight, roman column, snow peak, spring creek, bamboo, zen garden, moonlight	7	5	2	0	Nature
P9	waterfall, pawprint, seashell, sea stone, slinky, ribbon	6	4	0	2	Fashion
P10	flowing gown	1	0	0	1	Fashion
P11	floating bridge, lava, silk scarf, winding staircase	4	1	2	1	Architecture
P12	gaudi's architecture, ribbon, waterfall, running track curve	4	1	2	1	Architecture

Appendix C

GENErator: Supplementary Materials

C.1 Technical Validation

GENErator's genetic operations are only useful if the underlying style space is well-structured and the operations perform as intended. In this section, we validate three properties: that the embedding groups similar styles together, that complement and contrast mutations move in the intended directions, and that natural selection converges reliably toward a target aesthetic.

C.1.1 Style Embedding Quality

All genetic operations depend on the underlying embedding space placing similar styles nearby and dissimilar styles far apart. Without this, complement mutation would draw from unrelated styles and contrast would be indistinguishable from random. A simpler alternative is to encode each style's trigger words directly as an embedding using CLIP [313]. However, trigger words are often ambiguous: "chrome" conflates the material with the color and the web browser, and "glow" and "opalescent" share few linguistic traits despite producing similar luminous visual effects.

To test whether content-independent style embedding improves on this, we identified 10 groups of aesthetically similar genes (e.g., {`Chrome`, `Glossy`, `Mirror`, `Aluminum Foil`}, {`Glow`, `Neon`, `Bioluminescent`, `Opalescent`}) and measured average pairwise cosine similarity within each group. Our embedding achieves $M = 0.66$ ($SD = 0.06$) within-group similarity, compared to $M = 0.36$ ($SD = 0.13$) for text encoding, an 83% improvement in cluster tightness ($t(9) = 7.33$, $p < .01$, $d_s = 2.32$). Our technique produces tighter clusters in 9 of 10 groups. As a result, when complement mutation draws from the nearest 10% of genes, it more reliably returns styles in the same aesthetic family.

C.1.2 Semantic Directionality of Mutations

For complement and contrast mutations to serve different roles, complement targets should land closer to the source than random, and contrast targets should land farther. We verified this by measuring cosine distance from each of the 100 genes to its complement pool (nearest 10%), contrast pool (farthest 10%), and a random selection.

Complement stays closest ($M = 0.31$, $SD = 0.04$), 24% closer than random ($t(99) = 32.2$, $p < .01$, $d_s = 3.22$). Contrast moves farthest ($M = 0.52$, $SD = 0.04$), 27% farther than random ($t(99) = 47.6$, $p < .01$, $d_s = 4.76$). Random falls in the middle ($M = 0.41$, $SD = 0.04$). Overall, contrast targets land 1.7 times farther from the source than complement targets ($p < .01$). For example, `Chrome`'s complement is `Metallic` (similar reflective surface), while its contrast is `Denim` (a completely different texture). This separation lets designers

control how far to push from their current direction, rather than relying only on random mutation [275, 276].

C.1.3 Natural Selection Convergence

Natural selection is only useful if successive generations actually move toward the designer's target rather than drifting. We ran 10 evolution sessions targeting diverse aesthetics ("luxurious," "futuristic," "minimal," "organic," "brutalist," etc.) with a population of 5 and 5 generations each. The fitness score (semantic similarity between the generated image and the target description) increased monotonically from $M = 0.14$ ($SE = 0.01$) at generation 0 to $M = 0.87$ ($SE = 0.02$) at generation 5, with no session showing fitness regression (Figure C.1). This consistency comes from drawing each generation's mutation candidates from genes semantically related to the target description, rather than selecting randomly from the full library. Convergence held across all 10 aesthetics (e.g., from "minimal" to "brutalist"), suggesting the mechanism generalizes across the style space.

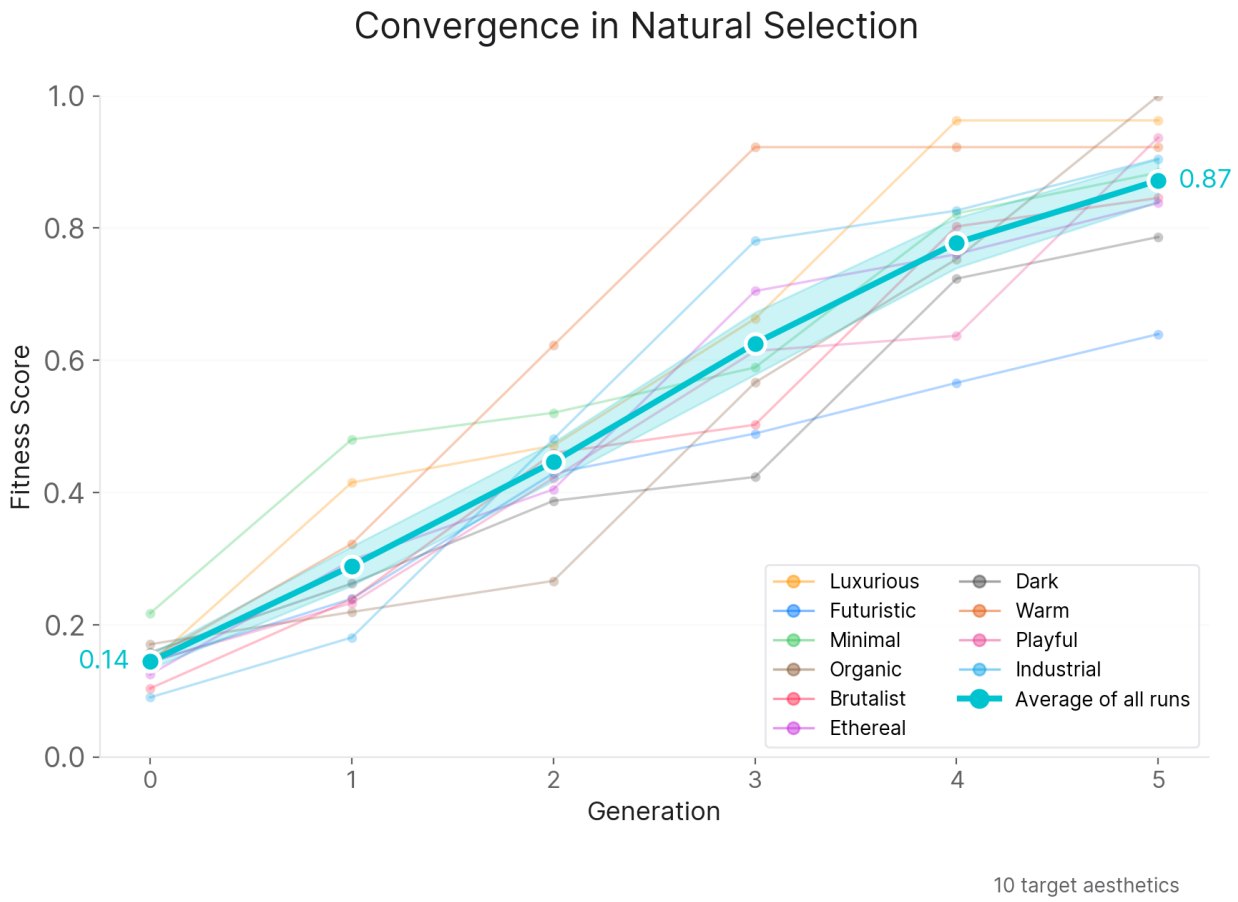


Figure C.1. Natural selection convergence across 10 target aesthetics. Average fitness score increases monotonically from 0.14 to 0.87 over 5 generations, with no session showing regression. The bold teal line shows the average across all runs.

C.2 Additional Figures

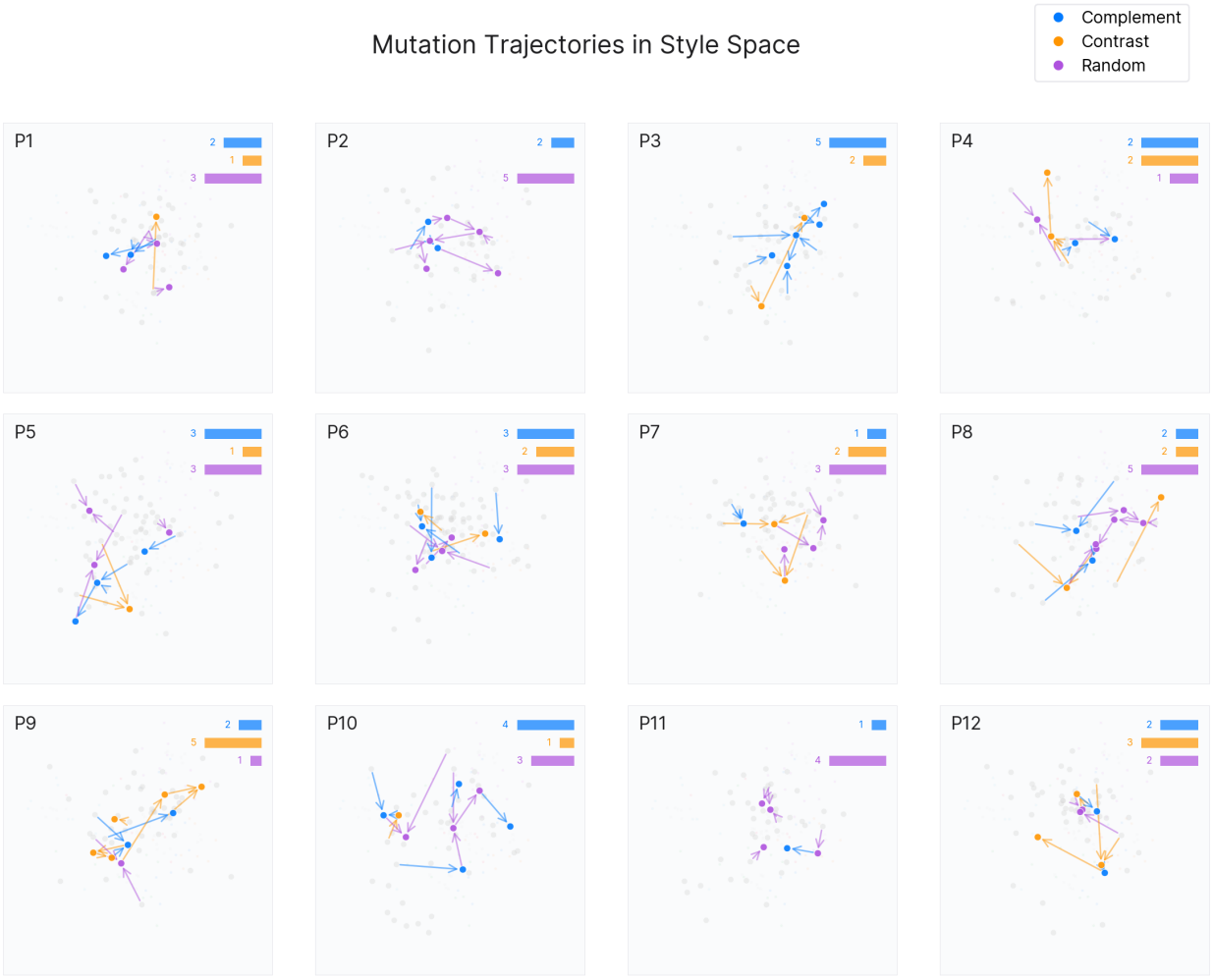


Figure C.2. Mutation trajectories per participant in PCA-projected style space. Each arrow connects a source gene to its mutation offspring. Complement mutations (blue) generally produce shorter arrows that stay near the source and contrast mutations (orange) generally produce longer arrows that jump to distant regions. Stacked bars (top right of each subplot) show each participant's mutation mode distribution, indicating that most participants used all three modes.

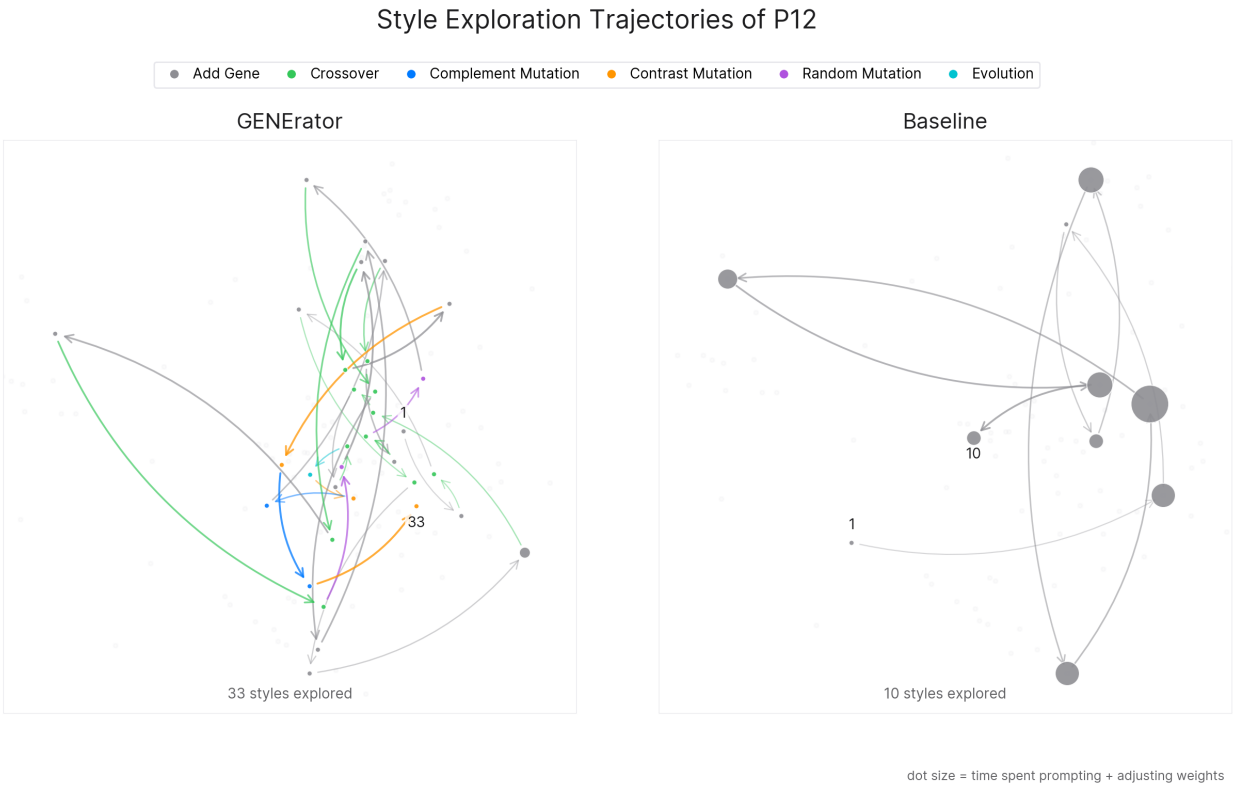


Figure C.3. Style exploration trajectories for P12 in PCA-projected semantic space. Dot size encodes time spent prompting and adjusting weights with the same gene combination. GENErator (left, 33 styles) shows rapid movement across the space through diverse operations (crossover, complement, contrast, evolution), with small dots indicating frequent transitions between styles. Baseline (right, 10 styles) shows fewer, larger dots concentrated in a small region, with repeated loops back to the same gene combinations, a characteristic of the fixation pattern where designers spend more time tuning familiar styles than discovering new ones.

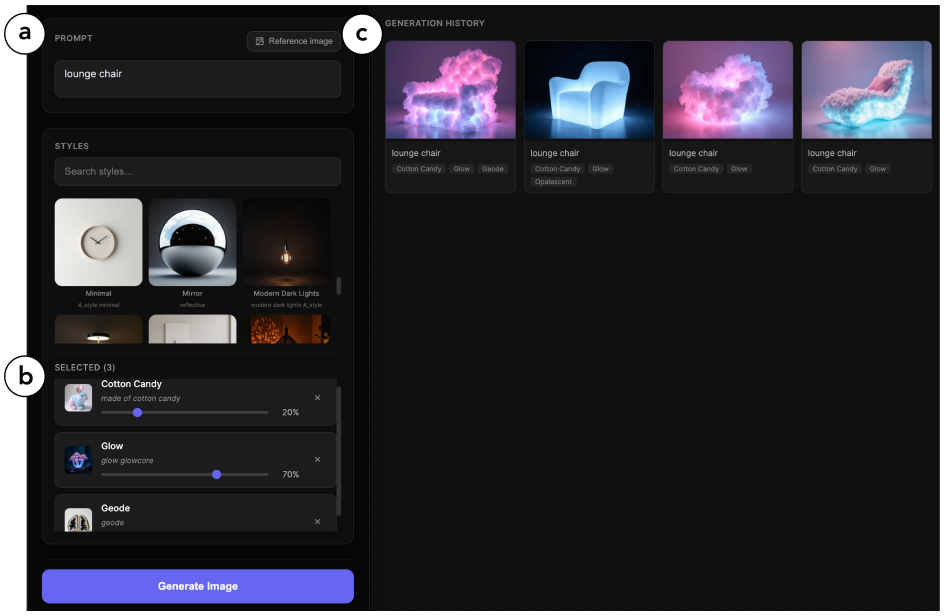


Figure C.4. The baseline interface. Designers select styles from the gene library (a), adjust weight sliders to control each style's influence (b), and generate images. A generation history grid (c) displays all previous outputs. Clicking any previous result restores its selected styles and weights, enabling one-click backtracking. This provides stronger backtracking than most real-world tools, which require designers to manually recreate previous configurations.

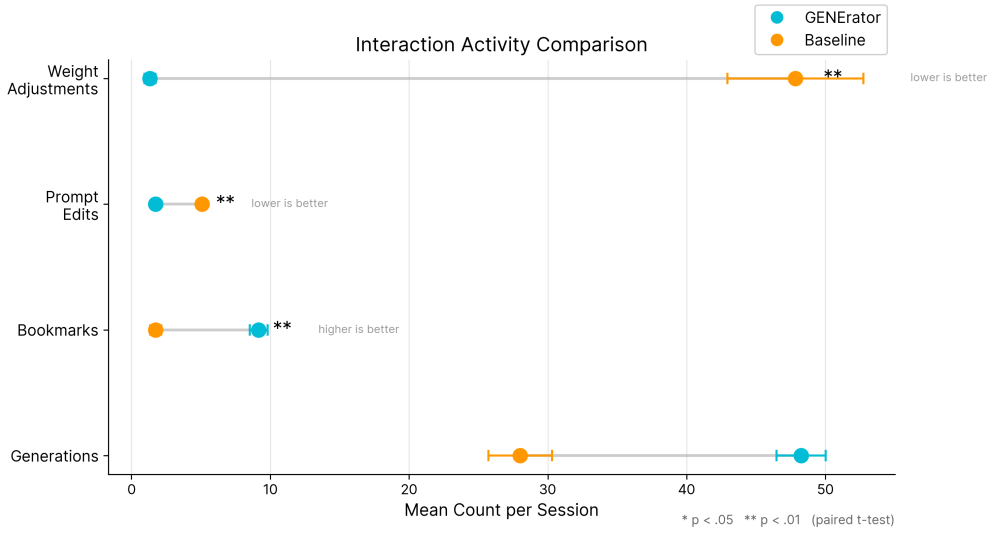


Figure C.5. Mean interaction counts per session for shared operations between GENErator and baseline (weight adjustments, prompt edits, bookmarks, generations). GENErator significantly reduced weight adjustments and prompt edits while increasing bookmarks (designs participants liked). Error bars show standard error.

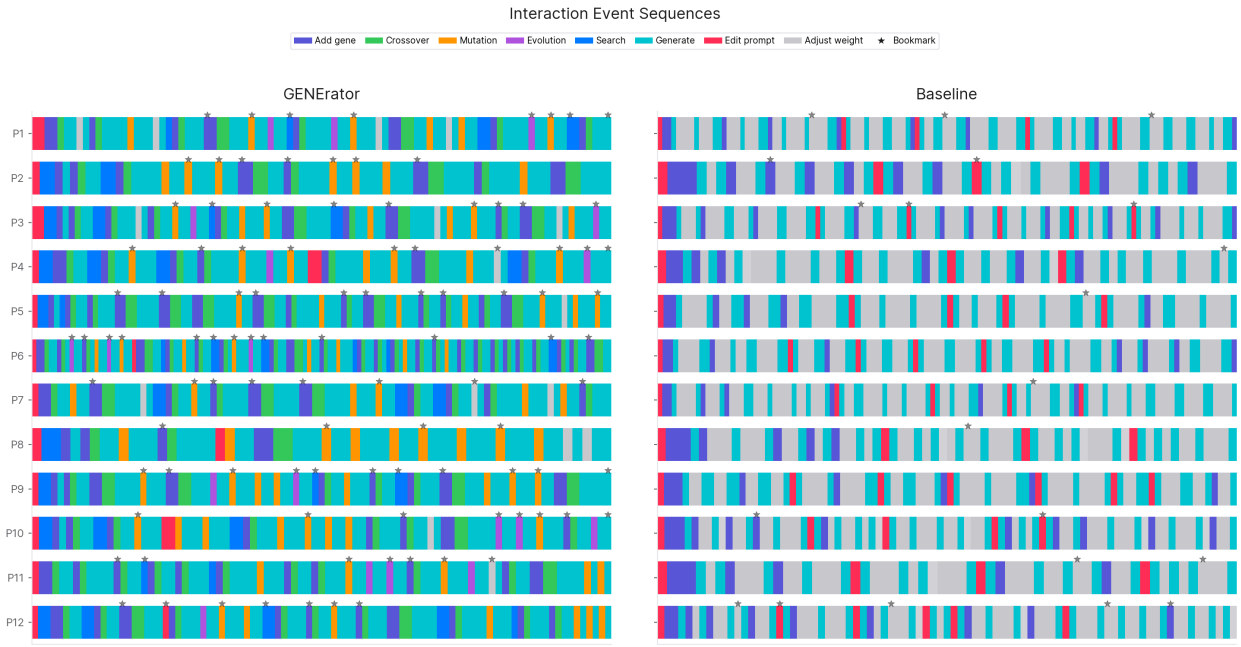


Figure C.6. Interaction event sequences for all 12 participants. Baseline (right) is dominated by trial-and-error weight adjustments and manual prompting. GENErator (left) shows a diverse mix of crossover, mutation, evolution, and semantic search operations across all participants. More designs are bookmarked (favorited) when using GENErator.

Bibliography

[1] Erin Cherry and Celine Latulipe. Quantifying the creativity support of digital tools through the creativity support index. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(4):1–25, 2014.

[2] 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*, pages 264–277, 2023.

[3] Erin Cherry and Celine Latulipe. Quantifying the creativity support of digital tools through the creativity support index. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(4):1–25, 2014.

[4] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

[5] Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.

[6] Bill Buxton. *Sketching user experiences: getting the design right and the right design*. Morgan kaufmann, 2010.

[7] Donald A. Schön. *The Reflective Practitioner: How Professionals Think in Action*. Basic Books, New York, 1983. ISBN 978-0465068784.

[8] Qian Yang. *Profiling Artificial Intelligence as a Material for User Experience Design*. Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, July 2020.

[9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[11] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. Why johnny can't prompt: How non-ai experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21. ACM, 2023. doi: 10.1145/3544548.3581388.

[12] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. Morgan Kaufmann, 2007.

[13] Douglas C. Engelbart. Augmenting human intellect: A conceptual framework. Technical Report AFOSR-3223, Stanford Research Institute, Menlo Park, CA, 1962.

[14] Ivan E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the AFIPS Spring Joint Computer Conference*, pages 329–346, 1963. doi: 10.1145/1461551.1461591.

[15] Ben Shneiderman. Direct manipulation: A step beyond programming languages. *Computer*, 16(8):57–69, 1983. doi: 10.1109/MC.1983.1654471. URL **https://doi.org/10.1109/MC.1983.1654471**.

[16] Edwin L. Hutchins, James D. Hollan, and Donald A. Norman. Direct manipulation interfaces. *Human–Computer Interaction*, 1(4):311–338, December 1985. ISSN 1532-7051. doi: 10.1207/s15327051hci0104_2. URL **http://dx.doi.org/10.1207/s15327051hci0104_2**.

[17] Donald A. Norman. *The Design of Everyday Things*. Basic Books, New York, 1988. ISBN 978-0465067107.

[18] James J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, Boston, 1979. ISBN 978-0898599596.

[19] Hariharan Subramonyam, Colleen Seifert, Sachi Shah, and Eytan Adar. Bridging the gulf of envisioning: Cognitive challenges in prompt-based interactions with LLMs. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–19. ACM, 2024. doi: 10.1145/3613904.3642754.

[20] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343, 1996. doi: 10.1109/VL.1996.545307.

[21] Benjamin B. Bederson and James D. Hollan. Pad++: A zooming graphical interface for exploring alternate interface physics. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*, pages 17–26, 1994.

[22] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. DirectGPT: A direct manipulation interface to interact with large language models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2024. doi: 10.1145/3613904.3642462.

[23] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 159–166. ACM, 1999. doi: 10.1145/302979.303030.

[24] Ben Shneiderman and Pattie Maes. Direct manipulation vs. interface agents. *Interactions*, 4(6):42–61, 1997. doi: 10.1145/267505.267514.

[25] Ben Shneiderman. Human-centered artificial intelligence: Reliable, safe & trustworthy. *International Journal of Human-Computer Interaction*, 36(6):495–504, 2020.

[26] Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13. ACM, 2019. doi: 10.1145/3290605.3300233.

[27] Raja Parasuraman, Thomas B. Sheridan, and Christopher D. Wickens. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, 30(3):286–297, 2000.

[28] John D. Lee and Katrina A. See. Trust in automation: Designing for appropriate reliance. *Human Factors*, 46(1):50–80, 2004.

[29] Margaret A. Boden. *The Creative Mind: Myths and Mechanisms*. Routledge, 2nd edition, 2004.

[30] Ben Shneiderman. Creativity support tools: Accelerating discovery and innovation. *Communications of the ACM*, 50(12):20–32, 2007.

[31] Patricia D. Stokes. *Creativity from Constraints: The Psychology of Breakthrough*. Springer Publishing Company, 2006.

[32] Oguz A. Acar, Murat Tarakci, and Daan van Knippenberg. Creativity and innovation under constraints: A cross-disciplinary integrative review. *Journal of Management*, 45 (1):96–121, 2019. doi: 10.1177/0149206318805832.

[33] Masaki Suwa, John Gero, and Terry Purcell. Unexpected discoveries and S-invention of design requirements: Important vehicles for a design process. *Design Studies*, 21(6): 539–567, 2000.

[34] Allan MacLean, Richard M. Young, Victoria M. E. Bellotti, and Thomas P. Moran. Questions, options, and criteria: Elements of design space analysis. *Human–Computer Interaction*, 6(3–4):201–250, 1991. doi: 10.1080/07370024.1991.9667168.

[35] David G. Jansson and Steven M. Smith. Design fixation. *Design Studies*, 12(1):3–11, 1991.

[36] Gabriela Goldschmidt and Maria Smolkov. Variances in the impact of visual stimuli on design problem solving performance. *Design Studies*, 27(5):549–569, 2006.

[37] Steven P Dow, Alana Glassco, Jonathan Kass, Melissa Schwarz, Daniel L Schwartz, and Scott R Klemmer. Parallel prototyping leads to better design results, more divergence, and increased self-efficacy. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 17(4):1–24, 2010.

[38] 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*, pages 87–96, 2009. doi: 10.1145/1518701.1518717.

[39] 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*, pages 2257–2266, 2010. doi: 10.1145/1753326.1753667.

[40] Hyeonsu B. Kang, David Chuan-En Lin, Yan-Ying Chen, Matthew K. Hong, Nikolas Martelaro, and Aniket Kittur. Biospark: Beyond analogical inspiration to LLM-augmented transfer. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025. doi: 10.1145/3706598.3714053.

[41] Milene Gonçalves, Carlos Cardoso, and Petra Badke-Schaub. Inspiration choices that matter: The selection of external stimuli during ideation. *Design Science*, 2:e10, 2016. doi: 10.1017/dsj.2016.10.

[42] Sangho Suh, Meng Chen, Bryan Min, Toby Jia-Jun Li, and Haijun Xia. Luminate: Structured generation and exploration of design space with large language models for human-AI co-creation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024.

[43] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[44] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents, 2022.

[45] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2I-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4296–4304, 2024. doi: 10.1609/aaai.v38i5.28226.

[46] Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22511–22521, 2023.

[47] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[48] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22500–22510, 2023.

[49] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. In *International Conference on Learning Representations*, 2023.

[50] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. GANSpace: Discovering interpretable GAN controls. In *Advances in Neural Information Processing Systems*, volume 33, pages 9841–9850, 2020.

[51] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of GANs for semantic face editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[52] Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. In *International Conference on Learning Representations*, 2023.

[53] 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*, pages 18392–18402, 2023.

[54] Hai Dang, Karim Benharrak, Florian Lehmann, and Daniel Buschek. Beyond text generation: Supporting writers with continuous automatic text summaries. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13, 2022.

[55] Michael Terry, Chinmay Kulkarni, Martin Wattenberg, Lucas Dixon, and Meredith Ringel Morris. Interactive AI alignment: Specification, process, and evaluation alignment, 2023.

[56] Vivian Liu, Han Qiao, and Lydia Chilton. Opal: Multimodal image generation for news illustration. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pages 1–17, 2022. doi: 10.1145/3526113.3545621.

[57] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. Promptchainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–10, 2022.

[58] Pegah Karimi, Jeba Rezwana, Safat Siddiqui, Mary Lou Maher, and Nasrin Dehbozorgi. Creative sketching partner: An analysis of human-AI co-creativity. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 221–230, 2020. doi: 10.1145/3377325.3377522.

[59] Li-Yuan Chiou, Peng-Kai Hung, Rung-Huei Liang, and Chun-Teng Wang. Designing with ai: An exploration of co-ideation with image generators. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*, pages 1941–1954, 2023.

[60] Yrjö Engeström. *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*. Orienta-Konsultit, 1987.

[61] Michel Beaudouin-Lafon. Instrumental interaction: An interaction model for designing post-WIMP user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 446–453, 2000.

[62] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[63] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. doi: 10.1109/CVPR.2019.00453.

[64] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in GANs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1532–1540, 2021.

[65] Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. Chainforge: A visual toolkit for prompt engineering and LLM hypothesis testing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–18, 2024. doi: 10.1145/3613904.3642016.

[66] Jeffrey Kodosky. Labview. *Proceedings of the ACM on Programming Languages*, 4 (HOPL):1–54, 2020.

[67] OpenJS Foundation. Node-RED: Low-code programming for event-driven applications, 2013. **https://nodered.org/**.

[68] James A. Landay and Brad A. Myers. Sketching interfaces: Toward more human interface design. *IEEE Computer*, 34(3):56–64, 2001.

[69] James Lin, Mark W. Newman, Jason I. Hong, and James A. Landay. DENIM: Finding a tighter fit between tools and practice for web site design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 510–517, 2000.

[70] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '94)*, pages 15–22. ACM, 1994. doi: 10.1145/192161.192167.

[71] Jimmy Secretan, Nicholas Beato, David B. D'Ambrosio, Adelein Rodriguez, Adam Campbell, and Kenneth O. Stanley. Picbreeder: Evolving pictures collaboratively online. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1759–1768, 2008.

[72] Surprising facts on the history of video editing, 2023. URL **https://www.videoeditinginstitute.com/surprising-facts-on-the-history-of-video-editing**.

[73] Working in the project panel, 2023. URL **https://helpx.adobe.com/premiere-pro/using/customizing-project-panel.html**.

[74] Yang Liu, Eunice Jun, Qisheng Li, and Jeffrey Heer. Latent space cartography: Visual analysis of vector space embeddings. In *Computer graphics forum*, volume 38, pages 67–78. Wiley Online Library, 2019.

[75] Kevin Lynch. *Reconsidering the image of the city*. Springer, 1984.

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

[77] Andreas Girgensohn, John Boreczky, Patrick Chiu, John Doherty, Jonathan Foote, Gene Golovchinsky, Shingo Uchihashi, and Lynn Wilcox. A semi-automatic approach to home video editing. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 81–89, 2000.

[78] Renan G Cattelan, Cesar Teixeira, Rudinei Goularte, and Maria Da Graça C Pimentel. Watch-and-comment as a paradigm toward ubiquitous interactive video editing. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 4(4):1–24, 2008.

[79] Diogo Cabral and Nuno Correia. Videoink: a pen-based approach for video editing. In *Adjunct proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 67–68, 2012.

[80] Ruo-Feng Tong, Yun Zhang, and Meng Ding. Video brush: A novel interface for efficient video cutout. In *Computer Graphics Forum*, volume 30, pages 2049–2057. Wiley Online Library, 2011.

[81] Diogo Cabral and Nuno Correia. Video editing with pen-based technology. *Multimedia tools and applications*, 76:6889–6914, 2017.

[82] Tomas Sokoler, Håkan Edeholt, and Martin Johansson. Videotable: a tangible interface for collaborative exploration of video material during design sessions. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, pages 656–657, 2002.

[83] Jamie Zigelbaum, Michael S Horn, Orit Shaer, and Robert JK Jacob. The tangible video editor: collaborative video editing with active tokens. In *Proceedings of the 1st international Conference on Tangible and Embedded interaction*, pages 43–46, 2007.

[84] Allison Merz, Annie Hu, and Tracey Lin. Clipworks: a tangible interface for collaborative video editing. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*, pages 497–500, 2018.

[85] Mackenzie Leake, Abe Davis, Anh Truong, and Maneesh Agrawala. Computational video editing for dialogue-driven scenes. *ACM Trans. Graph.*, 36(4):130–1, 2017.

[86] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. Text-based editing of talking-head video. *ACM Transactions on Graphics (TOG)*, 38 (4):1–14, 2019.

[87] Bernd Huber, Hijung Valentina Shin, Bryan Russell, Oliver Wang, and Gautham J Mysore. B-script: Transcript-based b-roll video editing with recommendations. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2019.

[88] Miao Wang, Guo-Wei Yang, Shi-Min Hu, Shing-Tung Yau, Ariel Shamir, et al. Write-a-video: computational video montage from themed text. *ACM Trans. Graph.*, 38(6): 177–1, 2019.

[89] Mackenzie Leake, Hijung Valentina Shin, Joy O Kim, and Maneesh Agrawala. Generating audio-visual slideshows from text articles using word concreteness. In *CHI*, volume 20, pages 25–30, 2020.

[90] Haijun Xia, Jennifer Jacobs, and Maneesh Agrawala. Crosscast: adding visuals to audio travel podcasts. In *Proceedings of the 33rd annual ACM symposium on user interface software and technology*, pages 735–746, 2020.

[91] Descript — all-in-one video & podcast editing, easy as a doc., 2023. URL **https://www.descript.com**.

[92] Type studio — edit your video by editing text, 2023. URL **https://www.typestudio.co**.

[93] Xian-Sheng Hua, Lie Lu, and Hong-Jiang Zhang. Ave: automated home video editing. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 490–497, 2003.

[94] David Chuan-En Lin, Fabian Caba Heilbron, Joon-Young Lee, Oliver Wang, and Nikolas Martelaro. Videogenic: Identifying highlight moments in videos with professional photographs as a prior. In *Proceedings of the 16th Conference on Creativity and Cognition*, 2024.

[95] Tinghuai Wang, Andrew Mansfield, Rui Hu, and John P Collomosse. An evolutionary approach to automatic video editing. In *2009 Conference for Visual Media Production*, pages 127–134. IEEE, 2009.

[96] Alejandro Pardo, Fabian Caba, Juan León Alcázar, Ali K Thabet, and Bernard Ghanem. Learning to cut by watching movies. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6858–6868, 2021.

[97] Yuzhong Huang, Xue Bai, Oliver Wang, Fabian Caba, and Aseem Agarwala. Learning where to cut from edited videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3215–3223, 2021.

[98] Chong Huang, Chuan-En Lin, Zhenyu Yang, Yan Kong, Peng Chen, Xin Yang, and Kwang-Ting Cheng. Learning to film from professional human motion videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4244–4253, 2019.

[99] Boris Chen, Amir Ziai, Rebecca S Tucker, and Yuchen Xie. Match cutting: Finding cuts with smooth visual transitions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2115–2125, 2023.

[100] Bryan Wang, Yuliang Li, Zhaoyang Lv, Haijun Xia, Yan Xu, and Raj Sodhi. Lave: Llm-powered agent assistance and language augmentation for video editing. *arXiv preprint arXiv:2402.10294*, 2024.

[101] David Chuan-En Lin, Anastasis Germanidis, Cristóbal Valenzuela, Yining Shi, and Nikolas Martelaro. Soundify: Matching sound effects to video. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13, 2023.

[102] Peggy Chi, Zheng Sun, Katrina Panovich, and Irfan Essa. Automatic video creation from a web page. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pages 279–292, 2020.

[103] Peggy Chi, Nathan Frey, Katrina Panovich, and Irfan Essa. Automatic instructional video creation from a markdown-formatted tutorial. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 677–690, 2021.

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

[105] Edwin L Hutchins, James D Hollan, and Donald A Norman. Direct manipulation interfaces. *Human–computer interaction*, 1(4):311–338, 1985.

[106] Don Kimber, Tony Dunnigan, Andreas Girgensohn, Frank Shipman, Thea Turner, and Tao Yang. Trailblazing: Video playback control by direct object manipulation. In *2007 IEEE International Conference on Multimedia and Expo*, pages 1015–1018. IEEE, 2007.

[107] Thorsten Karrer, Malte Weiss, Eric Lee, and Jan Borchers. Dragon: a direct manipulation interface for frame-accurate in-scene video navigation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 247–250, 2008.

[108] Cuong Nguyen, Yuzhen Niu, and Feng Liu. Direct manipulation video navigation in 3d. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1169–1172, 2013.

[109] Dan B Goldman, Chris Gonterman, Brian Curless, David Salesin, and Steven M Seitz. Video object annotation, navigation, and composition. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 3–12, 2008.

[110] Suporn Pongnumkul, Jue Wang, Gonzalo Ramos, and Michael Cohen. Content-aware dynamic timeline for video browsing. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 139–142, 2010.

[111] Justin Matejka, Tovi Grossman, and George Fitzmaurice. Swifter: improved online video scrubbing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1159–1168, 2013.

[112] Dan Jackson, James Nicholson, Gerrit Stoeckigt, Rebecca Wrobel, Anja Thieme, and Patrick Olivier. Panopticon: A parallel video overview system. In *proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 123–130, 2013.

[113] Yoshinobu Tonomura, Akihito Akutsu, Kiyotaka Otsuji, and Toru Sadakata. Videomap and videospaceicon: Tools for anatomizing video content. In *Proceedings of the INTERACT'93 and CHI'93 Conference on Human Factors in Computing Systems*, pages 131–136, 1993.

[114] Saelyne Yang, Jisu Yim, Aitolkyn Baigutanova, Seoyoung Kim, Minsuk Chang, and Juho Kim. Softvideo: Improving the learning experience of software tutorial videos with collective interaction data. In *27th International Conference on Intelligent User Interfaces*, pages 646–660, 2022.

[115] Jeongyeon Kim, Daeun Choi, Nicole Lee, Matt Beane, and Juho Kim. Surch: Enabling structural search and comparison for surgical videos. *arXiv preprint*, 2023.

[116] Amy Pavel, Colorado Reed, Björn Hartmann, and Maneesh Agrawala. Video digests: a browsable, skimmable format for informational lecture videos. In *UIST*, volume 10, pages 2642918–2647400. Citeseer, 2014.

[117] Haojin Yang and Christoph Meinel. Content based lecture video retrieval using speech and video text information. *IEEE transactions on learning technologies*, 7(2):142–154, 2014.

[118] Amy Pavel, Dan B Goldman, Björn Hartmann, and Maneesh Agrawala. Sceneskim: Searching and browsing movies using synchronized captions, scripts and plot summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pages 181–190, 2015.

[119] Yi-Hao Peng, Jeffrey P Bigham, and Amy Pavel. Slidecho: Flexible non-visual exploration of presentation videos. In *Proceedings of the 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–12, 2021.

[120] Minsuk Chang, Mina Huh, and Juho Kim. Rubyslippers: Supporting content-based voice navigation for how-to videos. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2021.

[121] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[122] Nicola Pezzotti, Thomas Höllt, B Lelieveldt, Elmar Eisemann, and Anna Vilanova. Hierarchical stochastic neighbor embedding. In *Computer Graphics Forum*, volume 35, pages 21–30. Wiley Online Library, 2016.

[123] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[124] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[125] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B Viégas, and Martin Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *arXiv preprint arXiv:1611.05469*, 2016.

[126] Klaus Eckelt, Andreas Hinterreiter, Patrick Adelberger, Conny Walchshofer, Vaishali Dhanoa, Christina Humer, Moritz Heckmann, Christian Steinparz, and Marc Streit. Visual exploration of relationships and structure in low-dimensional embeddings. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[127] Shusen Liu, Peer-Timo Bremer, Jayaraman J Thiagarajan, Vivek Srikumar, Bei Wang, Yarden Livnat, and Valerio Pascucci. Visual exploration of semantic relationships in neural word embeddings. *IEEE transactions on visualization and computer graphics*, 24(1):553–562, 2017.

[128] Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the eleventh acm international conference on web search and data mining*, pages 673–681, 2018.

[129] Minsuk Chang, Léonore V Guillain, Hyeungshik Jung, Vivian M Hare, Juho Kim, and Maneesh Agrawala. Recipescape: An interactive tool for analyzing cooking instructions at scale. In *Proceedings of the 2018 CHI conference on human factors in computing systems*, pages 1–12, 2018.

[130] Hai Dang and Daniel Buschek. Gesturemap: Supporting visual analytics and quantitative analysis of motion elicitation data by learning 2d embeddings. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2021.

[131] Nicolas Grossmann, Eduard Gröller, and Manuela Waldner. Concept splatters: Exploration of latent spaces based on human interpretable concepts. *Computers & Graphics*, 105:73–84, 2022.

[132] Venkatesh Sivaraman, Yiwei Wu, and Adam Perer. Emblaze: Illuminating machine learning representations through interactive comparison of embedding spaces. In *27th International Conference on Intelligent User Interfaces*, pages 418–432, 2022.

[133] Quan Li, Kristanto Sean Njotoprawiro, Hammad Haleem, Qiaoan Chen, Chris Yi, and Xiaojuan Ma. Embeddingvis: A visual analytics approach to comparative network embedding inspection. In *2018 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 48–59. IEEE, 2018.

[134] Angie Boggust, Brandon Carter, and Arvind Satyanarayan. Embedding comparator: Visualizing differences in global structure and local neighborhoods via small multiples. In *27th international conference on intelligent user interfaces*, pages 746–766, 2022.

[135] Kevin Wang, Deva Ramanan, and Aayush Bansal. Video-specific autoencoders for exploring, editing and transmitting videos. *arXiv preprint arXiv:2103.17261*, 2021.

[136] Brock Craft and Paul Cairns. Beyond guidelines: what can we learn from the visual information seeking mantra? In *Ninth International Conference on Information Visualisation (IV'05)*, pages 110–118. IEEE, 2005.

[137] Marius Hogräfer, Magnus Heitzler, and Hans-Jörg Schulz. The state of the art in map-like visualization. In *Computer Graphics Forum*, volume 39, pages 647–674. Wiley Online Library, 2020.

[138] Karel Reisz and Gavin Millar. *The technique of film editing*. Focal Press, 1971.

[139] Match cuts and creative transitions with examples — editing techniques, 2023. URL **https://www.studiobinder.com/blog/match-cuts-creative-transitions-examples**.

[140] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[141] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[142] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*, pages 364–371. Elsevier, 2003.

[143] Get directions and show routes, 2023. URL **https://support.google.com/maps/answer/144339**.

[144] Clayton Lewis. *Using the" thinking-aloud" method in cognitive interface design*. IBM TJ Watson Research Center Yorktown Heights, NY, 1982.

[145] Upwork, 2022. URL **https://www.upwork.com/**.

[146] Robert Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

[147] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, volume 2, pages 674–679, 1981.

[148] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[149] Reflections on foundation models, 2023. URL **https://hai.stanford.edu/news/reflections-foundation-models**.

[150] David Chuan-En Lin and Nikolas Martelaro. Jigsaw: Supporting designers to prototype multimodal applications by assembling ai foundation models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024.

[151] Nur Yildirim, Changhoon Oh, Deniz Sayar, Kayla Brand, Supritha Challa, Violet Turri, Nina Crosby Walton, Anna Elise Wong, Jodi Forlizzi, James McCann, and John Zimmerman. Creating design resources to scaffold the ideation of ai concepts. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*, DIS '23, page 2326–2346, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450398930. doi: 10.1145/3563657.3596058. URL **https://doi.org/10.1145/3563657.3596058**.

[152] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191.

[153] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[154] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[155] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[156] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023.

[157] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

[158] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. Audiogen: Textually guided audio generation. *arXiv preprint arXiv:2209.15352*, 2022.

[159] Chatgpt, 2023. URL **https://chat.openai.com/**.

[160] Midjourney, 2023. URL **https://www.midjourney.com/**.

[161] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[162] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *European Conference on Computer Vision*, pages 350–368. Springer, 2022.

[163] Brad A Myers. Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing*, 1(1):97–123, 1990.

[164] Kirsten N Whitley and Alan F Blackwell. Visual programming: the outlook from academia and industry. In *Papers presented at the seventh workshop on Empirical studies of programmers*, pages 180–208, 1997.

[165] Philip T Cox, FR Giles, and Tomasz Pietrzykowski. Prograph: a step towards liberating programming from textual conditioning. In *1989 IEEE Workshop on Visual languages*, pages 150–151. IEEE Computer Society, 1989.

[166] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[167] Neil Fraser et al. Blockly: A visual programming editor. *URL: https://code. google. com/p/blockly*, 42, 2013.

[168] Unity visual scripting, 2023. URL **https://unity.com/features/unity-visual-scripting**.

[169] Rhino - grasshopper - new in rhino 6, 2023. URL **https://www.rhino3d.com/6/new/grasshopper/**.

[170] What is max? — cycling '74, 2023. URL **https://cycling74.com/products/max**.

[171] Michelle Carney, Barron Webster, Irene Alvarado, Kyle Phillips, Noura Howell, Jordan Griffith, Jonas Jongejan, Amit Pitaru, and Alexander Chen. Teachable machine: Approachable web-based tool for exploring machine learning classification. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pages 1–8, 2020.

[172] Randi Williams, Michał Moskal, and Peli De Halleux. Ml blocks: A block-based, graphical user interface for creating tinyml models. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 1–5. IEEE, 2022.

[173] Flowiseai. **https://github.com/FlowiseAI/Flowise**, 2013.

[174] Langflow. **https://github.com/logspace-ai/langflow**, 2013.

[175] Ruofei Du, Na Li, Jing Jin, Michelle Carney, Scott Miles, Maria Kleiner, Xiuxiu Yuan, Yinda Zhang, Anuva Kulkarni, Xingyu Liu, et al. Rapsai: Accelerating machine learning prototyping of multimedia applications through visual programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–23, 2023.

[176] Comfyui. **https://github.com/comfyanonymous/ComfyUI**, 2013.

[177] Ai magic tools - runway, 2023. URL **https://runwayml.com/ai-magic-tools/**.

[178] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

[179] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. Research through design as a method for interaction design research in hci. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 493–502, 2007.

[180] Yang Shi, Tian Gao, Xiaohan Jiao, and Nan Cao. Understanding design collaboration between designers and artificial intelligence: A systematic literature review. *Proceedings of the ACM on Human-Computer Interaction*, 7(CSCW2):1–35, 2023.

[181] Qian Yang. Machine learning as a ux design material: how can we imagine beyond automation, recommenders, and reminders? In *AAAI Spring Symposia*, volume 1, pages 2–6, 2018.

[182] Hariharan Subramonyam, Colleen Seifert, and Eytan Adar. Towards a process model for co-creating ai experiences. In *Designing Interactive Systems Conference 2021*, pages 1529–1543, 2021.

[183] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. Re-examining whether, why, and how human-ai interaction is uniquely difficult to design. In *Proceedings of the 2020 chi conference on human factors in computing systems*, pages 1–13, 2020.

[184] Frederic Gmeiner, Humphrey Yang, Lining Yao, Kenneth Holstein, and Nikolas Martelaro. Exploring challenges and opportunities to support designers in learning to co-create with ai-based manufacturing design tools. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–20, 2023.

[185] Vivian Liu and Lydia B Chilton. Design guidelines for prompt engineering text-to-image generative models. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–23, 2022.

[186] JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21, 2023.

[187] Qian Yang, Alex Scuito, John Zimmerman, Jodi Forlizzi, and Aaron Steinfeld. Investigating how experienced ux designers effectively work with machine learning. In *Proceedings of the 2018 designing interactive systems conference*, pages 585–596, 2018.

[188] Nur Yildirim, Alex Kass, Teresa Tung, Connor Upton, Donnacha Costello, Robert Giusti, Sinem Lacin, Sara Lovic, James M O'Neill, Rudi O'Reilly Meehan, et al. 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*, pages 1–13, 2022.

[189] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[190] Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022.

[191] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

[192] David Weintrop and Uri Wilensky. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children*, pages 199–208, 2015.

[193] Gábor Karsai. A configurable visual programming environment: A tool for domain-specific programming. *Computer*, 28(3):36–44, 1995.

[194] Eldon Schoop, Forrest Huang, and Bjoern Hartmann. Umlaut: Debugging deep learning programs using program structure and model behavior. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–16, 2021.

[195] Peiling Jiang, Fuling Sun, and Haijun Xia. Log-it: Supporting programming with interactive, contextual, structured, and visual logs. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2023.

[196] Caneel K Joyce. *The blank page: Effects of constraint on creativity*. University of California, Berkeley, 2009.

[197] Gpt-4v(ision) system card, 2023. URL **https://cdn.openai.com/papers/GPTV_System_Card.pdf**.

[198] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[199] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*, 2023.

[200] Rongjie Huang, Mingze Li, Dongchao Yang, Jiatong Shi, Xuankai Chang, Zhenhui Ye, Yuning Wu, Zhiqing Hong, Jiawei Huang, Jinglin Liu, et al. Audiogpt: Understanding and generating speech, music, sound, and talking head. *arXiv preprint arXiv:2304.12995*, 2023.

[201] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

[202] A. de Rooij and M. Mose Biskjaer. Expecting the unexpected: A review of surprise in design processes. In C. Gray, E. Ciliotta Chehade, P. Hekkert, L. Forlano, P. Ciuccarelli, and P. Lloyd, editors, *DRS2024: Boston, 23–28 June*, Boston, USA, 2024. doi: 10.21606/drs.2024.333.

[203] David G Jansson and Steven M Smith. Design fixation. *Design studies*, 12(1):3–11, 1991.

[204] Samangi Wadinambiarachchi, Ryan M. Kelly, Saumya Pareek, Qiushi Zhou, and Eduardo Velloso. The effects of generative ai on design fixation and divergent thinking. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3642919. URL **https://doi.org/10.1145/3613904.3642919**.

[205] Minsuk Chang, Stefania Druga, Alexander J Fiannaca, Pedro Vergani, Chinmay Kulkarni, Carrie J Cai, and Michael Terry. The prompt artists. In *Proceedings of the 15th Conference on Creativity and Cognition*, pages 75–87, 2023.

[206] Ai is plundering the imagination and replacing it with a slot machine, 2024. URL **https://thebulletin.org/2022/10/ai-is-plundering-the-imagination-and-replacing-it-with-a-slot-machine/**.

[207] Ashok K Goel. Design, analogy, and creativity. *IEEE expert*, 12(3):62–70, 1997.

[208] J. S. Linsey, A. B. Markman, and K. L. Wood. Design by Analogy: A Study of the WordTree Method for Problem Re-Representation. *Journal of Mechanical Design*, 134 (4):041009, 04 2012. ISSN 1050-0472. doi: 10.1115/1.4006145. URL **https://doi.org/10.1115/1.4006145**.

[209] Shuo Jiang, Jie Hu, Kristin L Wood, and Jianxi Luo. Data-driven design-by-analogy: state-of-the-art and future directions. *Journal of Mechanical Design*, 144(2):020801, 2022.

[210] Stephen Brade, Bryan Wang, Mauricio Sousa, Sageev Oore, and Tovi Grossman. Promptify: Text-to-image generation through interactive prompt exploration with large language models. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–14, 2023.

[211] Kihoon Son, DaEun Choi, Tae Soo Kim, Young-Ho Kim, and Juho Kim. Genquery: Supporting expressive visual search with generative models. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2024.

[212] Richard Lee Davis, Thiemo Wambsganss, Wei Jiang, Kevin Gonyop Kim, Tanja Käser, and Pierre Dillenbourg. Fashioning creative expertise with generative ai: Graphical interfaces for gan-based design space exploration beter support ideation than text prompts for difusion models. In *Conference on Human Factors in Computing Systems-Proceedings*, 2024.

[213] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*, pages 1–10, 2022.

[214] 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*, pages 18392–18402, 2023.

[215] Raluca Budiu. Memory recognition and recall in user interfaces. *Nielsen Norman Group*, 1, 2014.

[216] Yong Jae Lee, C Lawrence Zitnick, and Michael F Cohen. Shadowdraw: real-time user guidance for freehand drawing. *ACM Transactions on Graphics (ToG)*, 30(4):1–10, 2011.

[217] Zhiyu Lin, Upol Ehsan, Rohan Agarwal, Samihan Dani, Vidushi Vashishth, and Mark Riedl. Beyond prompts: Exploring the design space of mixed-initiative co-creativity systems. *arXiv preprint arXiv:2305.07465*, 2023.

[218] Vivian Liu, Tao Long, Nathan Raw, and Lydia Chilton. Generative disco: Text-to-video generation for music visualization. *arXiv preprint arXiv:2304.08551*, 2023.

[219] Yihan Hou, Manling Yang, Hao Cui, Lei Wang, Jie Xu, and Wei Zeng. C2ideas: Supporting creative interior color design ideation with large language model. *arXiv preprint arXiv:2401.12586*, 2024.

[220] David Chuan-En Lin and Nikolas Martelaro. Jigsaw: Supporting designers to prototype multimodal applications by assembling ai foundation models. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024.

[221] Alice Cai, Steven R Rick, Jennifer L Heyman, Yanxia Zhang, Alexandre Filipowicz, Matthew Hong, Matt Klenk, and Thomas Malone. Designaid: Using generative ai and semantic diversity for design inspiration. In *Proceedings of The ACM Collective Intelligence Conference*, CI '23, page 1–11, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701139. doi: 10.1145/3582269.3615596. URL **https://doi.org/10.1145/3582269.3615596**.

[222] Antoine Bordas, Pascal Le Masson, and Benoit Weil. Switching perspectives on generative artificial intelligence: a design view for humans-generative ai co-creativity. In *R&D Management Conference 2024*, 2024.

[223] Chengzhi Zhang, Weijie Wang, Paul Pangaro, Nikolas Martelaro, and Daragh Byrne. Generative image ai using design sketches as input: Opportunities and challenges. In *Proceedings of the 15th Conference on Creativity and Cognition*, C&C '23, page 254–261, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701801. doi: 10.1145/3591196.3596820. URL **https://doi.org/10.1145/3591196.3596820**.

[224] Carlos Cardoso, Petra Badke-Schaub, and Ana Luz. Design fixation on non-verbal stimuli: The influence of simple vs. rich pictorial information on design problem-solving. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 49057, pages 995–1002, 2009.

[225] Peiyao Cheng, Ruth Mugge, and Jan PL Schoormans. A new strategy to reduce design fixation: Presenting partial photographs to designers. *Design Studies*, 35(4):374–391, 2014.

[226] Ashok K Goel, Swaroop Vattam, Bryan Wiltgen, and Michael Helms. Cognitive, collaborative, conceptual and creative—four characteristics of the next generation of knowledge-based cad systems: A study in biologically inspired design. *Computer-Aided Design*, 44(10):879–900, 2012.

[227] L Siddharth and Amaresh Chakrabarti. Evaluating the impact of idea-inspire 4.0 on analogical transfer of concepts. *Ai Edam*, 32(4):431–448, 2018.

[228] Julian FV Vincent and Darrell L Mann. Systematic technology transfer from biology to engineering. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 360(1791):159–173, 2002.

[229] E Kwon, A Pehlken, K-D Thoben, A Bazylak, and LH Shu. Visual similarity to aid alternative-use concept generation for retired wind-turbine blades. *Journal of Mechanical Design*, 141(3):031106, 2019.

[230] Zijian Zhang and Yan Jin. An unsupervised deep learning model to discover visual similarity between sketches for visual analogy support. In *International design engineering technical conferences and computers and information in engineering conference*, volume 83976. American Society of Mechanical Engineers, 2020.

[231] Shuo Jiang, Jianxi Luo, Guillermo Ruiz-Pava, Jie Hu, and Christopher L Magee. Deriving design feature vectors for patent images using convolutional neural networks. *Journal of Mechanical Design*, 143(6):061405, 2021.

[232] Mohammed Khaliq, Diego Frassinelli, and Sabine Schulte im Walde. Comparison of image generation models for abstract and concrete event descriptions. In *Proceedings of the 4th Workshop on Figurative Language Processing (FigLang 2024)*, pages 15–21, 2024.

[233] Jiayi Liao, Xu Chen, Qiang Fu, Lun Du, Xiangnan He, Xiang Wang, Shi Han, and Dongmei Zhang. Text-to-image generation for abstract concepts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 3360–3368, 2024.

[234] Zezhong Fan, Xiaohan Li, Kaushiki Nag, Chenhao Fang, Topojoy Biswas, Jianpeng Xu, and Kannan Achan. Prompt optimizer of text-to-image diffusion models for abstract concept understanding. In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 1530–1537, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701726. doi: 10.1145/3589335.3651927. URL **https://doi.org/10.1145/3589335.3651927**.

[235] Shengqiong Wu, Hao Fei, Hanwang Zhang, and Tat-Seng Chua. Imagine that! abstract-to-intricate text-to-image synthesis with scene graph hallucination diffusion. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

[236] Emmanuel Iarussi, Adrien Bousseau, and Theophanis Tsandilas. The drawing assistant: Automated drawing guidance and feedback from photographs. In *ACM Symposium on User Interface Software and Technology (UIST)*. ACM, 2013.

[237] Luca Benedetti, Holger Winnemöller, Massimiliano Corsini, and Roberto Scopigno. Painting with bob: assisted creativity for novices. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 419–428, 2014.

[238] Alex Limpaecher, Nicolas Feltman, Adrien Treuille, and Michael Cohen. Real-time drawing assistance through crowdsourcing. *ACM Transactions on Graphics (TOG)*, 32(4):1–8, 2013.

[239] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016.

[240] Zhengyu Huang, Yichen Peng, Tomohiro Hibino, Chunqi Zhao, Haoran Xie, Tsukasa Fukusato, and Kazunori Miyata. dualface: Two-stage drawing guidance for freehand portrait sketching. *Computational Visual Media*, 8:63–77, 2022.

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

[242] Nicholas Davis, Safat Siddiqui, Pegah Karimi, Mary Lou Maher, and Kazjon Grace. Creative sketching partner: A co-creative sketching tool to inspire design creativity. In *ICCC*, pages 358–359, 2019.

[243] Blake Williford, Samantha Ray, Jung In Koh, Josh Cherian, Paul Taele, and Tracy Hammond. Exploring creativity support for concept art ideation. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI EA '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394222. doi: 10.1145/3544549.3585684. URL **https://doi.org/10.1145/3544549.3585684**.

[244] How To Draw Cars. Pro designer teaches you how to create variations on your car design themes. **https://youtu.be/GxFloTllsKE?si=yWYl9w_etZVZPbnK&t=80**, 2017. Accessed: September 12, 2024.

[245] Vizcom, 2024. URL **https://app.vizcom.ai/**.

[246] Yutong Xie, Zhaoying Pan, Jinge Ma, Luo Jie, and Qiaozhu Mei. A prompt log analysis of text-to-image generation systems. In *Proceedings of the ACM Web Conference 2023*, pages 3892–3902, 2023.

[247] Jon-Michael Deldin and Megan Schuknecht. The asknature database: enabling solutions in biomimetic design. In *Biologically inspired design: Computational methods and tools*, pages 17–27. Springer, 2013.

[248] Hyeonsu B Kang, David Chuan-En Lin, Nikolas Martelaro, Aniket Kittur, Yan-Ying Chen, and Matthew K Hong. Biospark: An end-to-end generative system for biological-analogical inspirations and ideation. *arXiv preprint arXiv:2312.11388*, 2023.

[249] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[250] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[251] Dedre Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive science*, 7(2):155–170, 1983.

[252] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern recognition*, 106:107404, 2020.

[253] How to learn to draw by tracing, 2024. URL **https://monikazagrobelna.com/2020/08/16/how-to-learn-to-draw-by-tracing/**.

[254] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[255] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[256] Chengze Li, Xueting Liu, and Tien-Tsin Wong. Deep extraction of manga structural lines. *ACM Transactions on Graphics (TOG)*, 36(4):1–12, 2017.

[257] Hal R Arkes and Catherine Blumer. The psychology of sunk cost. *Organizational behavior and human decision processes*, 35(1):124–140, 1985.

[258] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

[259] Ken Kocienda. *Creative selection: Inside Apple's design process during the golden age of Steve Jobs*. Pan Macmillan, 2018.

[260] Seung Won Lee, Tae Hee Jo, Semin Jin, Jiin Choi, Kyungwon Yun, Sergio Bromberg, Seonghoon Ban, and Kyung Hoon Hyun. The impact of sketch-guided vs. prompt-guided 3d generative ais on the design exploration process. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '24, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703300. doi: 10.1145/3613904.3642218. URL **https://doi.org/10.1145/3613904.3642218**.

[261] Barrett R Anderson, Jash Hemant Shah, and Max Kreminski. Homogenization effects of large language models on human creative ideation. In *Proceedings of the 16th Conference on Creativity & Cognition*, C&C '24, page 413–425, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704857. doi: 10.1145/3635636.3656204. URL **https://doi.org/10.1145/3635636.3656204**.

[262] Vishnu Sarukkai, Lu Yuan, Mia Tang, Maneesh Agrawala, and Kayvon Fatahalian. Block and detail: Scaffolding sketch-to-image generation. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13, 2024.

[263] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[264] Kaiwen Zhang, Yifan Zhou, Xudong Xu, Bo Dai, and Xingang Pan. Diffmorpher: Unleashing the capability of diffusion models for image morphing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7912–7921, 2024.

[265] Ziheng Ouyang, Zhen Li, and Qibin Hou. K-lora: Unlocking training-free fusion of any subject and style loras. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13041–13050. IEEE, 2025.

[266] Jochen Suessmuth, Florian Fick, and Stan Van Der Vossen. Generative ai for concept creation in footwear design. In *ACM SIGGRAPH 2023 Talks*, pages 1–2, 2023.

[267] Jiangnan Li, Shuo Zhang, Li Sun, Jiantao Wu, Jian Song, Xiaotong Wang, and Jiamin Li. Enhancing product concept image generation through semantic feature prompts and lora training. *Scientific Reports*, 15(1):40795, 2025.

[268] David G Jansson and Steven M Smith. Design fixation. *Design studies*, 12(1):3–11, 1991.

[269] Samangi Wadinambiarachchi, Ryan M Kelly, Saumya Pareek, Qiushi Zhou, and Eduardo Velloso. The effects of generative ai on design fixation and divergent thinking. In *Proceedings of the 2024 CHI conference on human factors in computing systems*, pages 1–18, 2024.

[270] Richard Lee Davis, Thiemo Wambsganss, Wei Jiang, Kevin Gonyop Kim, Tanja Käser, and Pierre Dillenbourg. Fashioning creative expertise with generative ai: Graphical interfaces for design space exploration better support ideation than text prompts. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–26, 2024.

[271] DaEun Choi, Sumin Hong, Jeongeon Park, John Joon Young Chung, and Juho Kim. Creativeconnect: Supporting reference recombination for graphic design ideation with generative ai. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–25, 2024.

[272] David Chuan-En Lin, Hyeonsu B Kang, Nikolas Martelaro, Aniket Kittur, Yan-Ying Chen, and Matthew K Hong. Inkspire: Supporting design exploration with generative ai through analogical sketching. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–18, 2025.

[273] Hyeonsu B Kang, David Chuan-En Lin, Yan-Ying Chen, Matthew K Hong, Nikolas Martelaro, and Aniket Kittur. Biospark: Beyond analogical inspiration to llm-augmented transfer. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–29, 2025.

[274] Karl Sims. Artificial evolution for computer graphics. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 319–328, 1991.

[275] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2002.

[276] Yanan Wang and Yan Pei. A comprehensive survey on interactive evolutionary computation in the first two decades of the 21st century. *Applied Soft Computing*, 164: 111950, 2024.

[277] Yarden Frenkel, Yael Vinker, Ariel Shamir, and Daniel Cohen-Or. Implicit style-content separation using b-lora. In *European Conference on Computer Vision*, pages 181–198. Springer, 2024.

[278] Tuna Han Salih Meral, Enis Simsar, Federico Tombari, and Pinar Yanardag. Contrastive test-time composition of multiple lora models for image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18090–18100, 2025.

[279] Jimmy Secretan, Nicholas Beato, David B D'Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T Folsom-Kovarik, and Kenneth O Stanley. Picbreeder: A case study in collaborative evolutionary exploration of design space. *Evolutionary computation*, 19(3):373–403, 2011.

[280] Joel Simon. Artbreeder. **https://www.artbreeder.com**, 2018.

[281] Ole Hall and Anil Yaman. Collaborative interactive evolution of art in the latent space of deep generative models. In *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, pages 194–210. Springer, 2024.

[282] Sirui Tao, Ivan Liang, Cindy Peng, Zhiqing Wang, Srishti Palani, and Steven P Dow. Designweaver: dimensional scaffolding for text-to-image product design. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–26, 2025.

[283] Evans Xu Han, Alice Qian Zhang, Haiyi Zhu, Hong Shen, Paul Pu Liang, and Jane Hsieh. Poet: Supporting prompting creativity and personalization with automated expansion of text-to-image generation. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–18, 2025.

[284] Sangho Suh, Meng Chen, Bryan Min, Toby Jia-Jun Li, and Haijun Xia. Luminate: Structured generation and exploration of design space with large language models for human-ai co-creation. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–26, 2024.

[285] Mingxu Zhou, Dengming Zhang, Weitao You, Ziqi Yu, Yifei Wu, Chenghao Pan, Huiting Liu, Tianyu Lao, and Pei Chen. Stylefactory: Towards better style alignment in image creation through style-strength-based control and evaluation. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–15, 2024.

[286] CivitAI. CivitAI: Open-source generative AI platform. **https://civitai.com**, 2023. Community platform hosting 100,000+ LoRA style models.

[287] Vizcom. Vizcom: AI-powered industrial design tool. **https://www.vizcom.ai**, 2024. AI rendering tool for industrial designers.

[288] AUTOMATIC1111. Stable diffusion web ui. **https://github.com/AUTOMATIC1111/stable-diffusion-webui**, 2022. Open-source web interface for Stable Diffusion with multi-LoRA support.

[289] Anil R Doshi and Oliver P Hauser. Generative ai enhances individual creativity but reduces the collective diversity of novel content. *Science advances*, 10(28):eadn5290, 2024.

[290] Björn Hartmann, Loren Yu, Abel Allison, Yeonsoo Yang, and Scott R Klemmer. Design as exploration: creating interface alternatives through parallel authoring and runtime tuning. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 91–100, 2008.

[291] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9243–9252, 2020.

[292] Qian Wan and Zhicong Lu. Gancollage: A gan-driven digital mood board to facilitate ideation in creativity support. In *Proceedings of the 2023 ACM Designing Interactive Systems Conference*, pages 136–146, 2023.

[293] Tomas Cabezon Pedroso and Daragh Byrne. Browsing the latent space: A new approach to interactive design exploration for volumetric generative systems. In *Proceedings of the 15th Conference on Creativity and Cognition*, pages 330–333, 2023.

[294] David Chuan-En Lin, Fabian Caba Heilbron, Joon-Young Lee, Oliver Wang, and Nikolas Martelaro. Videomap: Supporting video exploration, brainstorming, and prototyping in the latent space. In *Proceedings of the 16th Conference on Creativity & Cognition*, pages 311–327, 2024.

[295] Shm Garanganao Almeda, John Joon Young Chung, Bjoern Hartmann, Sophia Liu, Brett Halperin, Yuwen Lu, and Max Kreminski. Artographer: a curatorial interface for art space exploration. *arXiv preprint arXiv:2512.02288*, 2025.

[296] John Joon Young Chung and Max Kreminski. Patchview: Llm-powered worldbuilding with generative dust and magnet visualization. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–19, 2024.

[297] Nicolai Marquardt, Asta Roseway, Hugo Romat, Payod Panda, Michel Pahud, Gonzalo Ramos, Steven M Drucker, Andrew D Wilson, Ken Hinckley, and Nathalie Riche. Imaginationvellum: Generative-ai ideation canvas with spatial prompts, generative strokes, and ideation history. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–19, 2025.

[298] Sangho Suh, Michael Lai, Kevin Pu, Steven P Dow, and Tovi Grossman. Storyensemble: Enabling dynamic exploration & iteration in the design process with ai and forward-backward propagation. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pages 1–36, 2025.

[299] Hanshu Shen, Lyukesheng Shen, Wenqi Wu, and Kejun Zhang. Ideationweb: Tracking the evolution of design ideas in human-ai co-creation. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–19, 2025.

[300] Wen-Fan Wang, Chien-Ting Lu, Nil Ponsa i Campanyà, Bing-Yu Chen, and Mike Y Chen. Aideation: Designing a human-ai collaborative ideation system for concept designers. In *Proceedings of the 2025 chi conference on human factors in computing systems*, pages 1–28, 2025.

[301] Ming Zhong, Yelong Shen, Shuohang Wang, Yadong Lu, Yizhu Jiao, Siru Ouyang, Donghan Yu, Jiawei Han, and Weizhu Chen. Multi-lora composition for image generation. *arXiv preprint arXiv:2402.16843*, 2024.

[302] Xiandong Zou, Mingzhu Shen, Christos-Savvas Bouganis, and Yiren Zhao. Cached multi-lora composition for multi-concept image generation. *arXiv preprint arXiv:2502.04923*, 2025.

[303] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. *arXiv preprint arXiv:2309.08532*, 2023.

[304] Xiaohan Peng, Janin Koch, and Wendy E Mackay. Designprompt: Using multimodal interaction for design exploration with generative ai. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference*, pages 804–818, 2024.

[305] Yuwen Lu, Alan Leung, Amanda Swearngin, Jeffrey Nichols, and Titus Barik. Misty: Ui prototyping through interactive conceptual blending. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–17, 2025.

[306] Richard Dawkins. *The blind watchmaker: Why the evidence of evolution reveals a universe without design*. WW Norton & Company, 1996.

[307] Philip Bontrager, Wending Lin, Julian Togelius, and Sebastian Risi. Deep interactive evolution. In *International Conference on Computational Intelligence in Music, Sound, Art and Design*, pages 267–282. Springer, 2018.

[308] Jacob Schrum, Jake Gutierrez, Vanessa Volz, Jialin Liu, Simon Lucas, and Sebastian Risi. Interactive evolution and exploration within latent level-design space of generative adversarial networks. In *Proceedings of the 2020 genetic and evolutionary computation conference*, pages 148–156, 2020.

[309] Carlos Tejeda-Ocampo, Armando Lopez-Cuevas, and Hugo Terashima-Marin. Improving deep interactive evolution with a style-based generator for artistic expression and creative exploration. *Entropy*, 23(1):11, 2020.

[310] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[311] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D^3 data-driven documents. *IEEE transactions on visualization and computer graphics*, 17(12):2301–2309, 2011.

[312] Black Forest Labs. FLUX.1: Open-weight text-to-image models. **https://github.com/black-forest-labs/flux**, 2024. Open-weight diffusion model with native multi-LoRA support.

[313] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[314] Bret Victor. Inventing on principle. Invited talk, Canadian University Software Engineering Conference (CUSEC), 2012. URL **https://vimeo.com/36579366**.

[315] Jacob O. Wobbrock, Shaun K. Kane, Krzysztof Z. Gajos, Susumu Harada, and Jon Froehlich. Ability-based design: Concept, principles and examples. *ACM Transactions on Accessible Computing*, 3(3):9:1–9:27, 2011. doi: 10.1145/1952383.1952384.