Interactive Flexible Style Transfer for Vector Graphics

Jeremy Warner
UC Berkeley
Berkeley, CA, USA

Kyu Won Kim
UC Berkeley
Berkeley, CA, USA

Björn Hartmann
UC Berkeley
Berkeley, CA, USA

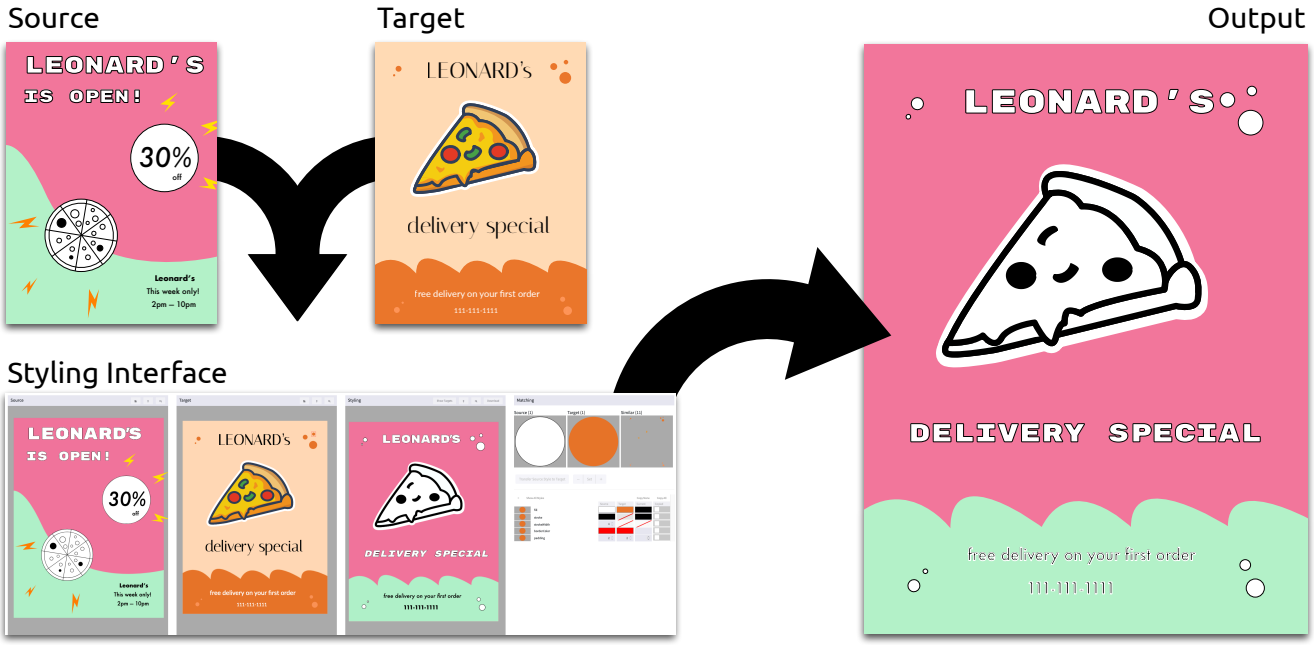

Figure 1: VST generates new output graphics by transferring visual styles from source graphics onto target graphics. The styling interface lets designers customize which styles to transfer, filter which elements to stylize, and preview the new stylized output graphics. The output graphics retain a similar structure to the target graphics while bearing styles from the source graphics.

ABSTRACT

Vector graphics are an industry-standard way to represent and share visual designs. Designers frequently source and incorporate styles from existing designs into their work. Unfortunately, popular design tools are not well suited for this task. We present VST, *Vector Style Transfer*, a novel design tool for flexibly transferring visual styles between vector graphics. The core of VST lies in leveraging automation while respecting designers' tastes and the subjectivity inherent to style transfer. In VST, designers tune a cross-design element correspondence and customize which style attributes to change. We report results from a user study in which designers used VST to control style transfer between several designs, including designs participants created with external tools beforehand. VST shows that enabling design correspondence tuning and customization is one way to support interactive, flexible style transfer.

UIST '23, October 29–November 01, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0132-0/23/10.
https://doi.org/10.1145/3586183.3606751

CCS CONCEPTS

• **Computing methodologies** → **Computer graphics**; *Image manipulation*; *Graphics systems and interfaces.*

KEYWORDS

vector graphics, style transfer, graphic design, creativity support tools, human-AI collaboration, computational design tools

ACM Reference Format:
Jeremy Warner, Kyu Won Kim, and Björn Hartmann. 2023. Interactive Flexible Style Transfer for Vector Graphics. In *The 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23), October 29–November 01, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3586183.3606751

1 INTRODUCTION

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

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

One potential solution is to use document-level themes or rules that consistently apply visual attributes to classes of objects. This approach is standard across many design and presentation software tools. For example, web pages use CSS (Cascading Style Sheets) to enable document-level styling, but these style-content links must be manually created and maintained. A notable downside of using document themes or stylesheets is their *rigidity*. Compelling themes require element class information and pre-planning, introducing *viscosity* [15] into the authoring process. Despite CSS support in SVG [62] via the `<use>` tag [42], most vector graphics avoid it.

Another promising direction is to *automatically* transfer visual styles between graphics using information on how two given designs relate to each other. However, this approach often fails to transfer styles as each designer uniquely intends. This failure stems from two sources: 1) the accuracy limitations of the algorithm and 2) the inherent subjectivity around *good* style and varying tastes that designers may have. A fully automated approach may transfer styles in undesired or unpredictable ways. The lack of adequate designer controls is a clear barrier to levering automation [49].

A tool should enable rapid iterating on different possible style transfer results to address the shortcomings of a fully automatic style transfer approach. Our research aims to combine the benefits of automation with effective controls for customizing and exploring design variations., Our approach combines automatically generated design correspondences with interactive control of how and where to transfer styles. We leverage prior work [54] on generating an automatic correspondence between vector graphics. This method yields a between-design element correspondence (Fig. 2) and element-wise similarity along multiple dimensions.

We present a new design tool, VST, short for *Vector Style Transfer*. VST provides designers with an interface to visualize and customize how style flows across designs (Fig. 3). VST displays a dynamic list of element styles, allowing designers to easily copy, reset, and customize element style attributes (see Appendix C for all attributes). With VST, designers can map and remap example Source element styles onto contextually similar elements. VST also features fast and flexible ways to identify, select, and style Target elements. The Output canvas re-renders the stylized Target graphics in real-time with any changes, providing immediate visual feedback.

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

To evaluate VST's style transfer capability, we recruited six designers to transfer styles between nine designs. Each designer participating in the study successfully used VST to interactively transfer styles to their satisfaction and make nine new Output designs. In a follow-up design replication study, we recruited four expert designers to each manually replicate six of these Output designs in their preferred design tool. The results from this preliminary study suggest that someone using VST may reduce the time and work for this style transfer task compared to experienced designers using industry-standard tools. Our contributions include the following:

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

2 RELATED WORK

The most relevant prior work follows several themes: supporting creative processes with automation, inferring design structures, automatic transfer techniques, and other advanced vector graphics design tools. We review each of these in turn.

2.1 Supporting Creative Processes with AI

While automation is powerful, gracefully integrating it into existing creative practices demands care. Regarding working with AI as a design material, scholars have elaborated on the need for retaining control [45, 49, 50, 55, 59, 68]. For GUI design, Dayama et al. present a method for interactive layout transfer, where the layout of a source design is transferred automatically using a selected template layout while complying with relevant guidelines [6]. In photography, researchers have provided mechanisms for guiding photographers to optimize image aesthetics [35] and to find ideal portrait lighting conditions [11]. Goal-oriented transformations can also be applied to existing designs (e.g., improving accessibility) [69] or to produce alternative designs for different viewports [21].

Our rationale for using element relationships between designs as a primary mechanism for transfer is that this mirrors how designers tend to work already when manually transferring styles. Highly related to our line of work are feedforward and example-driven corrections. Feedforward work refers to showing the user the output or result of their action before it happens–a preview of applying different interface actions [10, 29, 61]. For example, OctoPocus provides dynamic guidance to bolster users' ability to learn stroke-based gestures [2]. Example-driven corrections and interaction models like those in FlashMeta [46] or programming-by-demonstration disambiguation models [41] provide alternative techniques that address similar problems. Feedforward and inherent feedback can promote

UI element functionality understanding to users, though computing this information fast enough for live, interactive contexts can be challenging. With that said, cluing in authors on their actions' impact is valuable. For example, the Lightspeed rendering pipeline enabled interactive prototyping of professional 3D graphics, enabling more design variation exploration [47]. One approach might leverage lower-fidelity previews of variations when interacting with automation, such as design galleries. We avoid using design galleries as our early prototypes showed the varying complexity and breadth were visually overwhelming. For an analogy in text editing: VST spell-checks the entire document, while feedforward suggests autocompletion options given what is already written.

Example-based corrections generate a program that satisfies all demonstrated changes, iteratively growing more complex. Example-based style retargeting for websites provides a successful analog to vector graphic style transfer in HTML/CSS [3, 34]. Example galleries can effectively support open-ended design authoring, where styles come from potentially multiple sources [36]. While the document-object-model hierarchy is essential to styling web pages, such grouping structures and labels are entirely optional and often absent in vector graphics. Groups may be constructed arbitrarily (e.g., for editing convenience) rather than having any consistent semantic meaning. Designers can encode hierarchical information through groups but frequently opt to style elements directly [50]. Bringing interactive style transfer to vector graphics is a unique problem.

2.2 Inferring Design Structures

Researchers have used several approaches to infer underlying or implicit structures in visual designs. Traditionally, this work primarily operates on some structured representation (like HTML or SVG). For user interfaces, large libraries have helped to characterize and infer document structure [7, 33]. Linking styles via direct manipulation and element cloning provide a clear view and control of an element's style properties [20]. There is also work to recognize higher-level design patterns through designs by inducting grammars [57]. For the domain of D3 visualizations, Hoque et al. map data types onto shapes/axes to help search for relevant designs [22]. Harper et al. showcase tools for deconstructing and restyling a D3 visualization by extracting the data and modifying visual attributes of marks [16]. More recent work also focuses on inferring design structure from images directly. Computer vision techniques are improving on reverse engineering user interface models directly from screenshots [12, 53, 64]. Similar work using vision-based methods has helped leverage attention towards answering questions and understanding mobile UIs [38, 52, 56]. Reddy et al. use differentiable compositing to identify pattern instances within a design [48]. Scene graphs have also characterized structural relationships within and between 3D environments [14]. For vector graphics, Shin et al. demonstrate a technique using graph kernels to find relationships between elements of designs [54]. We leverage this preexisting automatic technique to compute a correspondence between design elements (like those shown in Fig. 2). The contribution of this work centers on our novel design tool that goes beyond pure algorithmic automation by enabling flexible interactions between the capabilities of such an algorithm and the designer's high-level styling goals.

2.3 Automatic Transfer Techniques

While automatic style transfer techniques can generate impressive image transformations, they are generally functional as *theme selections*. Due to the broad range of shape primitives, graphic designs do not immediately lend themselves to this document-level style transfer approach. The selective extraction and transfer of specific styles are too precise to be encoded in a one-dimensional slider [24, 28]. The variations of vector designs also make mapping onto an otherwise standard template difficult (e.g., facial key points) [58]. Additionally, text can be used to edit image content and style directly [4]. While layout is not our tool's focus, prior work highlights optimization techniques that can be used to automatically format text documents [23]. ImagineNet restyles mobile apps with neural style transfer and updating assets in place [13]. To be stylized with image-based techniques, vector graphics must first be rasterized, losing future object-level awareness and scaling abilities. The state of the art in automatic vector generation includes leveraging pixel-based diffusion models [27] by leveraging a differentiable vector graphics representation [39]. DeepSVG uses GANs to generate and interpolate between SVG icons and shares a large-scale SVG dataset [5]. Kotovenko et al. model a painting using discrete strokes to recreate style transfer better [32]. Within font, some work shows the possibility of even inferring and transferring style between font glyphs [8, 40]. These techniques often give users little to no control of *how* the style is transferred. Our work focuses on optimizing the potential value that these automatic approaches can provide by introducing meaningful high-leverage interactions to customize and control generated output while retaining the core vector graphics representation that designers are familiar with working with.

2.4 Vector Graphics Design Tools

Several techniques for authoring or adjusting vector graphics exist and inform this work. Object-Oriented Drawing introduces a new way to create and style elements directly on the canvas [65]. DataInk supports cloning and binding user-generated symbols to data, facilitating lightweight restyling [66]. Sketch-n-Sketch links drawing code and vector graphics, letting users directly edit the SVG in a canvas, modifying the code which generates it [18]. For mathematical diagramming, Penrose uses layout energy-minimization techniques coupled with a language for specifying explicit styles and content of what to render [67]. Falx uses user demonstrations and program synthesis to create new visualizations [63]. Existing tools can even convert web designs into a vector layout [9]. Para supports binding procedural art generation constraints with graphics, including cases where there are many-to-many constraints [26]. A follow-up project, Dynamic Brushes, combined procedural programming into brush behavior and design, enabling more custom expression [25]. Other design tools have looked at supporting design layout [30, 44], fashion [60] and design coloring [17, 70].

3 VECTOR STYLE TRANSFER

When transferring styles between vector graphics, designers may identify an inspirational style they want to copy from a Source design. Next, in a Target design, they may identify design elements they would like to stylize. Then, they will update the stylistic attributes of those relevant Target elements using the Source style

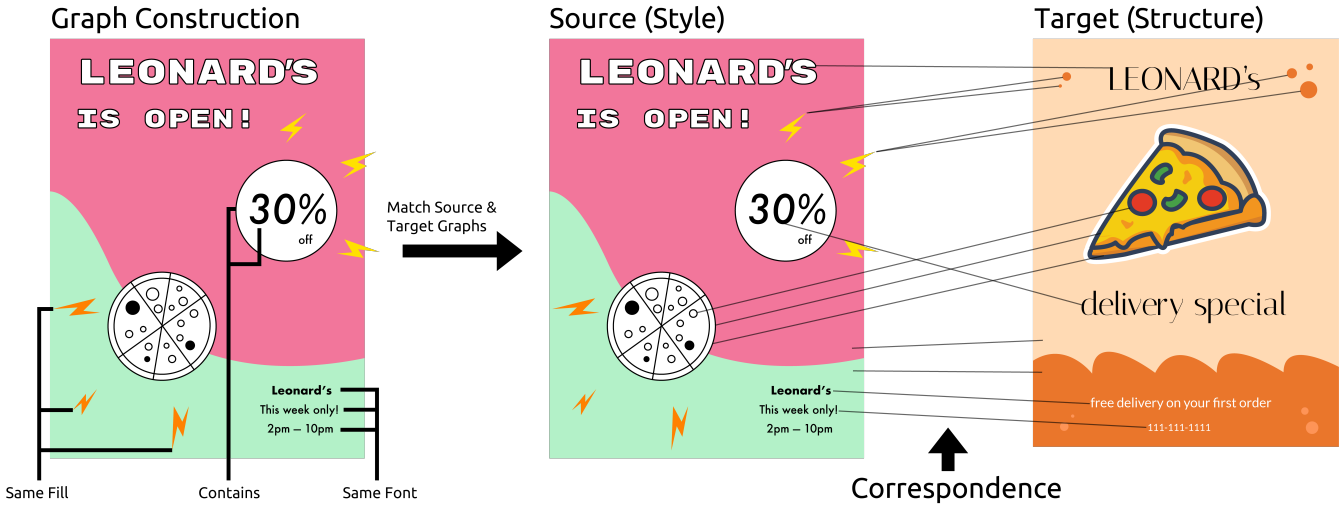

Figure 2: An overview of automated design correspondence. To relate design elements, we first construct a graph from each given design, where the *vertices* are primitive design elements (e.g., shapes, text, images) and *edges* are semantic relationships (e.g., same fill, containment, same font). Once the Source and Target graphs are constructed, we then compute a correspondence between the two designs' elements using the technique previously detailed in [54]. This automatically generated correspondence is VST's basis for (a)how to find similar elements *within* a design (e.g., for easier selection/styling) and (b) identifying which elements are similar to each other *across* designs (e.g., determining which initial styles to transfer). Each Target element is linked to a single Source element. Only a subset of links between these designs' elements are shown.

as a reference. Alternatively, they may first focus on the Target design they wish to change and pull stylistic influences in from a range of Sources, exploring possible variations. Generally, this styling is an iterative and flexible process that involves reasoning about (a) *which elements correspond to each other across designs* and (b) *which style attributes to transfer*. There is subjectivity regarding the most desired application of style, and higher-level considerations like the overall cohesion of the Target design after styles have transferred further complicate this task. The resulting Output design has the *style* of one design and the *content/structure* of another – though this distinction is still inherently subjective. Still, this task (using examples to update existing graphics with new visual styles) is expected in the graphic design process [19, 31, 37].

3.1 Design Goals

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

- **DG1** Let designers powerfully tune design correspondences.
- **DG2** Enable flexible control over which styles are transferred.
- **DG3** Reduce the work and time needed for transferring styles.

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

3.2 Exemplar Scenario

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

This style unification process Xavier faces involves many repeated manual edits and cross-references. Instead of manually ensuring exact visual consistency, he opens VST and loads in both graphics (Source: the new flyer, Target: the previous advertisement). VST computes a correspondence between elements of these two designs and automatically copies styles between matches. This correspondence technique ensures a one-to-many mapping from the Source elements to the Target elements. This ensures that every Target element will be matched, while some Source elements may not be initially matched. Xavier then sees the Output canvas update with newly stylized graphics (Fig. 3).

Figure 3: An overview of the VST interface, including (A) the Source graphics (where the style is sourced from), (B) the Target graphics, (C) the Output canvas (the current style transfer result), and (D) customization controls for matching element styles across canvases and filtering which style attributes to copy or modify. Designers can filter this list of attributes (shown in D) based on the current selection to do more focused editing or instead modify shared style attributes across the entire design.

For each Target element, styles are copied from the most similar Source element as determined by the design correspondence algorithm [54]. In addition to seeing the updated target graphics, a list of changed style attributes is displayed on the right-hand side of the interface (Fig. 3D). The breadth of style attributes and the range of possible valid matches between elements makes using a fully automatic approach difficult. The inherent subjectivity of style also means this first attempt will not always be correct, especially for more complex and open-ended designs.

Xavier immediately detects outlier text elements that are visually misaligned with the Source style directly on the Output canvas (Fig. 3). Designers are trained to use gestalt principles of perception to organize a design. Incorrect style transfer will lead to visual violations of these principles, which are often easy to detect [43]. This means that some elements likely have been 'mismatched' by the correspondence algorithm (Fig. 4). Using the Source canvas (Fig. 3A), Xavier can then specify which Source element the incorrectly styled text fields should visually match. When he presses the *Transfer Source Style to Target* button (Fig. 10), VST renders styles from the Source element onto the Target selection in the rightmost Output canvas (Fig. 3C). Behind the scenes, VST applies these fixes to a copy of the original correspondence, avoiding recomputing the entire correspondence after updates.

Still, manually selecting each target element to update is tedious. To enable faster transfer, designers can double-click on any Target element to select similar elements, as determined by the design correspondence. Repeatedly double-clicking an element iteratively grows the set of selected Target elements. This feature mirrors the multi-click selection in other media, like toggling between word-sentence-paragraph selections within a text document. Here, we use the underlying within-document element-wise similarity score to intelligently add elements most similar to the currently active selection. A similarity score is computed for each element relative to the currently selected elements, and the elements with the highest score is added to the active selection. Double-clicking on a Source element conversely selects all Target elements currently matched to that element, which shows how style flows from the Source to Target design. The customization panel shows a pane of similar elements, where Xavier can preview this selection (Fig. 10).

Despite Xavier updating the Source-Target correspondence, the resulting Output design still has some problems. For example, while the font and color are corrected, the copied font size makes some elements not fit neatly in the new design (Fig. 11). Once matched, VST has controls for customizing which specific style attributes are transferred. To focus on the desired element, he clicks *Show Filtered Style* to only see the styling applied to the text element (Fig. 11). He toggles the `fontSize` attribute, resetting that element's font size and updating the Output canvas. Similar attribute values are grouped in this view to make selecting and editing easier. He continues this style transfer process until he is satisfied with the quality of the new design. Internally, these changes build up a list of attribute transformations to apply to the Target design. The customization pane can highlight just the modified attributes, summarizing stylistic changes at a glance. Finally, Xavier downloads the Output graphics from VST as an SVG file to save his work.

3.3 Implementation

We used ReactJS to build the VST interface and deployed our prototype online. Vector graphics are rendered using FabricJS, a vector graphics library leveraging the HTML5 canvas backbone. SVG files, such as those exported from industry-standard design tools like Sketch, Figma, Canva, and Adobe Illustrator, can be directly imported. Once VST has imported the input Source and Target

graphics, we compute a correspondence between the two designs using a comparison technique introduced by Shin et al. [54]. This technique represents each design as a multigraph (rather than a typical parent-child hierarchy tree) to support matching elements across a broader range of similar attributes. Vertices are primitive design elements (e.g., shapes, text, images), and edges represent semantic relationships between elements (e.g., alignment, containment, same fill). This correspondence contains per-element similarity scores across several dimensions (e.g., color, shape, size, and text). In our implementation, correspondences between 20 or fewer elements are generally computed in real-time (< 1s). Though slower, our study's larger design pairs are still tractable to match, with the largest pair (185 total elements) taking about 100s. Our example set's average matching time per design pair (across Style Transfer Tasks 1 and 2) is 7.78s. Once obtained, match information can be exported and saved for later use. A version of VST for styling pre-matched design pair examples is available at: https://berkeleyhci.github.io/vst/.

4 EVALUATIONS

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

RQ1 How would designers use VST for style transfer?
RQ2 Could VST stylize realistic, open-ended designs?
RQ3 Could VST reduce the time or work of styling?

4.1 Style Transfer Evaluation

Method – To answer RQ1 and RQ2, we ran an exploratory study with six experienced designers (D1-6). Before the study began, we asked designers to create a new design from a given prompt with their preferred design tool. The prompt requested a single menu page design for a local restaurant's (*Leonard's*) mobile phone application. The goal was to include designer-provided source graphics to create a more realistic style transfer scenario. More methodology details are available in Appendix A, and more information about the participant's background is in Appendix B.

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

Task 2: Open-Ended Transfer – To observe how VST handled styling more open-ended realistic designs (RQ2), designers transferred styles from their externally created designs onto three new related templates (T2.1-3). In these tasks, the Source was a menu page created by each designer before the study with their preferred design tool. We matched their designs to three new template pages (a loading screen, a reviews page, and a checkout cart), all for *Leonard's* mobile app. The generated output design correspondences (Fig. 2) were not hand-tuned at all before the study.

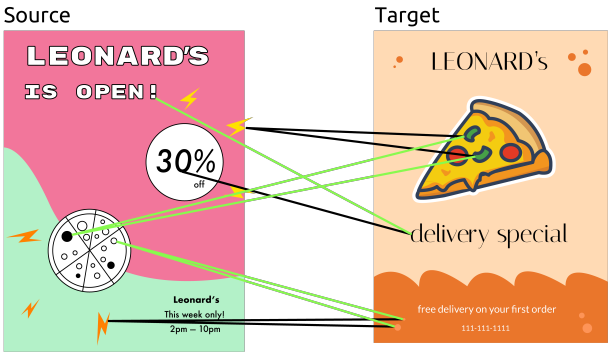

Figure 4: The black lines show an initial correspondence between the elements of the Source and Target designs. The green lines show an alternative, more desired set of links. When users select their desired Source and Target elements and press *Transfer Source Style*, VST will update these links, redirecting the flow of visual styles across designs.

4.2 Style Transfer Results

Our style transfer evaluation study found that designers could use VST to control style transfer across basic designs (RQ1), even generating variety in their Output designs from the same inputs. Those designers successfully used VST to flexibly transfer styles from more realistic, open-ended designs created with external tools (RQ2). We take this as an indication that VST enabled the style transfer it was designed to support. Each designer participating in the study (D1-6) used VST to generate eight new Output designs successfully. Designers also answered Likert-scale questions regarding their experience with VST (Fig. 8). Style transfer examples from the evaluation are shown in Figures 5 and 6.

Designers, despite never using a similar interface before, used VST's features to both (a) modify design correspondences (DG1) and (b) filter and edit styles per correspondence (DG2). Software instrumentation revealed that almost all designers on almost all tasks used VST to tune computed correspondence matches. On average, designers performed 6 such corrections per task. While making these corrections, designers used the functionality to *select similar* elements to the ones they manually selected. On average, designers performed 7.3 similarity selections and spent about 4.8 minutes per task. As a reminder, designers were only instructed to match the styles to the best of their ability – not to do so as quickly or efficiently as possible. We showcase additional, more complex VST graphics made outside of this study in the Appendix (Fig. 12) and in our paper's accompanying project video.

VST let designers tune design correspondences (DG1). Overall, designers appreciated the style transfer control that VST provided them. The designers' Likert-scale responses indicated they could produce designs they were satisfied with (Fig. 8). Most designers could see themselves using the tool again and found VST flexible enough to perform style transfer as they intended. Their verbal remarks are corroborated by the frequency with which they used the correspondence correction feature (Average: μ : 6.0, Standard Deviation: σ = 3.8) and attribute editing feature (μ : 24.0, σ = 17.3).

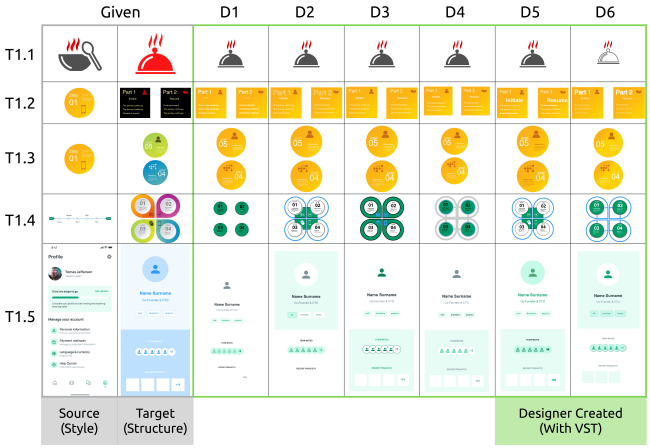

Figure 5: Task 1 (Style Transfer) – Basic Graphics Pairs. Here, designers D1-6 **used VST to transfer styles from the** Source **to the** Target **graphics. Both** Source **and** Target **designs were provided to the designers. In simpler cases, the design transfer result is uniform across designers (T1.1-3). Still, despite each designer starting from the same pair of designs, variations arose in more complex design pairs (T1.4-5).**

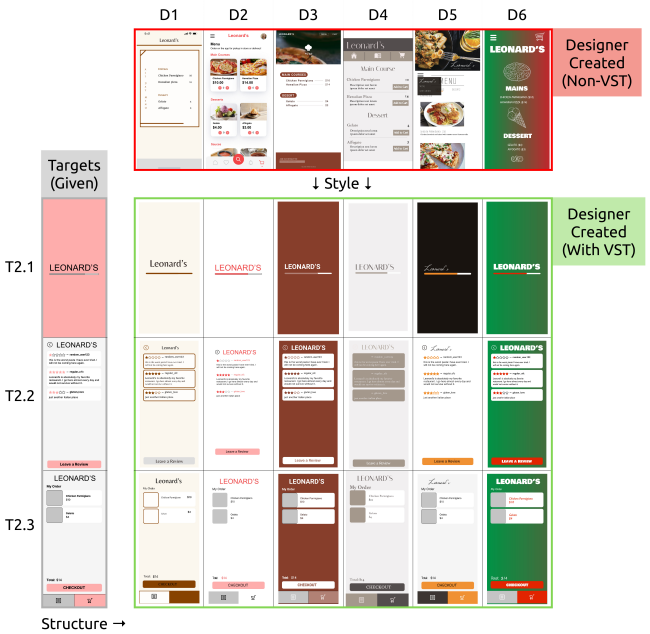

Figure 6: Task 2 (Style Transfer) – Open-Ended Transfer. Before the study, we gave designers (D1-6) **a prompt for a menu design with specific elements without any style instructions. The column header shows designs that they brought into the study (**Sources**), and the row header shows design templates (**Targets**). The inner table shows new designs created by applying styles from their externally created** Source **design onto previously unseen** Target **templates. Inspecting each column shows a unified visual style inherited from the** Source **document, while rows show the** Target **structure.**

VST enabled flexible control of style transfer (DG2). The designers created a wide variety of designs, even when given the same input graphics (Fig. 5). For their own provided graphics, designers reproduced a consistent theme across a set of provided vector graphics templates (Fig. 6). Several designers remarked on the convenience of reusing visual styles directly. D4: Very fun! Appealing to a visual thinker who values efficiency and hates repeatedly doing the same things. Magical, "it read my mind!" kind of feeling. While most found it clear how to use the different parts of the prototype to achieve their desired style transfer, there was also feedback that the transfer results were sometimes surprising. This surprise likely stemmed from having multiple ways to style elements (e.g., tuning the correspondence vs. what styles the correspondence transfers).

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

Correspondence-based transfer presents novel controls. No designer reported using a similar style transfer design tool before this study. D6: I have not used anything that performed this exact function before, but I've used a tool to try to analyze an image and find out what fonts were used. It was not as reliable as this tool. While most designers (4/6) indicated an interest in using the tool again, others were hesitant, citing VST's deviation from the types of tools they were familiar with. Some designers recognized the value of a style transfer tool: D4: I have manually copied styles and have had other humans manually copy my own. When successful, this tool manages to give you that feeling of empathy and creative connection ("Wow, the other designer understood my aesthetic and was able to replicate it! I feel they really understand my vision"). When it is not successful, it is easier and less stressful to correct than a human might be. Plus, it is faster than asking another designer, fewer resources, less risk, and when it is successful, high reward!

4.3 Design Replication Evaluation

Method – To answer RQ3, we ran a follow-up study. Our goal with this study was to compare the time and work required for style transfer in VST with that of an expert using industry-standard design software. We recruited four new expert designers as replication designers (RD1-4). More information about their background is in Appendix B. They were tasked with recreating a subset of the Output graphics from the previous study (T2.1-3) in their preferred design tool (Adobe Illustrator). Given that VST is a novel design tool there are no users with equivalent VST expertise comparable

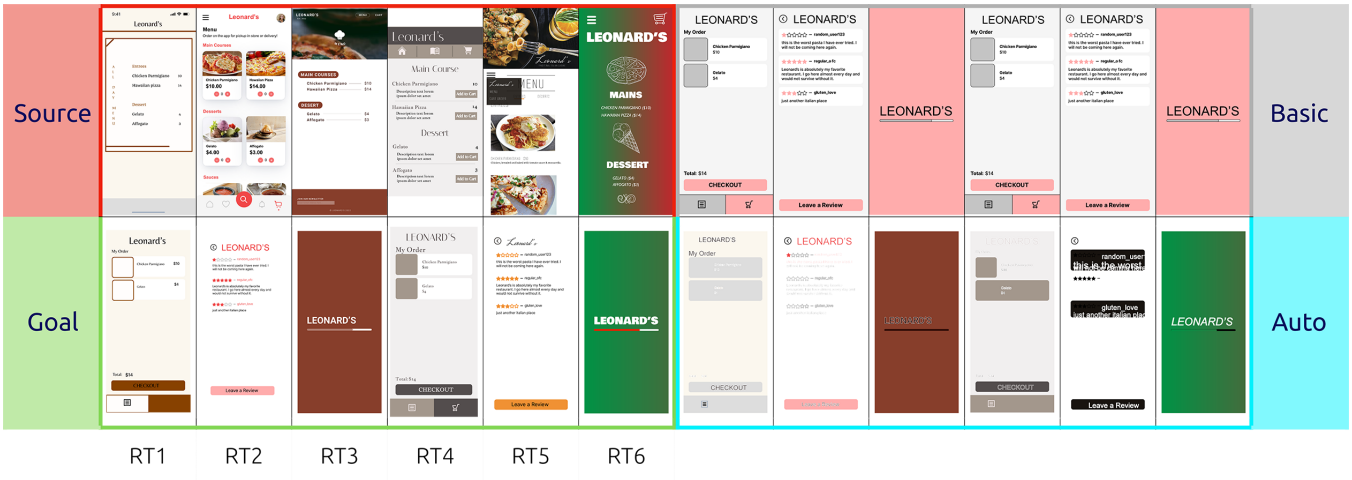

Figure 7: Design Replication Task – A new set of expert designers (replication designers RD1-4) replicated six reference designs (RT1-6) from the previous style transfer evaluation tasks (Fig. 6) using two different starting points: *Basic* and *Auto*. The first approach involved using Illustrator to transform the *Basic* input design to the replication goal. The second approach again used Illustrator, but instead has the algorithmic output (*Auto*) as the starting point. We provided the RDs with source styles and target structures from the previous study in vector form and a reference image of the replication goal for both approaches.

to the RDs' Illustrator skill. To approximate the performance of an expert VST user, the authors used VST to generate the same Output designs using the same input materials provided to the RDs. This data is labeled VST in Table 1. Further methodology details are in Appendix A. We report the comparison between these three design replication methods in our results.

Task: Design Replication – We selected six Output design examples from Fig. 6 for this designer to replicate in Illustrator (*Goal* in Fig. 7). We selected designs to include both graphics from every task (T2.1-3) that we gave the original designers and to include one example per designer (D1-6). We provided the RDs with the Source and Target vector graphics files and an image of the generated Output (created initially by D1-6). The RDs were then tasked with transforming the Target graphics to resemble the provided Output. To measure what human adjustment is needed when working with the automatically stylized designs, we also asked the RDs to replicate the Output starting with the initial automatically stylized Output graphics from VST. These graphics (*Auto*) are created by copying all styles using the initial automatic Source and Target correspondence. We asked the RDs to transform the now-partially stylized Target graphics to resemble the Output image. Any difference between these two sets (*Basic* and *Auto*) would highlight the algorithm's impact on the task time and work. To compare the potential of VST and existing tools, the authors also replicated the same Output designs from the previous study using VST (RT1-6). The same input materials were used as in the Illustrator replication: the Source and Target vector graphics files and an Output image.

4.4 Design Replication Results

In our study, using VST to transfer styles was faster than expert replication designers (RD1-4) transferring styles within their preferred design tool (RQ3). The RDs also performed more edit and selection operations using Illustrator than the authors using VST. We report total work as a combination of selection and edit operations. On average, the RDs spent 534 seconds replicating from scratch (*Basic*) and 774 seconds replicating from the output of the correspondence algorithm (*Auto*). In comparison, the authors required, on average, 129 seconds to match styles using VST. A plot of the duration for each task is shown in Fig. 9. Stats averaged over all tasks (RT1-6) are shown in Table 1. Each replication designer also reported the style replication task as difficult and tedious.

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

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

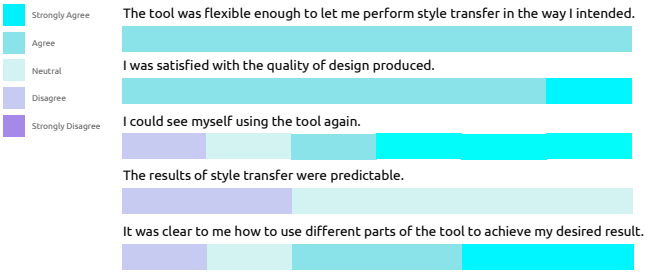

Figure 8: Summary of Likert survey data from designers D1-6**.**

time cleaning. RD4: I largely had a similar approach to both design files, though the original [Basic] one tended to be easier.

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

5 DISCUSSION

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

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

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

The Cost of Automation – One notable point in our results is that starting with the algorithm's output (*Auto*) did not make replication easier. In fact, the RDs reported that starting with the automatically generated algorithm output was more difficult and

Table 1: Replication work data – usage statistics averaged over replication tasks RT1-6 (see Fig. 7). The *Basic* and *Auto* columns show aggregate data collected from the four expert replication designers (RD1-4)**, while the *VST* column shows data from the paper authors using VST to replicate designs.**

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

less fun. Simply throwing automation into existing tools and processes may backfire. This is backed by our quantitative results: the *Auto* designs, on average, required more work to style than the corresponding *Basic* starting point (*Basic*: 265 operations, *Auto*: 383 operations). This is jarring, as applying the style transfer algorithm should have the opposite effect — otherwise, why apply it at all?

First, applying a semi-correct transformation reduces cohesion in the design. The lack of cohesion commonly found in *Auto* designs reduces the efficiency of applying gestalt principles. This makes selecting similar elements to style them together harder. Second, the vast scope of the copied attributes may introduce new work. Incorrectly changing an attribute does not create new work if it already needs to be changed. However, if part of a Source style is not desired in the Output graphics, those attributes must be manually reset to their original Target value. Current design software fails to support this type of style transfer interaction. In contrast, VST features convenient ways to quickly select and explore element styles (double-clicking an element/selection, precision selection controls, visually selecting via the same attribute value). Current correspondence algorithms do not seem to reduce the total work in style transfer otherwise. This is especially true for more complex examples where correspondence accuracy is often lower.

6 LIMITATIONS AND FUTURE WORK

6.1 Limitations

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

We also did not measure the impact of algorithm matching performance on this task. Informally, study participants D1-6 updated the correspondence an average of six times per task, though our study instrumentation did not record the number of adjusted elements per update. In Shin's prior work [54], the average match accuracy was 95% (ranging from 78—100%). However, their evaluation [54]

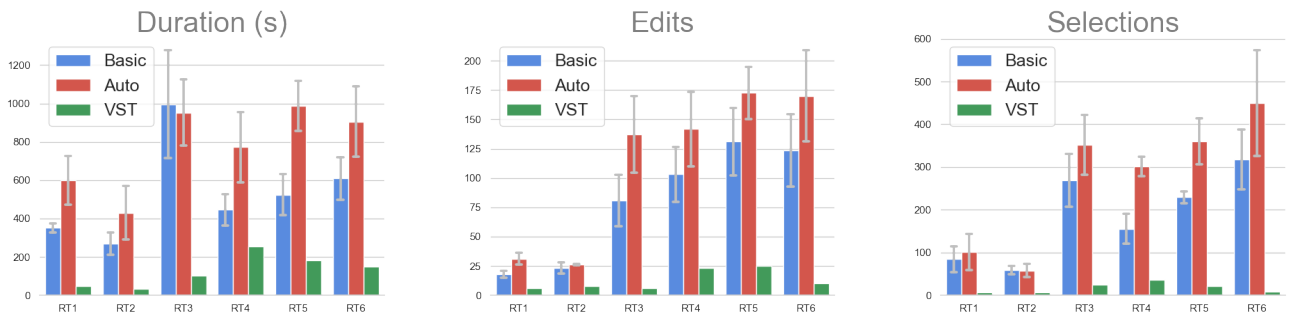

Figure 9: Plots of the duration, edits, and selections data from the design replication (RT1-6). Along each recorded measure (duration, edits, and selections), the authors using VST outperformed all four expert designers using Adobe Illustrator in replicating the stylized designs. The *Basic* and *Auto* plots also include ticks showing the *standard error* for each task computed over RD1-4. VST was only used once per task to obtain a baseline, so there are no comparable ticks to show.

was performed with the Source as an element group within a Target design, rather than a separate design. Explicitly varying the match quality and leveraging different matching techniques are opportunities for future work. Another limitation of this work is the smaller scale of the surveyed designer population (10 unique designers across both studies). For our design replication study, we worked with four expert designers. While this smaller study size allowed us to deepen the level of feedback and data we gathered, future studies could evaluate a larger expert population to get additional feedback. Future work could conduct a larger-scale study with more designers to potentially collect insights into a broader set of behaviors that designers exhibit. Also, when comparing VST to other tools, the authors have more awareness of the replication goal and task, which likely improves their relative performance. Another evaluation could train experienced designers with VST and have them replicate graphics from the original study.

6.2 Future Work

Images can naturally add vibrancy to a design, though VST's style transfer only applies to vector graphics. One future direction is sourcing vector styles directly from images. RD1: It would be great to apply bitmap styling to my vector design. This use case is more common in my workflow. This requires converting the image to vector graphics or a novel style extraction technique. Some features (e.g., colors) are simple to extract, while other features like paths, gradients, shapes, and fonts are potentially much harder to source from an image correctly. For image-to-vector graphics conversion, some research methods [51] and commercial tools [1] exist. However, these methods tend to optimize pixel-based similarity to the source image over a consistent output structure or element resolution. The internal document complexity makes determining correspondences much more challenging. Rasterizing vector graphics is a lossy process with no perfect inverse. Still, given the ubiquity of image-based inspiration, a vector styling tool that uses images as a styling source is an exciting future direction.

Better correspondence algorithms may reduce the need for a corrective interface like VST. Consider automatic speech transcription as an analogy: under a certain accuracy threshold, manually transcribing speech is easier than correcting a low-quality generated transcript. The work required to fix the algorithm's output exceeds that of simply creating that same output manually. There is room for improvement in design correspondence accuracy for vector graphics. However, even with the best algorithm, some cases will still need manual tuning. This ambiguity stems from the inherent subjectivity around *good* style and varying designer tastes.

Primarily, our style transfer with this prototype addresses element size, font, stroke, and fill. While designers can modify other features, this feature subset visually dominates the result. A complete list of transferrable properties is in Appendix C. Future work could serve as a larger-scale multi-design style linter or unification technique where many designs are edited simultaneously. The design layout and structure are held constant throughout our style transfer process. Applying the layout from source to target is an exciting and relevant next direction.

7 CONCLUSION

We presented a novel design tool called VST (Vector Style Transfer) for flexibly transferring styles across vector graphics designs. We conducted two studies to investigate (1) how designers may use correspondence-based transfer tools like VST and (2) the potential of these tools in relation to traditional industry-standard design tools (e.g., Adobe Illustrator). The first study, an open-ended style transfer evaluation, revealed that despite not previously using any similar tools, experienced designers could effectively transfer styles even across graphics independently created using other design tools. The second study, a preliminary design replication evaluation, suggests that tools like VST may reduce the time and work required to transfer styles across designs compared to traditional design tools. These expert designers also found directly editing automatically stylized graphics more difficult and tedious than the original baseline design templates. This work provides two design recommendations for future design tools to support flexible user control: enable tuning generated design correspondences and customizing how these correspondences transform designs.

REFERENCES

[1] Adobe. 2022. *Adobe Express: Convert Images to SVG.* Adobe. Retrieved August 10, 2022 from https://www.adobe.com/express/feature/image/convert/svg

[2] Olivier Bau and Wendy E. Mackay. 2008. OctoPocus: A Dynamic Guide for Learning Gesture-Based Command Sets. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology* (Monterey, CA, USA) *(UIST '08)*. Association for Computing Machinery, New York, NY, USA, 37–46. https://doi.org/10.1145/1449715.1449724

[3] Edward O. Benson and David R. Karger. 2013. Cascading Tree Sheets and Recombinant HTML: Better Encapsulation and Retargeting of Web Content. In *Proceedings of the 22nd International Conference on World Wide Web* (Rio de Janeiro, Brazil) *(WWW '13)*. Association for Computing Machinery, New York, NY, USA, 107–118. https://doi.org/10.1145/2488388.2488399

[4] Tim Brooks, Aleksander Holynski, and Alexei A Efros. 2023. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 18392–18402.

[5] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. 2020. Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems* 33 (2020), 16351–16361.

[6] Niraj Ramesh Dayama, Simo Santala, Lukas Brückner, Kashyap Todi, Jingzhou Du, and Antti Oulasvirta. 2021. Interactive Layout Transfer. In *26th International Conference on Intelligent User Interfaces* (College Station, TX, USA) *(IUI '21)*. Association for Computing Machinery, New York, NY, USA, 70–80. https://doi.org/10.1145/3397481.3450652

[7] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 845–854. https://doi.org/10.1145/3126594.3126651

[8] Praveen Kumar Dhanuka, Nirmal Kumawat, and Nipun Jindal. 2019. Vector based glyph style transfer. In *ACM SIGGRAPH 2019 Posters*. Association for Computing Machinery, New York, NY, USA, 1–2.

[9] divRIOTS. 2023. *html.to.design*. divRIOTS. https://www.figma.com/community/plugin/1159123024924461424/html.to.design

[10] Tom Djajadiningrat, Kees Overbeeke, and Stephan Wensveen. 2002. But how, Donald, tell us how? On the creation of meaning in interaction design through feedforward and inherent feedback. In *Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques*. Association for Computing Machinery, New York, NY, USA, 285–291.

[11] Jane L. E, Ohad Fried, and Maneesh Agrawala. 2019. Optimizing Portrait Lighting at Capture-Time Using a 360 Camera as a Light Probe. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. ACM, Association for Computing Machinery, New York, NY, USA, 221–232. https://doi.org/10.1145/3332165.3347893

[12] Shirin Feiz, Jason Wu, Xiaoyi Zhang, Amanda Swearngin, Titus Barik, and Jeffrey Nichols. 2022. Understanding Screen Relationships from Screenshots of Smartphone Applications. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 447–458. https://doi.org/10.1145/3490099.3511109

[13] Michael H. Fischer, Richard R. Yang, and Monica S. Lam. 2020. ImagineNet: Restyling Apps Using Neural Style Transfer. *CoRR* abs/2001.04932 (2020). arXiv:2001.04932 https://arxiv.org/abs/2001.04932

[14] Matthew Fisher, Manolis Savva, and Pat Hanrahan. 2011. Characterizing Structural Relationships in Scenes Using Graph Kernels. In *ACM SIGGRAPH 2011 Papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. Association for Computing Machinery, New York, NY, USA, Article 34, 12 pages. https://doi.org/10.1145/1964921.1964929

[15] Thomas R. G. Green and Marian Petre. 1996. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages & Computing* 7, 2 (1996), 131–174.

[16] Jonathan Harper and Maneesh Agrawala. 2014. Deconstructing and Restyling D3 Visualizations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 253–262. https://doi.org/10.1145/2642918.2647411

[17] Lena Hegemann, Niraj Ramesh Dayama, Abhishek Iyer, Erfan Farhadi, Ekaterina Marchenko, and Antti Oulasvirta. 2023. CoColor: Interactive Exploration of Color Designs. In *Proceedings of the 28th International Conference on Intelligent User Interfaces* (Sydney, NSW, Australia) *(IUI '23)*. Association for Computing Machinery, New York, NY, USA, 106–127. https://doi.org/10.1145/3581641.3584089

[18] Brian Hempel, Justin Lubin, and Ravi Chugh. 2019. Sketch-n-Sketch: Output-Directed Programming for SVG. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 281–292. https://doi.org/10.1145/3332165.3347925

[19] Scarlett R Herring, Chia-Chen Chang, Jesse Krantzler, and Brian P Bailey. 2009. Getting inspired! Understanding how and why examples are used in creative design practice. In *Proceedings of the SIGCHI conference on human factors in computing systems*. Association for Computing Machinery, New York, NY, USA, 87–96.

[20] Raphaël Hoarau and Stéphane Conversy. 2012. Augmenting the Scope of Interactions with Implicit and Explicit Graphical Structures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 1937–1946. https://doi.org/10.1145/2207676.2208337

[21] Jane Hoffswell, Wilmot Li, and Zhicheng Liu. 2020. Techniques for Flexible Responsive Visualization Design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376777

[22] Enamul Hoque and Maneesh Agrawala. 2020. Searching the Visual Style and Structure of D3 Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 26 (2020), 1236–1245.

[23] Nathan Hurst, Wilmot Li, and Kim Marriott. 2009. Review of Automatic Document Formatting. In *Proceedings of the 9th ACM Symposium on Document Engineering* (Munich, Germany) *(DocEng '09)*. Association for Computing Machinery, New York, NY, USA, 99–108. https://doi.org/10.1145/1600193.1600217

[24] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 1125–1134.

[25] Jennifer Jacobs, Joel Brandt, Radomír Mech, and Mitchel Resnick. 2018. Extending manual drawing practices with artist-centric programming tools. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13.

[26] Jennifer Jacobs, Sumit Gogia, Radomír Měch, and Joel R Brandt. 2017. Supporting expressive procedural art creation through direct manipulation. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 6330–6341.

[27] Ajay Jain, Amber Xie, and Pieter Abbeel. 2023. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 1911–1920.

[28] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, IEEE Computer Society, Los Alamitos, CA, USA, 694–711.

[29] Victor Kaptelinin and Bonnie Nardi. 2012. Affordances in HCI: Toward a Mediated Action Perspective. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 967–976. https://doi.org/10.1145/2207676.2208541

[30] Kotaro Kikuchi, Mayu Otani, Kota Yamaguchi, and Edgar Simo-Serra. 2021. Modeling Visual Containment for Web Page Layout Optimization. In *Computer Graphics Forum*, Vol. 40-7. Wiley Online Library, Wiley Online Library, New York, NY, USA, 33–44.

[31] Janin Koch, Magda Laszlo, Andres Lucero, and Antti Oulasvirta. 2018. Surfing for Inspiration: digital inspirational material in design practice. In *Design Research Society International Conference*. Design Research Society, DRS, New York, NY, USA, 1247–1260.

[32] Dmytro Kotovenko, Matthias Wright, Arthur Heimbrecht, and Björn Ommer. 2021. Rethinking Style Transfer: From Pixels to Parameterized Brushstrokes, In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 1, 12191–12200. https://doi.org/10.1109/CVPR46437.2021.01202

[33] Ranjitha Kumar, Arvind Satyanarayan, Cesar Torres, Maxine Lim, Salman Ahmad, Scott R. Klemmer, and Jerry O. Talton. 2013. Webzeitgeist: Design Mining the Web. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 3083–3092. https://doi.org/10.1145/2470654.2466420

[34] Ranjitha Kumar, Jerry O. Talton, Salman Ahmad, and Scott R. Klemmer. 2011. Bricolage: Example-Based Retargeting for Web Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2197–2206. https://doi.org/10.1145/1978942.1979262

[35] Jane L., Kevin Y. Zhai, Jose Echevarria, Ohad Fried, Pat Hanrahan, and James A. Landay. 2021. Dynamic Guidance for Decluttering Photographic Compositions. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 359–371. https://doi.org/10.1145/3472749.3474755

[36] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R. Klemmer. 2010. Designing with Interactive Example Galleries. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia,

USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 2257–2266. https://doi.org/10.1145/1753326.1753667

[37] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R Klemmer. 2010. Designing with interactive example galleries. In *Proceedings of the SIGCHI conference on human factors in computing systems*. Association for Computing Machinery, New York, NY, USA, 2257–2266.

[38] Gang Li and Yang Li. 2023. Spotlight: Mobile UI Understanding using Vision-Language Models with a Focus. arXiv:2209.14927 [cs.CV]

[39] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. 2020. Differentiable Vector Graphics Rasterization for Editing and Learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 39, 6 (2020), 193:1–193:15.

[40] Raphael Gontijo Lopes, David R Ha, Douglas Eck, and Jonathon Shlens. 2019. A Learned Representation for Scalable Vector Graphics. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* 1, 1 (2019), 7929–7938.

[41] Mikaël Mayer, Gustavo Soares, Maxim Grechkin, Vu Le, Mark Marron, Oleksandr Polozov, Rishabh Singh, Benjamin Zorn, and Sumit Gulwani. 2015. User Interaction Models for Disambiguation in Programming by Example. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 291–301. https://doi.org/10.1145/2807442.2807459

[42] MDN contributors. 2023. <use> - SVG: Scalable Vector Graphics | MDN. https://developer.mozilla.org/en-US/docs/Web/SVG/Element/use. [Online; accessed 6-July-2023].

[43] Kevin Mullet and Darrell Sano. 1996. Designing visual interfaces. *ACM SIGCHI Bulletin* 28, 2 (1996), 82–83.

[44] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. Association for Computing Machinery, New York, NY, USA, 1221–1224.

[45] Srishti Palani, David Ledo, George Fitzmaurice, and Fraser Anderson. 2022. "I Don't Want to Feel like I'm Working in a 1960s Factory": The Practitioner Perspective on Creativity Support Tool Adoption. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 379, 18 pages. https://doi.org/10.1145/3491102.3501933

[46] Oleksandr Polozov and Sumit Gulwani. 2015. FlashMeta: A Framework for Inductive Program Synthesis. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications* (Pittsburgh, PA, USA) *(OOPSLA 2015)*. Association for Computing Machinery, New York, NY, USA, 107–126. https://doi.org/10.1145/2814270.2814310

[47] Jonathan Ragan-Kelley, Charlie Kilpatrick, Brian W Smith, Doug Epps, Paul Green, Christophe Hery, and Frédo Durand. 2007. The lightspeed automatic interactive lighting preview system. In *ACM SIGGRAPH 2007 papers*. Association for Computing Machinery, New York, NY, USA, 25–es.

[48] Pradyumna Reddy, Paul Guerrero, Matt Fisher, Wilmot Li, and Niloy Jyoti Mitra. 2020. Discovering pattern structure using differentiable compositing. *ACM Transactions on Graphics (TOG)* 39 (2020), 1–15.

[49] Quentin Roy, Futian Zhang, and Daniel Vogel. 2019. Automation Accuracy Is Good, but High Controllability May Be Better. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3290605.3300750

[50] Quentin Roy, Futian Zhang, and Daniel Vogel. 2019. Automation Accuracy Is Good, but High Controllability May Be Better. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3290605.3300750

[51] Othman Sbai, Camille Couprie, and Mathieu Aubry. 2018. Vector Image Generation by Learning Parametric Layer Decomposition. *ArXiv* abs/1812.05484 (2018).

[52] Eldon Schoop, Xin Zhou, Gang Li, Zhourong Chen, Bjoern Hartmann, and Yang Li. 2022. Predicting and Explaining Mobile UI Tappability with Vision Modeling and Saliency Analysis. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 36, 21 pages. https://doi.org/10.1145/3491102.3517497

[53] Chanon Seel-audom, Wassana Naiyapo, and Varin Chouvatut. 2017. A search for geometric-shape objects in a vector image: Scalable Vector Graphics (SVG) file format. *2017 9th International Conference on Knowledge and Smart Technology (KST)* 1, 1 (2017), 305–310.

[54] Hijung V. Shin, Jeremy Warner, Björn Hartmann, Celso Gomes, Holger Winnemöller, and Wilmot Li. 2021. Multi-level Correspondence via Graph Kernels for Editing Vector Graphics Designs. In *Proceedings of Graphics Interface 2021 (GI 2021)*. Canadian Information Processing Society, Virtual Event, 97 – 107. https://doi.org/10.20380/GI2021.12

[55] Minhyang (Mia) Suh, Emily Youngblom, Michael Terry, and Carrie J Cai. 2021. AI as Social Glue: Uncovering the Roles of Deep Generative AI during Social Music Composition. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 582, 11 pages. https://doi.org/10.1145/3411764.3445219

[56] Amanda Swearngin and Yang Li. 2019. Modeling mobile interface tappability using crowdsourcing and deep learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–11.

[57] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah Goodman, and Radomír Měch. 2012. Learning Design Patterns with Bayesian Grammar Induction. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 63–74. https://doi.org/10.1145/2380116.2380127

[58] Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. 2018. Face2Face: Real-Time Face Capture and Reenactment of RGB Videos. *Commun. ACM* 62, 1 (dec 2018), 96–104. https://doi.org/10.1145/3292039

[59] Josh Urban Davis, Fraser Anderson, Merten Stroetzel, Tovi Grossman, and George Fitzmaurice. 2021. Designing Co-Creative AI for Virtual Environments. In *Creativity and Cognition* (Virtual Event, Italy) *(C&C '21)*. Association for Computing Machinery, New York, NY, USA, Article 26, 11 pages. https://doi.org/10.1145/3450741.3465260

[60] Mariya I. Vasileva, Bryan A. Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Alexander Forsyth. 2018. Learning Type-Aware Embeddings for Fashion Compatibility. *ArXiv* abs/1803.09196 (2018).

[61] Jo Vermeulen, Kris Luyten, Elise van den Hoven, and Karin Coninx. 2013. Crossing the Bridge over Norman's Gulf of Execution: Revealing Feedforward's True Identity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 1931–1940. https://doi.org/10.1145/2470654.2466255

[62] W3C SVG Working Group. 2018. Scalable Vector Graphics (SVG) 2. https://www.w3.org/TR/SVG2/. [Online; accessed 6-July-2023].

[63] Chenglong Wang, Yu Feng, Rastislav Bodik, Isil Dillig, Alvin Cheung, and Amy J Ko. 2021. Falx: Synthesis-Powered Visualization Authoring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 106, 15 pages. https://doi.org/10.1145/3411764.3445249

[64] Jason Wu, Xiaoyi Zhang, Jeff Nichols, and Jeffrey P Bigham. 2021. Screen Parsing: Towards Reverse Engineering of UI Models from Screenshots. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 470–483. https://doi.org/10.1145/3472749.3474763

[65] Haijun Xia, Bruno Araujo, Tovi Grossman, and Daniel Wigdor. 2016. Object-Oriented Drawing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4610–4621. https://doi.org/10.1145/2858036.2858075

[66] Haijun Xia, Nathalie Henry Riche, Fanny Chevalier, Bruno De Araujo, and Daniel Wigdor. 2018. DataInk: Direct and Creative Data-Oriented Drawing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173797

[67] Katherine Q. Ye, Wode Ni, Max Krieger, Dor Ma'ayan, Jenna Wise, Jonathan Aldrich, Joshua Sunshine, and Keenan Crane. 2020. Penrose: from mathematical notation to beautiful diagrams. *ACM Trans. Graph.* 39 (2020), 144.

[68] Nur Yildirim, Alex Kass, Teresa Tung, Connor Upton, Donnacha Costello, Robert Giusti, Sinem Lacin, Sara Lovic, James M O'Neill, Rudi O'Reilly Meehan, Eoin Ó Loideáin, Azzurra Pini, Medb Corcoran, Jeremiah Hayes, Diarmuid J Cahalane, Gaurav Shivhare, Luigi Castoro, Giovanni Caruso, Changhoon Oh, James McCann, Jodi Forlizzi, and John Zimmerman. 2022. 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* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 483, 13 pages. https://doi.org/10.1145/3491102.3517491

[69] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. 2021. Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 275, 15 pages. https://doi.org/10.1145/3411764.3445186

[70] Nanxuan Zhao, Quanlong Zheng, Jing Liao, Ying Cao, Hanspeter Pfister, and Rynson W. H. Lau. 2021. Selective Region-Based Photo Color Adjustment for Graphic Designs. *ACM Trans. Graph.* 40, 2, Article 17 (apr 2021), 16 pages. https://doi.org/10.1145/3447647

A METHODOLOGY DETAILS

Style Transfer Evaluation – Four designers used Figma to generate their initial designs they brought into the study, while the other two designers used Adobe Illustrator. After designers responded to the prompt, we hosted an hour-long Zoom session with each designer. We instrumented the interface to log relevant events with timestamps (e.g., loading, saving, editing). We sought to gather rich commentary and reflection from designers as they engaged with the prototype. We invited designers to verbally share any thoughts on their experience and highlight any surprising interactions throughout the study. While we recorded usage times per example, designers were not told this nor instructed to be as efficient as possible. The designers moved on only after indicating satisfaction with the relative appearance of their stylized Output graphics. Finally, designers answered a brief survey about their experience using VST, including Likert-scale (Fig. 8) and open-ended questions. Designers sent all styled designs and an interface usage log to the authors and received a $30 Amazon gift card.

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

B PARTICIPANT BACKGROUNDS

Style Transfer Evaluation (D1-6) – We recruited designers via design-oriented email lists at a large research university. Designers included undergraduates (4), Ph.D. students (1), and design professionals (1). Each participating student had completed multiple design internships, bolstering their relevant experience. Their preferred tools included Figma, Adobe Illustrator, and Canva. They had an average of 4.7 years of design experience (2–10 years).

Design Replication Evaluation (RD1-4) – We recruited from the same design community as before, now selecting only the most experienced designers. All RDs had professionally worked as designers. One was the instructor for a university course teaching students how to use Illustrator, and another held a residence in a design lab guiding student projects. These designers had, on average, 6.5 years of design experience and used Illustrator daily. None of these expert designers participated in the original study.

C TRANSFERRABLE SVG ATTRIBUTES

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

D CUSTOMIZATION UI TECHNIQUES

Also see our online demo: https://berkeleyhci.github.io/vst/

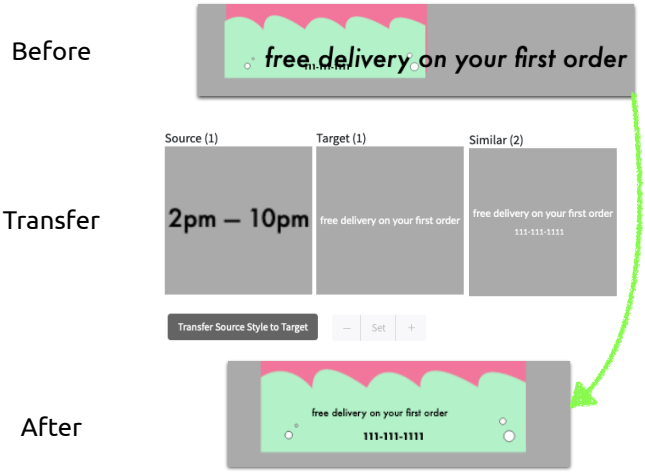

Figure 10: The Customization UI shows the Source and Target selections and similar Target elements. The similarity controls [-/set/+] can adjust the selection to the desired Target elements. Once satisfied with the Source-Target mapping, pressing *Transfer Source Style* will transfer all styles from the Source elements to the active Target selection.

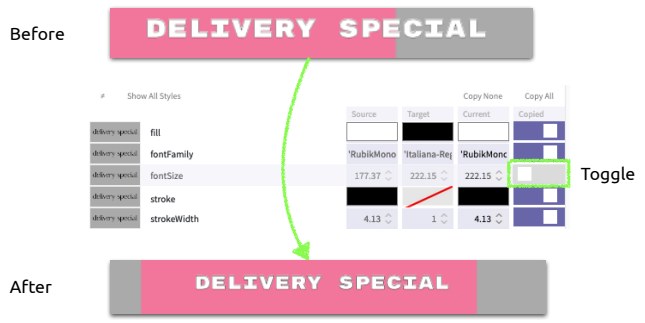

Figure 11: The Customization UI also provides fine-grained control over which styles to transfer. Element style attributes can be copied, reset, or customized for each set of similar values. This list can be filtered only to show styles for the current selection and only to show modified attributes. The UI also features the *Copy All* and *Copy None* buttons – *Copy All* blindly copies all styles for every matched element (e.g., the fully automatic output), and *Copy None* restores the Output graphics to the original Target state.

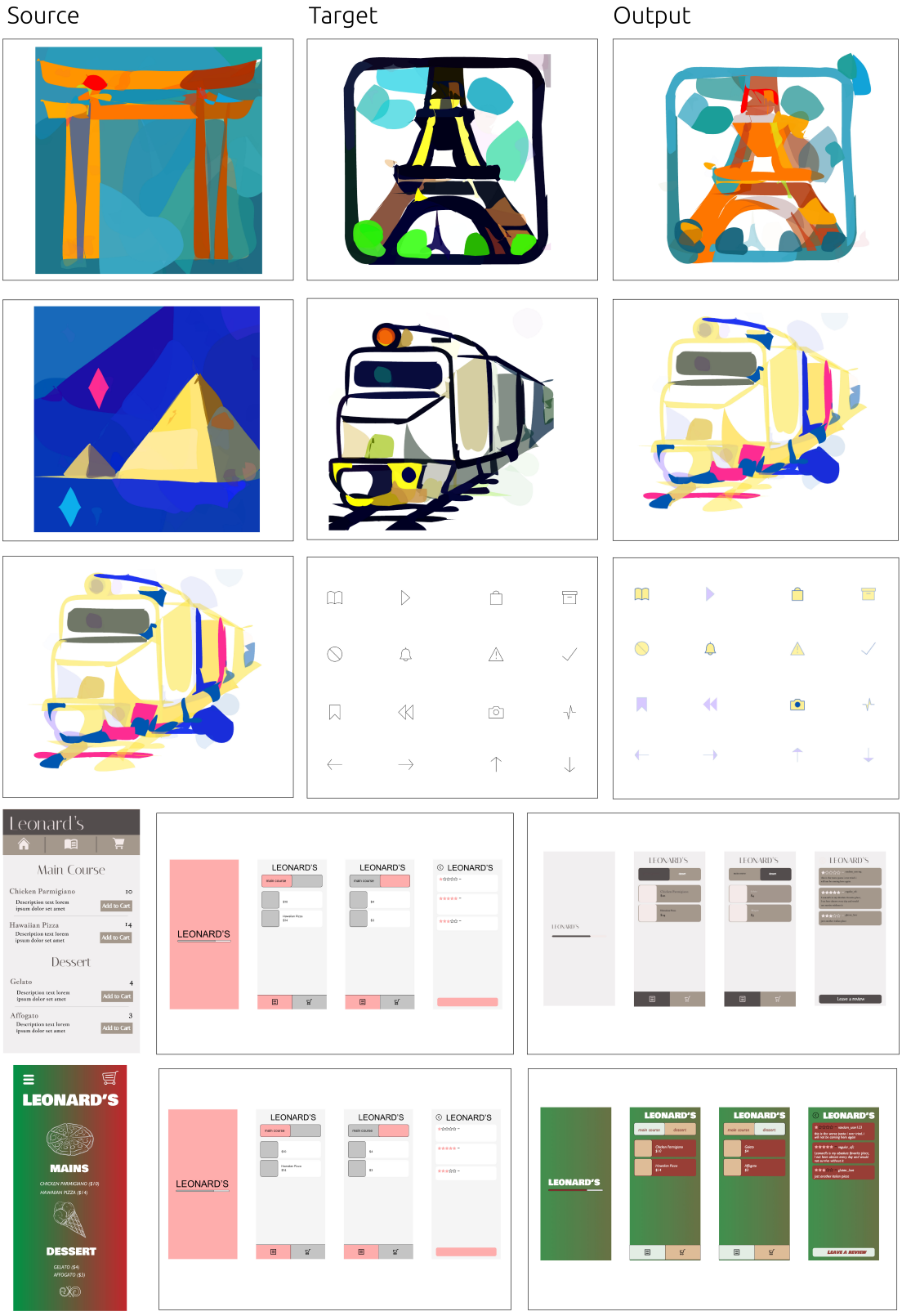

Figure 12: Additional graphics generated by transferring styles with VST.