FONTCRAFT: Multimodal Font Design Using Interactive Bayesian Optimization

Yuki Tatsukawa
The University of Tokyo
Japan

I-Chao Shen
The University of Tokyo
Japan

Mustafa Doga Dogan
Adobe Research
Switzerland

Anran Qi
Centre Inria d'Université Côte d'Azur
France

Yuki Koyama
National Institute of Advanced
Industrial Science and Technology
Japan

Ariel Shamir
Reichman University
Israel
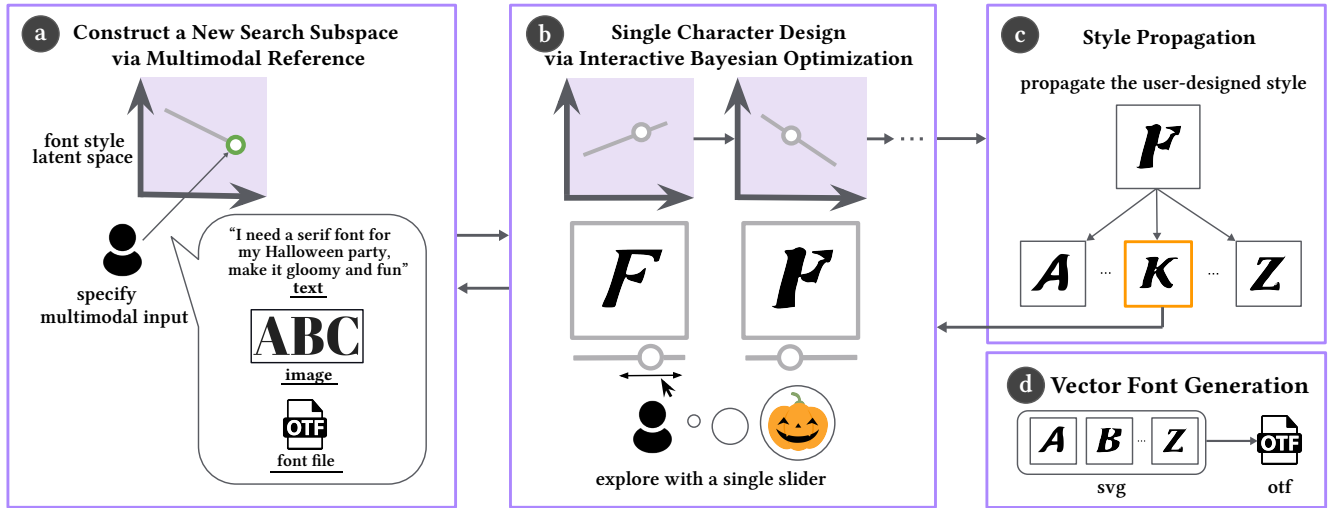
Takeo Igarashi
The University of Tokyo
Japan

Figure 1: FONTCRAFT allows non-expert users to create a font without pre-designed characters through four key steps. (a) Users input multimodal data (text, images, font files) to construct a new search subspace. (b) Users repeatedly explore the search subspace recommended by Bayesian optimization or constructed by multimodal reference using a slider. (c) Users can propagate an edited character's style to the remaining characters and refine any unsatisfactory characters (e.g., "K") by repeating tasks (a) and (b). (d) The system generates *OpenType Font* (OTF) file.

ABSTRACT

Creating new fonts requires a lot of human effort and professional typographic knowledge. Despite the rapid advancements of automatic font generation models, existing methods require users to prepare pre-designed characters with target styles using font-editing software, which poses a problem for non-expert users. To address this limitation, we propose FONTCRAFT, a system that enables font generation without relying on pre-designed characters. Our approach integrates the exploration of a font-style latent space with human-in-the-loop preferential Bayesian optimization and multimodal references, facilitating efficient exploration and enhancing user control. Moreover, FONTCRAFT allows users to revisit previous designs, retracting their earlier choices in the preferential Bayesian optimization process. Once users finish editing the style of a selected character, they can propagate it to the remaining characters and further refine them as needed. The system then generates a complete outline font in OpenType format. We evaluated the effectiveness of FONTCRAFT through a user study comparing it to

a baseline interface. Results from both quantitative and qualitative evaluations demonstrate that FONTCRAFT enables non-expert users to design fonts efficiently.

CCS CONCEPTS

• **Human-centered computing** → *Human computer interaction (HCI)*.

KEYWORDS

font design, outline fonts, human-in-the-loop, latent space exploration, novice user support tools, generative models, typography tools

1 INTRODUCTION

Designing a new font for posters, websites, and advertisement banners is a challenging task, even for professional designers. It requires a significant amount of repetitive manual effort because the designers need to create a whole set of characters. For example, a Roman font contains 62 characters including "A–Z", "a–z", and "0–9". Moreover, when it comes to other writing systems, for example, it takes about 12 months for three to five experts to design a *GB18030-2000* Chinese font comprising 27,533 characters, according to *FounderType*[1], a Chinese font company. Additionally, font design necessitates adherence to typography-specific criteria to ensure that fonts maintain consistency, meaning all characters share a uniform style.

To reduce the manual effort required for designing fonts, many previous methods have leveraged generative models to generate accurate and diverse fonts [13, 19, 20, 29–32, 34–37, 40, 41]. These methods formulate font generation as a *style transfer* problem: The style of predesigned character examples is transferred to the target characters while preserving their structure. Although these methods generate high-quality fonts, they still have a limitation that hinders their usefulness for non-expert users in designing new fonts. Specifically, they require users to create character examples in desired font styles using font-editing software, which poses a challenge for non-expert users. For instance, *DualVector* [20], one of the latest Roman font generation models, requires 3–5 predesigned character examples.

To tackle the challenges of font creation, we propose FONTCRAFT, a novel system that allows users to create new fonts ***without needing to prepare specific character examples***, making it especially user-friendly for non-experts. As shown in Figure 1, users can create desired fonts containing numerous characters by iteratively adjusting a slider and providing multimodal references. Our system allows them to explore within a subspace of the font style latent space of a font generative model. The process begins with users exploring the style for a single character, such as "A" or "z," and

then propagating a selected style to the remaining characters. Once the style is propagated, users can choose an unsatisfactory character (if any) and refine it using the same design procedure as they did with the initial character. By repeating this process, users can ultimately design a font that meets their preferences. Users then obtain a complete outline font in OpenType format.

The core of our system is a novel latent space exploration method that combines human-in-the-loop preferential Bayesian optimization (PBO) and multimodal references. Our method has two main technical contributions: ***multimodal-guided subspace*** and ***retractable preference modeling***, which addresses two key limitations in existing human-in-the-loop PBO.

Human-in-the-loop PBO has been widely used to obtain the optimal solution of user preference function for visual design parameter adjustment [17], photographic lighting design [39], and exploring generative images and melodies [7, 42]. Similarly, in our system, users explore the font style latent space of a font generation model by selecting their preferred styles from candidates recommended through Bayesian optimization. However, relying solely on PBO in the design process can diminish users' sense of agency, creativity, and ownership [6]. To address this issue, we propose to construct multimodal-guided subspaces, which enables users to directly convey their preference to the PBO process using texts and images. Specifically, we map user-provided multimodal references to points in the search subspace by encoding fonts that are similar to these references from an existing font database [22]. To retrieve these similar fonts, our method leverages FontCLIP [28], a typography visual-language model, and constructs a new search subspace that incorporates the encoded points from the retrieved fonts. By combining the multimodal-guided subspace with the subspace generated by the previous Bayesian optimization method [17], our approach enables users to design their desired fonts more efficiently.

Additionally, previous PBO methods assume that users' preferences remain consistent throughout the design process [16]. As a result, users cannot retract their preferences during the font design process. To overcome this limitation, we introduce a history interface that supports retractable preference modeling. This interface allows users to review their design history, revisit earlier states, and restart from a specified past design. This feature is particularly valuable when users change their preferences during the design process, freeing them from the limitations of an irretractable workflow.

Furthermore, we introduce a style propagation and refinement feature, enabling users to achieve consistent styling across all characters easily. Once users design a character with the desired style, they can propagate that style to all other characters. They can then fine-tune any characters that require additional adjustments until satisfied. This feature not only simplifies the design process but also ensures consistent styling throughout the entire font set.

To the best of our knowledge, FONTCRAFT is the first system that enables font design utilizing efficient font style exploration without requiring pre-designed character examples, significantly lowering the barrier to font design. To accomplish this, our system integrates a human-in-the-loop Bayesian optimization method utilizing multimodal input with well-organized features such as history interface and style propagation.

---

[1]https://www.foundertype.com/

We conducted a user study to evaluate how efficiently non-expert users could design fonts using our system, both quantitatively and qualitatively. The study compared our system to a baseline system that solely relied on a single slider with basic PBO. The study aimed to verify the advantages of key features in our proposed system, including *the integration of PBO with **multimodal input*** and features such as *history interface for retractable preference modeling* and *the combination of style propagation and refinement*. In the user study, participants without font design experience were tasked with creating fonts using both our system and the baseline. We also collected feedback from the participants to assess their satisfaction with our system. Both quantitative and qualitative analyses of the fonts created by participants revealed that our system produced better font designs compared to the baseline. Survey responses also confirmed that the proposed system features significantly enhanced the efficiency of the font design process. Additionally, we demonstrated that our system ***supports other writing systems***, such as Chinese, Japanese, and Korean (CJK), and is effective in practical applications, including logo and advertisement design. These findings show that our system is not only applicable to non-Roman font design but also useful in practical, real-world design scenarios.

Contributions. To sum up, we make the following contributions:

- We present FONTCRAFT, an interactive font design system that simplifies the process of creating fonts across various writing systems, making the design process accessible even to non-experts.
- We propose a method that combines human-in-the-loop Bayesian optimization and multimodal references , enabling users to explore and interact with the multimodal-guided subspace.
- We introduce a history interface that allows users to retract and update their preferences during the design process, which cannot be done in previous human-in-the-loop PBO methods.
- We develop an iterative style propagation and refinement method to ensure consistent style throughout the entire font set.

2 RELATED WORK

2.1 Automatic Font Generation

Font generation aims to create characters with a specific font style, ultimately leading to the creation of new font libraries. Researchers have proposed various methods for generating bitmap Roman fonts, such as blending styles from template fonts [27], constructing a font manifold [4], and manipulating attribute scores [33]. Additionally, recent studies have focused on synthesizing outline fonts in vector format using deep generative networks [20, 29, 34–36]. These works tackle the challenges by representing character outlines as sequences of tokens [34, 35] or signed distance functions (SDFs) [20, 36]. While these approaches successfully synthesize vector fonts, non-expert users may find it difficult to use them directly, as they require pre-designed characters in target fonts or manipulating various kinds of attribute scores [33].

On the other hand, creating Chinese, Japanese, and Korean (CJK) fonts, which consist of a vast number of complex characters, requires different approaches than generating Roman fonts. Some methods attempt to generate CJK fonts by utilizing extracted metadata, such as radicals and strokes [18, 38, 42, 44]. However, these approaches face significant challenges, particularly the need for a large number of character examples. For example, generating a font with 2,550 characters requires 522 character examples in the desired font style [18].

To overcome these problems, recent deep learning-based works [5, 13, 23, 26, 31] treat the font generation problem as a style transfer problem. However, these methods require labeled data, such as radicals of characters. In contrast, several approaches aim to train font-generation models without relying on domain knowledge [13, 19, 37, 40, 41]. Among them, *DG-Font* [37] combines style and content using adaptive instance normalization (*AdaIN* [12]), a straightforward yet effective style transfer technique that aligns the mean and variance of content with those of style. This method requires only a few character examples in the desired font. More recently, diffusion model-based methods [11] have achieved high-quality and high-resolution font generation [8, 10, 19, 40].

In this paper, we utilize the extended *DG-Font* to generate fonts without the need to prepare character examples. Although *DG-Font* is not the latest model, its latent space is easier to explore than the latent space of diffusion-based font-generative models [10, 19, 40]. Notably, our proposed system is compatible with other pretrained font generative models that utilize a font style latent space.

2.2 Human-in-the-Loop Bayesian Optimization

Bayesian optimization [2, 25] is a widely used method for optimizing black-box functions. It is particularly useful for functions that are expensive to evaluate because it aims to find the optimal value with minimal iteration, which is achieved by selecting queries that are most effective in terms of exploration and exploitation.

To reduce the number of expensive human evaluations, researchers have tried to integrate Bayesian optimization with human-in-the-loop systems [1, 3, 7, 14, 16, 17, 21, 43]. For instance, Koyama *et al.* [17] propose a method called Sequential Line Search (SLS), which finds the optimal value in the multi-dimensional space by tweaking a one-dimensional slider that is easy for humans to perform. The line explored by the one-dimensional slider connects the point expected to be optimal and the point at which the acquisition function is maximized. Building on SLS, Zhou *et al.* [43] propose a framework for generating melody compositions. Their framework transforms the task of adjusting a one-dimensional slider into selecting the most favorable candidate from a set of options. This adaptation enhanced user interaction while leveraging the strengths of Bayesian optimization. Kadner *et al.* [14] introduce a human-in-the-loop system for font generation, focusing on optimizing fonts for readability through Bayesian optimization. In contrast, our work expands the scope of font design by integrating SLS with multimodal references, simplifying the creation of fonts for a variety of applications beyond readability.

A notable limitation of human-in-the-loop Bayesian optimization is that it tends to reduce user agency in the design process and decrease their sense of ownership over the outcomes [6]. Chan *et al.* [6] suggest that enhancing users' ability to express their ideas to the optimizer can effectively improve both agency and ownership. Previous works have addressed this issue by enabling users to directly incorporate preferences in various approaches, such as specify areas in the design space they wish to exclude [21], edit the generated

melody [43] and images [7] directly. While these approaches allow users to incorporate their preferences directly into the Bayesian optimization process, they assume that users' preferences are time-invariant [16] and restrict users to a forward-directed design work-flow, limiting flexibility in revisiting or re-evaluating earlier steps.

In contrast, our proposed method enables users to incorporate their preferences into the Bayesian optimization process using multimodal references, including text input. This multimodal inter-active capability is a novel improvement over previous methods. Additionally, we provide a user interface that effectively visualizes the interaction history between the user and the system. This visualization allows users to easily understand the design history and revisit or re-evaluate earlier points, freeing them from the constraints of a strictly forward-directed optimization process.

3 FONTCRAFT SYSTEM OVERVIEW

3.1 System Architecture

The overall architecture of FONTCRAFT, an effective system for font design, consists of two main components: the user interface (UI) and the font generative model. Users use UI to explore the font-style latent space and select desired font styles. The font generative model generates the bitmap representation of characters from latent variables (see Section 4.1).

3.2 User Interface

The UI is designed to be simple and user-friendly, particularly for users who have never designed fonts before. As shown in Figure 2, our UI contains the character design area, the multimodal input area, the character collection area, and the history area. Users interact with the system by adjusting a slider or providing multimodal data, and they can check real-time previews of generated fonts. This interactive feedback loop allows users to iteratively refine their font choices based on visual aesthetics. Each of these elements is crucial in facilitating the font design process. We introduce them in this subsection.

3.2.1 Character Design Area. The character design area is the most frequently used element for exploration within the UI. It consists of a single slider, an image viewer, and three control buttons. The slider enables users to explore the one-dimensional latent subspace determined by either Bayesian optimization or multimodal reference inputs (see Section 4.4). The image viewer shows the currently focused character ("A" in Figure 2) in the selected style, rendered in vector format. We generate the vector character in the following steps. First, the bitmap character is generated by the font generative model using the style latent vector selected by the handle on the slider. Then, we reduced artifacts in the generated bitmap character image by simply setting any pixel with a grayscale value above a certain threshold to white. Finally, we converted the filtered bitmap character into SVG format by tracing the outlines using the Potrace algorithm [24].

The three control buttons, RESET, UPDATE, and UPDATE ALL, serve specific functions:

- RESET: to clear any accumulated preferences in Bayesian optimization for a focused character and reinitialize it, allowing

users to start exploring the font style for that character from scratch.
- UPDATE: to submit the selected point on the slider as the current user preference for a focused character, requesting the Bayesian optimization process to recommend a new search subspace for exploration in the next iteration.
- UPDATE ALL: to propagate the style the user prefers for all characters and request the Bayesian optimization process to recommend the next search subspace.

3.2.2 Multimodal Input Area. The multimodal input area allows users to provide multimodal references, including text, images, and existing font files. These references are used to initialize or customize the font style exploration process by constructing a new search subspace. By providing specific references, users can directly influence the system's output, making it easier for them to design desired fonts. We explain how to encode the multimodal references into the font style latent space in Section 4.4.

3.2.3 Character Collection Area. The character collection area displays previews of the generated characters in vector format. Users can zoom in on each character to inspect for any defects and select a specific character to focus on. Once a character is selected, users can refine its font style in the character design area.

3.2.4 History Area. The history area displays a sequence of user inputs and system outputs, allowing users to track their design progress. Users can select any previous output to revert to that stage, which enables them to undo actions and restart the font design process from a specific point. This feature allows users to retract undesired preferences and update their preferences during the design process.

4 METHOD

4.1 Preliminary of Font Generative Model

We use *DG-Font* [37] as our font generative model. This model takes a style image I_S representing the desired font style, and a content image I_C representing the desired character, as input. It then generates the character image that represents the desired character in the desired font style as the output. As illustrated in Figure 3, the architecture of *DG-Font* is an encoder-decoder model with two encoders: a style encoder E_S and a content encoder E_C, along with a content decoder G_C. The generation process starts by extracting the style latent vector from the style image using the style encoder and the content latent vector from the content image. Then, the content decoder takes both the style and content latent vectors as input to generate the desired font image $\hat{I}$, which maintains a similar style to the style image while preserving the structure of the content image. Overall, the generation process during the training process can be formulated as:

$$z_S = E_S(I_S), \quad z_C = E_C(I_C), \quad \hat{I} = G_C(z_S, z_C). \tag{1}$$

In our work, we use an enhanced version of *DG-Font*, which includes an additional content discriminator. Notably, our system (FONTCRAFT) only needs the pretrained model to generate new fonts instead of training a model from the beginning. We will provide the details of this additional model architecture and training process in the supplemental material.
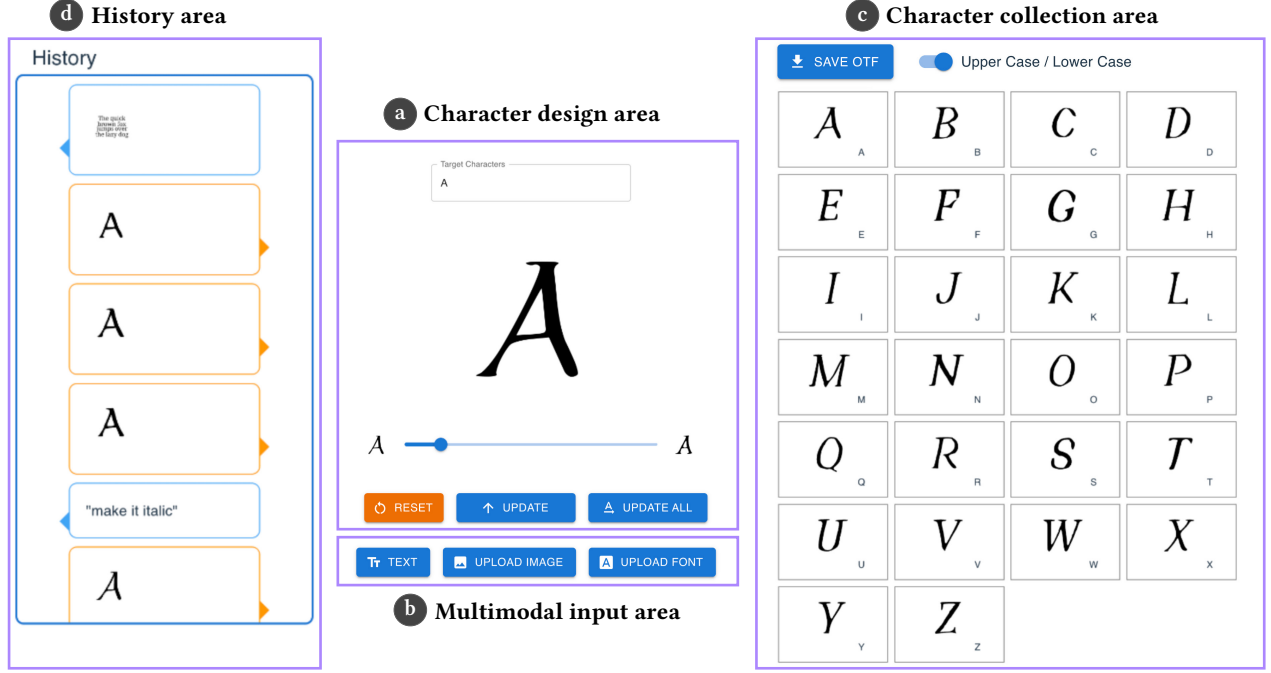
Figure 2: FONTCRAFT UI. Users manipulate the slider in (a) the character design area to explore the line search subspace provided by the system. They can also input multimodal references using (b) the multimodal input area. They can obtain a new recommendation by pressing the UPDATE button. Once users are satisfied with the current style of the focused character, they can propagate its style to all other characters by pressing the UPDATE ALL button, and the results can be viewed in (c) the character collection area. Optionally, users can select another character and further refine it. (d) The history area shows the sequence of user inputs and system outputs, enabling users to easily track their exploration history and revert to a specific checkpoint if needed.

In the rest of the paper, if we need to specify the content image I_C or its latent vector z_C for a specific character such as "A", we will denote it as $I_C[$"A"$]$ $z_C[$"A"$]$. Otherwise, we will use I_C or z_C for abbreviation. This also applied to the generated image $\hat{I}$.
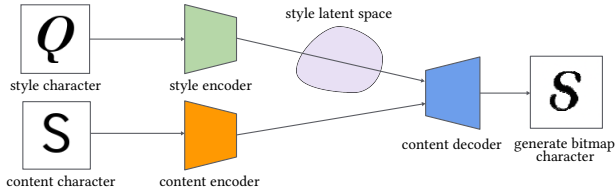
Figure 3: Overview of *DG-Font*. *DG-Font* is an encoder-decoder model that takes a character image representing style and a character image representing content as input, and outputs a character image that combines the content with the specified style. During font designing in our system, users use our human-in-the-loop optimization to explore the style latent space of the style encoder. Please find the detailed architecture of the encoder and decoder in the supplemental material.

4.2 Preliminary of FontCLIP

To incorporate multimodal input when designing fonts, we use FontCLIP [28] to extract typographical features from both text and image input. FontCLIP is a visual-language model that bridges the semantic understanding of a large vision-language model with typographical knowledge. It consists of a text encoder and a visual encoder and builds a joint latent space that encodes typographical knowledge. In this joint latent space, similar font images and text prompts will have similar latent vectors. For example, a bold font will have a similar latent vector to the text prompt "This is a *strong* and *thick* font" compared to the text prompt "This is a *thin* font". Therefore, the FontCLIP latent vector can be used to retrieve similar fonts using text or image input. In our system, we utilize both the FontCLIP text encoder and visual encoder to extract a latent vector from the multimodal input to customize the linear subspace.

4.3 Preliminary of Human-in-the-Loop Bayesian Optimization

4.3.1 Problem Formulation. Human-in-the-loop optimization is a computational approach used to solve parameter optimization problems involving human evaluators in its iterative algorithm. It

is commonly used to support design tasks that involve generating visual content defined by a set of parameters x, with the aim of achieving certain subjective design goals. Specifically, we can formulate such optimization problem as:

$$x^* = \arg\max_{x \in \mathcal{X}} f(x), \tag{2}$$

where $\mathcal{X}$ is the search space, and $f : \mathcal{X} \to \mathbb{R}$ is the goodness function to evaluate a subjective design goal (e.g., the aesthetics of the current design). We aim to find the optimal value x^* with the fewest trials because evaluating $f(\cdot)$ is costly. However, solving Equation 2 using traditional Bayesian optimization (BO) might not be suitable for many design tasks. This is because it is often difficult to assign an exact value to a sample, whereas comparing a couple of samples and choosing the preferred one is more intuitive. For example, it is hard for users to give a score to a font individually, but easier for them to choose the preferred font from a set of candidates. Therefore, in this work, we choose to use preferential Bayesian optimization (PBO) [15], which is a variant of Bayesian optimization (BO) that runs with relative preferential data. In particular, we build our method upon Sequential Line Search (SLS) [17], a PBO method that constructs a sequence of linear subspaces that leads to the optimal parameters that match the user's need.

4.3.2 Sequential Line Search (SLS). With SLS, the user can search for his/her preference by adjusting a slider. At each iteration of the optimization, SLS constructs a linear subspace using the current-best position x^+ and the best-expected-improving position x^{EI}. Suppose we already have t observed response so far, then the next linear subspace $\mathcal{S}_{t+1}$ is constructing by connecting:

$$x_t^{\mathrm{EI}} = \arg\max_{x \in \mathcal{X}} a^{\mathrm{EI}}(x; \mathcal{D}_t) \tag{3}$$

$$x_t^+ = \arg\max_{x \in \{x_i\}_{i=1}^{N_t}} \mu_t(x)$$

where $\{x_i\}_{i=1}^{N_t}$ denotes the set of points observed so far, μ_t and a^{EI} are the predicted mean function and the acquisition function calculated from the current data. We use the expected improvement (EI) criterion as the acquisition function to choose the next sampling point that is likely to optimize the function f and at the same time its evaluation is more informative:

$$a^{\mathrm{EI}}(x; \mathcal{D}) = \mathbb{E}\left[\max\left\{f(x) - f^+, 0\right\}\right]. \tag{4}$$

After the t-th iteration, we suppose that we obtained a set of t single slider responses, which is represented as

$$\mathcal{D}_t = \left\{x_i^{\mathrm{chosen}} > \left\{x_{i-1}^+, x_{i-1}^{\mathrm{EI}}\right\}\right\}_{i=1}^t, \tag{5}$$

where x_i^{chosen} represent the position chosen by the user at the t-th iteration.

Let f_i be the goodness function value at a data point x_i, i.e., $f_i = f(x_i)$, and f be the concatenation of the goodness values of all data points:

$$f = [f_1, f_2, \ldots, f_N]. \tag{6}$$

Under the assumption of Gaussian process (GP) prior on f, we use θ to represent the hyperparameters of the multivariate Gaussian
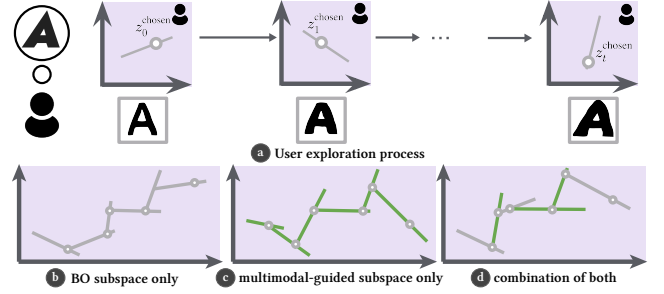
Figure 4: Exploration of the font style latent space using a single slider. (a) Users explore a one-dimensional search subspace within the font style latent space using a single slider. At each iteration, users choose a point in the latent subspace and submit it as their current preference z_t^{chosen}. After a couple of iterations, users gradually converge to their desired font style. The overall exploration process, users can explore (b) BO subspace only, (c) multimodal-guided subspace only, and (d) combination of both.

distribution. Since the goodness values f and the hyperparameters θ are correlated, we infer f and θ jointly by using MAP estimation:

$$(f^{\mathrm{MAP}}, \theta^{\mathrm{MAP}}) = \arg\max_{(f,\theta)} p(f, \theta \mid \mathcal{D})$$

$$= \arg\max_{(f,\theta)} p(\mathcal{D} \mid f, \theta) p(f \mid \theta) p(\theta). \tag{7}$$

Once f^{MAP} and θ^{MAP} have been estimated, we can compute $\mu(\cdot)$, $\sigma(\cdot)$, and $a^{\mathrm{EI}}(\cdot)$ in order to construct the next slider subspace $\mathcal{S}_{t+1}$. We only describe the minimum concept of how SLS constructs the linear subspace to optimize the function f for understanding how we incorporate it in the style latent space of a font generative model and multimodal input. For more details, please refer to the supplemental material.

4.4 Multimodal Bayesian Optimization for Font Generation

Specifically, following Equation 2, the objective function for designing a character can be formulated as:

$$z^* = \arg\max_{z \in \mathcal{Z}} f(G(z)), \tag{8}$$

where $z \in \mathcal{Z}$ is the style latent vector of the font generative model, which is the search space $\mathcal{X}$ in our problem setting. Moreover, G is the decoder of the font generative model, and $f : \mathcal{Z} \to \mathbb{R}$ is the user preference function that measures how good the currently generated character is perceived by the user. To solve Equation 8 and obtain the desired font, users can perform three different actions: *explore the font style latent space, retract previous preferences*, and *propagate style to other characters* at each iteration.

4.4.1 Action 1: Explore the Font Style Latent Space. The primary task for users is to explore the one-dimensional font-style search subspace using a slider. By repeating the slide manipulation, users gradually converge on a point that aligns with their desired font style as illustrated in Figure 4. This subspace is constructed by the

system in two ways: one solely follows the SLS method and another utilizes multimodal references. Note that, regardless of these two different approaches to constructing the subspace, the interaction (i.e., manipulating the slider and submitting a preferred point to the system) remains consistent.

Constructing a SLS subspace. Using the SLS method outlined in Section 4.3.2, our system constructs a linear subspace $\mathcal{S}_t$ by connecting the current best point (z_t^+) and the point that maximizes the acquisition function (z_t^{EI}) using Equation 3 at the t-th iteration. Then, users can choose a style latent vector z using the slider within $\mathcal{S}_t$ and view the generated character $G(z)$. Once satisfied, users submit their preferred point on the slider z_t^{chosen} by clicking the UPDATE or UPDATE ALL button and request the system to construct a new linear subspace $\mathcal{S}_{t+1}$ for the $(t+1)$-th iteration using Equation 7.

Constructing a multimodal-guided subspace. While exploration with a single slider is useful, the linear subspace determined solely by Bayesian optimization sometimes fails to capture user preferences effectively, which leads to an increasing number of iterations and potentially causes frustration and a diminished sense of agency. To address this issue, we allow users to intervene in the linear subspace construction by providing multimodal references at any iteration. At $(t+1)$-th iteration, instead of exploring the linear subspace $\mathcal{S}_{t+1} = (z_t^+, z_t^{\text{EI}})$ constructed solely by Bayesian optimization, the user explores the multimodal-guided subspace: $\mathcal{S}_{t+1}^{mm} = (z_t^+, z_t^{mm})$. Here, z_t^{mm} is the style latent vector obtained by retrieving the most similar fonts to the multimodal reference provided at $(t+1)$-th iteration from a font database containing 1,169 Roman fonts collected by O'Donovan *et al.* [22]. Specifically, we retrieve n fonts and use the mean of their latent vectors as z_t^{mm} (we use $n = 5$ in our implementation). Once the user is satisfied with the current generated character, the slider response: $(z_{t+1}^{\text{chosen}}, z_t^+, z_t^{mm})$ will be recorded in $\mathcal{D}_{t+1}$ (Equation 5) and used for constructing the subspace in the future iteration. This means that all multimodal references provided until iteration t will affect the subspace constructed at $(t+1)$-th iteration. In our current implementation, at each iteration, users can provide only a single multimodal reference, and we construct a new search subspace by connecting the current chosen point and the latent vector of the provided multimodal reference. At 0-th iteration, we construct the initial linear subspace $\mathcal{S}_0 = (z^{\text{init}}, z_0^{mm})$, where z^{init} represents the style latent vector of a commonly used font (we use "IPAex gothic" font in our implementation). Note that the multimodal references are only used to construct the linear subspace for the user to explore, not being directly used to infer the user preferences.r

To construct a multimodal subspace with the user-provided multimodal reference, our system identifies a suitable font in our font database that corresponds to the input text or image. For text input, the system first extracts font attributes such as "formal," "italic," and "happy" using a Large Language Model (LLM). The LLM selects relevant font attributes based on the given text. We utilize 37 types of font attributes compiled by O'Donovan *et al.* [22] (see the supplemental material for more details). Once the font attributes are extracted, the system obtains the feature vector of these text attributes using the text encoder of FontCLIP [28] and retrieves fonts

from our font database based on feature similarity. The retrieved fonts are used as the most suitable options, and the mean of their style latent vectors z^{mm} is obtained using the style encoder E_S.

For the text-rendered image reference, our system retrieves its most similar font in the font database using the FontCLIP feature. Similarly, we then project the retrieved font image into the style latent space using E_S and obtain z^{mm}. Finally, for font file input, our system directly uses the provided font as the suitable font and projects it into the latent space, similar to the process for text and image inputs. We illustrate how to obtain the style latent vector of the multimodal references in Figure 5(c).

4.4.2 Action 2: Retract Previous Preferences.
At each iteration, if the user is not satisfied with the current design and the recommended candidates, they can choose to retract previous preferences. Specifically, at $(t+1)$-th iteration, if the user wishes to retract the last two slider manipulations, then the last two slider responses stored in $\mathcal{D}_t$ will be discarded. Then, if the user opts to explore a new subspace by providing a new multimodal reference, they will then explore the subspace $\mathcal{S}_{t-1}^{mm} = (z_{t-2}^+, z_{t-2}^{mm})$. Otherwise, the user will explore a subspace constructed using SLS solely: $\mathcal{S}_{t-1} = (z_{t-2}^+, z_{t-2}^{\text{EI}})$. By retracting previous preferences, users can update their preferences during the design process.

4.4.3 Action 3: Propagate Style to Other Characters.
Once the user is satisfied with the design of the focused character (e.g., "A") and obtains the style latent vector z_S^*, our system can propagate the style to all other characters and finish designing a single character. Specifically, to propagate the style vector to character "B":

$$\hat{I}[\text{"B"}] = G_C(z_S^*[\text{"A"}], z_C[\text{"B"}]). \qquad (9)$$

Next, users can check all characters with the propagated styles. If they are unsatisfied with the result of another character, they can perform action 1 or action 2 to design that character. If they are satisfied, the resulting font is exported as an outline font.

5 EVALUATION

5.1 Simulated Evaluation of Multimodal Reference

To quantitatively evaluate how multimodal reference can help our human-in-the-loop optimization, we designed a simulation test to compare two linear subspace initialization methods: using multimodal reference and random fonts.

5.1.1 Procedure.
We illustrate the procedure of the simulation test in Figure 6. Given a base font character (e.g., "A"), the goal of the simulation test is to resemble the target font character (Figure 6(d)) by exploring the style latent space through optimization. Specifically, we simulate user selections using the following process. At each iteration, our method selects a point in the slider's search subspace with the minimum perceptual metric (we use *DreamSim* [9]) against the target font character. Then, the selected point is used to request Bayesian optimization to recommend the next linear subspace. We iterate this process to observe the convergence of the optimization progress using both initialization methods.

For initializing using multimodal reference, we test *text input* and *font file input* in this experiment. For the text input, we create
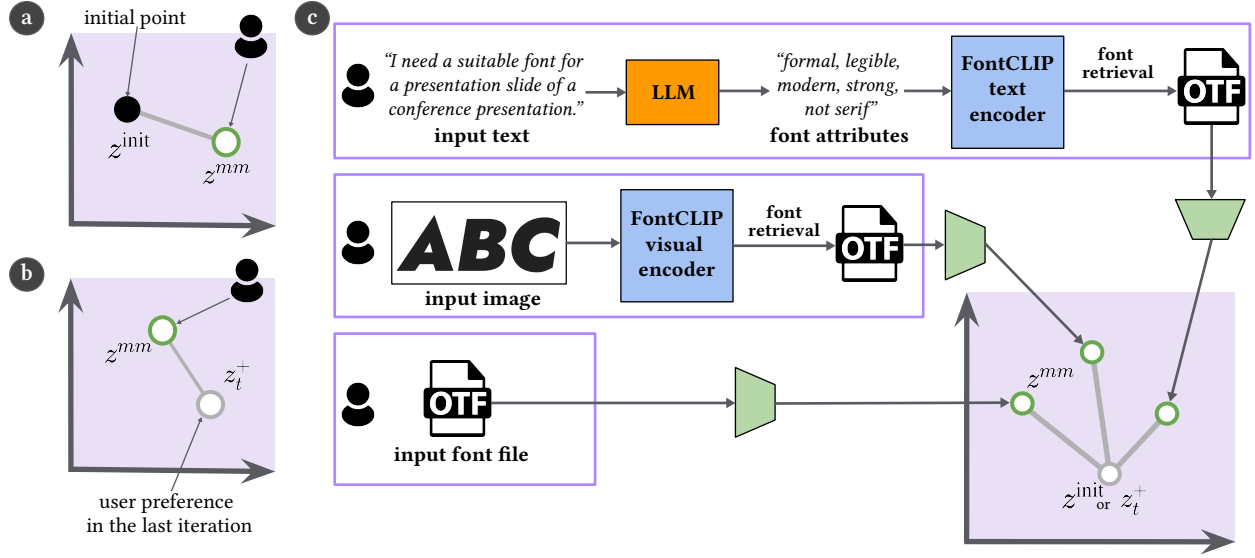
Figure 5: Constructing linear subspaces using multimodal references. (a) At the start of the font design process using our proposed method, the user inputs text, an image, or a font file. The system encodes this input into a font style latent vector and initializes the line search space by connecting the latent vector and a fixed point predetermined by the system. (b) Additionally, the user can introduce multimodal inputs at any stage of the design process. When the user provides new input, the system generates a new line search subspace by connecting the last user preference point with the newly encoded point. (c) Our system encodes multimodal input into the style latent space by leveraging LLM and FontCLIP text and visual encoders.

a descriptive text that characterizes the target font and use it to initialize the search subspace. For font file input, we manually select a font from candidate fonts that closely resembles the target font and use it to initialize the search subspace. Finally, for the baseline method, we choose a font randomly from our font database and use it to construct the initial search subspace.
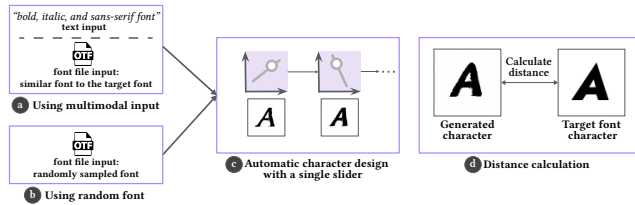
Figure 6: Evaluation of linear subspace initialization methods. We compared two initialization methods for exploration with Bayesian optimization. (a) One method uses input text or a similar font file for initialization, while (b) the other initialize method uses a randomly sampled font from a font database. After initialization, both methods follow the same automatic exploration process (c), where the optimal point on the single linear subspace is repeatedly identified and submitted to the system. In each iteration, we measure the distance between the generated character and the target font character to identify the optimal point, as shown in (d). Note that we use the bitmap format of the character for distance calculation, without vectorizing it.

We conducted this experiment using the character "A" for 10 different target fonts, randomly selected from our font database. We collected 12 kinds of fonts from which we chose a similar font to each target font for the font file input. For each target font, we choose the most similar font out of the 12 candidate fonts. The 12 candidate fonts consist of two popular font families, *Roboto* and *NotoSerif*, and each font family has six variations: *Light*, *Light Italic*, *Regular*, *Regular Italic*, *Bold*, and *Bold Italic*. This selection simulated a scenario where users start with popular fonts and design new fonts based on one of these similar candidates. For each target fonts, the optimization process includes 10 iterations of Bayesian optimization.

5.1.2 Results. In Figure 7, we show the mean and standard deviation of the distances between the optimized results and all target fonts. We can observe that the optimization processes with text and font file references converge to a lower *DreamSim* distance to the target font character compared to those initialized with a randomly selected font. These results indicate that using multimodal references for initializing the human-in-the-loop optimization leads to more effective exploration than random initialization.

5.2 User Study

To evaluate the effectiveness of our proposed system, we conducted a user study in which participants were asked to design fonts using both a baseline system and our system. The goals of this study were threefold:
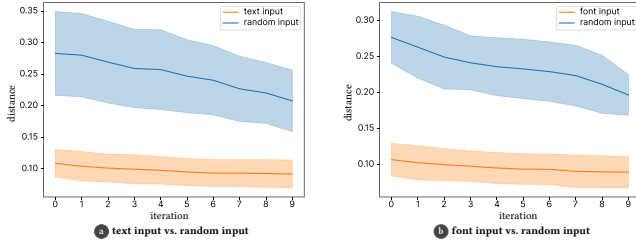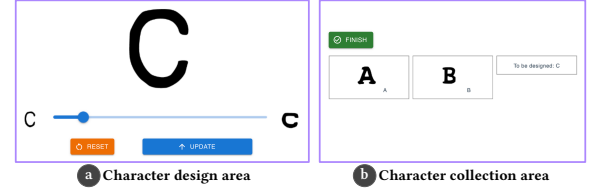
Figure 7: Convergence comparison between two initialization methods. The figure illustrates how the *DreamSim* distance between the designed font character and the target font character converges during exploration with human-in-the-loop optimization. The optimization processes initialized by text font references (orange) obtain better results compared to processes initialized by random font (blue).

Figure 8: User interface of the baseline system. In the (a) character design area, users use a slider to explore the one-dimensional subspace within the font style latent space recommended by Bayesian optimization. By clicking the RESET button, users can reset their preference history in the Bayesian optimization process, randomly reinitializing the search subspace. The users can check the characters that they have already designed are displayed in the (b) character collection area.

- to assess the overall effectiveness of our system, including the integration of Bayesian optimization, multimodal reference, history interface, and style propagation.
- to compare the fonts designed by participants both qualitatively and quantitatively against those created using the baseline system.
- to gather qualitative feedback on the user experience with our system.

5.2.1 Comparison Systems. For the user study, we added a special feature called FONT PALETTE to our proposed system. By clicking the FONT PALETTE button, users can view a visualization of the 12 popular fonts described in Section 5.1 and select one to input as their preference, simplifying the process of inputting a font file. Additionally, we removed the UPLOAD IMAGE and UPLOAD FONT buttons from the UI in Figure 2 for simplicity. As a result, users can now easily input text and font files using the TEXT and FONT PALETTE buttons, respectively.

To assess the effectiveness of the multimodal reference and style propagation features in our system, we created a baseline system that includes only a single slider, as illustrated in Figure 8. In this baseline system, users can explore the font style latent space solely by adjusting the slider, guided by the Bayesian optimization process. Unlike our proposed system, the baseline's one-dimensional search space is initialized by connecting a fixed point with a randomly initialized point. The fixed point corresponds to the style of the *IPAex Gothic* font, as described in Section 5.1. If users encounter difficulties during exploration, they can reset their preference history in the Bayesian optimization process and restart from a newly randomized search subspace. Additionally, this baseline method lacks a style propagation function, requiring users to design each character individually.

5.2.2 Procedure. We recruited ten people for the user study. Each participant was presented with a target font and asked to design three characters, "A", "B", and "C" that closely match the target font using both FONTCRAFT and the baseline system. For this user study, we prepared two target fonts, Font 1 and Font 2. Each font design session continued until one of the following conditions was met: (1) the participant was satisfied with the quality of the

characters they designed, (2) they felt that further improvement was difficult, or (3) the 7-minute time limit was reached. The user study followed this sequence: (Tutorial of FONTCRAFT → Font 1 with FONTCRAFT → Font 2 with FONTCRAFT → Tutorial of baseline → Font 1 with baseline → Font 2 with baseline → Survey). The order of using FONTCRAFT and the baseline system was randomized for each participant. After the font design sessions, participants were asked to complete a questionnaire that validated our system. The entire user study took approximately 60 minutes, with each tutorial lasting 10 minutes, each font design session 7 minutes, and the survey 10 minutes.

5.2.3 Results and Discussion. We compared the designed fonts using our system and the baseline system both quantitatively and qualitatively. For the quantitative evaluation, we calculated the distance between the target font characters and the designed characters in the *DreamSim* latent space. As shown in Table 1(a), the characters designed with our system closely resembled the target font characters compared to those designed with the baseline system. Additionally, we measured the style consistency between all characters designed by each participant by calculating the mean distance between the characters "A", "B", and "C" in the *DreamSim* latent space. As shown in Table 1(b), the distance is smaller when using our system to the baseline system, indicating our system enables more style-consistent character design. In Figure 9, we showed the characters designed by all participants (P1–P10). For Font 1, the "A" characters designed by P2, P3, P4, P6, and P8 using our system closely matched the slanted style of the target "A," while they failed to design the slanted style using the baseline system, indicating that participants effectively captured the italic feature through multimodal reference. On the other hand, characters designed by P1, P2, P3, P4, P5, P8, and P10 using the baseline system showed inconsistencies in style within the same font (e.g., variations in size, height, and weight). In contrast, characters designed with our system exhibited greater consistency, suggesting that style propagation helped create more cohesive designs.

Next, we evaluated participant feedback to validate the effectiveness of our system. We asked questions about the functions in our system, including slider operation, multimodal reference, style
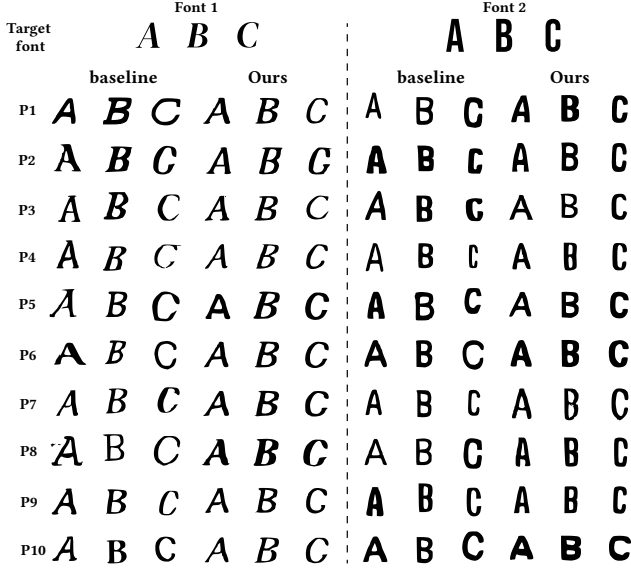
Figure 9: Characters designed by user study participants. Our system enables users to design characters that are more similar to the target font characters and maintain higher consistency between each other. In the case of Font 1, participants successfully designed all characters with the slant style using our system, while some participants failed to create the slant style for "A" using the baseline system. For Font 2, all participants designed characters with consistent styles using our system, whereas the styles of characters designed using the baseline system were inconsistent.

(a) Target font similarity ↓			(b) Designed character consistency ↓		
	Font 1	Font 2		Font 1	Font 2
Baseline	0.1680	0.1416	Baseline	0.3303	0.2893
FONTCRAFT	0.1591	0.1355	FONTCRAFT	0.2983	0.2793

Table 1: (a) We calculated the distance between the characters designed by the participants and the target font characters. Each value represents the mean distance across the 12 characters ("A", "B", "C" designed by the four participants). The characters designed using our system are closer to the ground truth compared to those with the baseline system. (b) We measured the character consistency between the characters "A", "B", and "C" designed by each participant. Each value represents the mean distance across the three characters designed by each participant. The distance among the three characters designed using our system is smaller than that with the baseline system, which indicates our system enables more style-consistent character design. (↓ denotes the lower values are better and we highlight the best result for each target font.)

propagation, and history interface. When we asked the question *"Were you satisfied with the designed characters?"*, seven out of the ten participants answered yes, while P2 and P10 commented neutral, and P9 expressed no. P9 noted that he observed distortions in the generated characters and felt the system was not good at generating straight lines. In response to the question *"Do you think you were able to design fonts easily with the system?"*, all ten participants answered yes, demonstrating the system's effectiveness in enabling non-expert users to design fonts with ease.

In response to the question *"Do you think you were able to effectively use the slider operation for font design?"*, nine participants answered yes. P4, who answered no, expressed dissatisfaction, stating that while the combination of slider manipulation and multimodal reference was effective, using only the slider and repeatedly clicking the UPDATE button sometimes resulted in a linear subspace that excluded the desired character style. P4 emphasized the importance of using multimodal reference at the right moments to avoid unsatisfactory suggestions and stated that relying solely on the slider was not effective. P4 also highlighted that the history interface was useful for reverting to a previous point, leading to the escape of an undesirable search subspace suggested by the system. P4's feedback reflects the findings suggested in Chan *et al.* [6], which indicate that designers working with BO may experience a loss of agency. In contrast, our method provides users with a way to contribute

concrete ideas that guide the BO process, thereby helping them regain a sense of agency.

In response to the question, *"Do you think you were able to effectively use text input?"*, eight of ten participants answered yes. P1, P2, P4, P6, P7, P9, and P10 found text input helpful for making broad changes, such as adjusting weight or slant, but not for fine-tuning details or specifying complicated characteristics. Additionally, P1, P6, and P7 mentioned that understanding typographical terms like "bold" and "italic" was necessary. This feedback indicates that while text input is useful for exploring rough font styles, it has limitations in designing font details and requires some typographic knowledge.

Regarding the question, *"Do you think you were able to effectively use the similar fonts provided by font palette?"*, eight of ten answered yes. P4 and P7 commented that the font palette is particularly helpful when it is difficult to describe the desired font style in texts. P8, P9, and P10 stated that initializing the search subspace using the font palette function allowed them to begin the design task more smoothly compared to the baseline system. However, P5 expressed dissatisfaction, stating that the style of the character generated did not perfectly align with the font they selected from the font palette. This discrepancy, caused by the encoding-decoding process of the font generative model, could lead to confusion among users. To address this issue, it is important to communicate to users that the generated characters may not always perfectly match the multimodal reference. Additionally, we anticipate that newer font generative models could help mitigate this discrepancy. It is worth emphasizing that our proposed system is compatible with any font generative model, provided an efficient font style latent space can be established within it. On the other hand, P2 explained that he did not use the font palette because he preferred to describe the target font style using text input. This feedback suggests that using similar font files and text input complement each other.

When asked, *"Do you think you were able to effectively use the* Update all *button?"*, nine participants responded positively, with eight participants noting that it was more convenient than designing each character individually. P10, who answered no, expressed dissatisfaction, commenting that it would be more convenient if users could toggle between adjusting either all characters at once or individually. In particular, he felt that having a feature to switch to individual adjustments is crucial during the fine-tuning stage. We focus on the simplicity of the UI in this user study and this individual adjustments function is effective especially when designing many characters like all Roman characters.

In response to the question, "*Do you feel that you could design characters with a sense of agency using our system?*", posed only to P5–P10, all six participants responded affirmatively. P7 and P10 noted that in the baseline system, the line search space was initialized randomly, making the process feel highly dependent on luck. In contrast, they appreciated that our proposed system allowed them to control the initialization by specifying their preferences through multimodal references. This insight aligns with the findings in Section 5.1, which show the initialization using multimodal references leads to better results compared to random initialization. Additionally, P9 commented that he felt he could convey his intentions to the system by inputting texts. These insights indicate that our system, leveraging multimodal references, provides users with a greater sense of agency compared to the baseline system.

Seven participants highlighted the usefulness of the history interface during the design process. P5 remarked that the feature was particularly effective, as there were times when he felt a previous font was better. In such cases, the history interface allowed him to revisit and continue from that point, saving effort. He also noted the inconvenience of the baseline system lacking this feature. P6 commented that comparing the current font displayed on the slider with previously created fonts helped him determine which one aligned more closely with his intended design. He also said that in the baseline system, he found it challenging to reset after creating a satisfactory font. In contrast, our system's history interface made him feel more confident about updating or resetting, as it allowed him to aim for even better results without hesitation. This exemplifies that the history interface is useful not only for storing the designed characters and enabling the users to go back to a past point but also for making them advance the design process as boldly as they want. It also indicates that the history interface reduces stress and increases freedom and creativity in the design task.

Overall, the feedback suggests that the proposed functions in our system effectively support font design for different participants based on their design preferences and familiarity with typography.

6 APPLICATION DEMONSTRATIONS

In this section, we will show a case study where participants are required to design characters in more realistic situations and the adaptation to another writing system rather than Roman characters.

6.1 Designing Fonts for Graphic Design Purposes

In practical usage, it is important to evaluate whether our system can support font design for graphic design purposes, such as logo design or advertisement design, as noted by professional designers in Section 7. To explore this, we asked participants to create suitable characters for specific design contexts.

To begin with the conclusion, from the feedback, we observed that participants using our system did not initially have a clear vision of the font they wanted. However, as they explored different font styles, they drew inspiration from the designs they encountered, ultimately creating their own unique characters. While participants occasionally struggled with fine adjustments, such as correcting distorted lines, they generally felt they were able to create fonts that aligned with their intended concepts. In the following sections, we present two design scenarios: conference logo design and advertisement poster design. In the conference logo task, four participants (P11-P14) created characters for a logo, demonstrating a variety of font styles using our system. In the advertisement poster task, another six participants (P15-P20) designed characters for different posters, tailoring their fonts to the target concepts.

6.1.1 Design a Conference Logo. In this experiment, participants with no prior font design experience were tasked with designing a conference logo. Specifically, they were asked to create the characters "CHI 2025" to complement the cherry blossom motif in the conference logo. After receiving an introduction to using our system, participants completed the task, and their feedback was collected. During the design process, participants were only shown the cherry blossom logo and were not aware of the characters in the official conference logo. As shown in Figure 10, the designs varied among participants. P11 noted that her designed characters complemented the cherry blossom logo, highlighting that her favorite aspect was the fading central lines in "H" and "5." This fading part appeared accidentally but complements the logo from her point of view, so she adopted it. She also attempted to replicate this effect in "2," but it was unsuccessful. P12 said that he thought a decent and calm font was suitable for the conference logo and tried to make such a font. He also commented that the font he designed was 90 out of 100 in terms of satisfaction, though his attempt to make "I" more straight was not successful. P13 commented that he aimed to create a cute font inspired by the Japanese subculture, opting for a bold and rounded design. For the initial step, he input the text, "I want a cute and thick font" and found that the system performed as he hoped. He also noted that the slider was effective in fine-tuning character details and eliminating unwanted distortions. He was proud of the font he designed and believed it could be used in real-world applications, as the style of each character was well aligned. P14 noted that he thought a thin and brush-style font fitted the Japanese-style logo and tried to make it. He found multimodal input effective for the early stages of rough font design but felt it was less suitable for detailed exploration, ultimately relying on slider manipulation to refine the characters. He was confident with the quality except for the noise and distortion in the designed characters.

6.1.2 Design Advertisement Posters. This demonstration shows font design for advertisement posters using our system, as illustrated in Figure 11. During the design process, participants (P15-P20), who were introduced to the use of our system, were given a scenario and shown only the background image, with the task of creating characters that matched the visual context. P15 and P16, both familiar with CJK writing systems, designed characters for an

Figure 10: Designed characters for the conference logo. Four participants designed the characters for the conference logo. They designed a diverse range of fonts based on their unique sensibilities.

autumn foliage festival. P5 rated his design 9 out of 10, expressing satisfaction with the traditional and formal font style he aimed to achieve. He efficiently initialized the search space by inputting the text "yu-mincho, serif" (with "yu-mincho" being one of the most popular CJK fonts). P16 commented that he envisioned a calm and warm font for the festival and was pleased with the result. He noted that their initial idea was simply based on the keyword "warm," which he input into the system. As he explored various styles, he gradually refined his design and reached a point of satisfaction. P17, who designed the summer sale poster, aimed for a thin and refreshing font. He observed elements in the background image such as the central white line, the seagull's wings, and the wave's border, and decided that the character weight should align with these features. By adjusting the slider, he was able to find a suitable font weight, though he expressed some dissatisfaction with the distortion of the top horizontal bar in the letter "E."

P18, tasked with designing a Halloween poster, felt that a twisted font suited the Halloween theme. She also believed a cute, handwritten style complemented the surrounding elements like the pumpkin, house, and bat, and was satisfied with the bold characters she created. She began by inputting a bold and italic font into the system, then continued refining the design using only the slider suggested by the Bayesian optimization process. She expressed confidence in using the system, despite having no prior font design experience, and enjoyed the process. She effectively utilized the system's features, such as reverting to previous iterations via the history area when her exploration veered in an undesired direction, and she repeatedly refined each character after the initial style propagation. For details on P18's design process, refer to the supplemental material. P19 designed characters for a birthday card, aiming for cursive and fashionable font, and expressed satisfaction with the result. P20 created a font for a movie poster, aiming for characters that were "scary," "thick," and "retro," with shapes fitting within a rectangular form (e.g., the shape of "S" resembling a rectangle). While he was generally satisfied with the overall design, he found it challenging to achieve symmetry in characters like "A," "M," and "T." Overall, although participants encountered challenges in addressing minor distortions and style inconsistencies, all expressed satisfaction with their final designs.

6.2 Designing CJK Fonts

In the user study Section 5.2, we demonstrated that participants could efficiently design Roman characters using our proposed system. By swapping the font generative model, the system can also

Figure 11: Designed characters for the advertisement posters. The participants designed the characters while viewing background images for the posters. The two posters on the top left are for an autumn foliage festival. P17 created a poster for a summer sale, while P18 designed characters for a Halloween event. P19 developed a font for a birthday card, and P20 created one for a movie poster.

support other writing systems, including Chinese, Japanese, and Korean (CJK). As shown in Figure 12, users can efficiently design CJK characters without the need of predesigned examples. Once the design process is complete, users can download their custom fonts as OTF files.
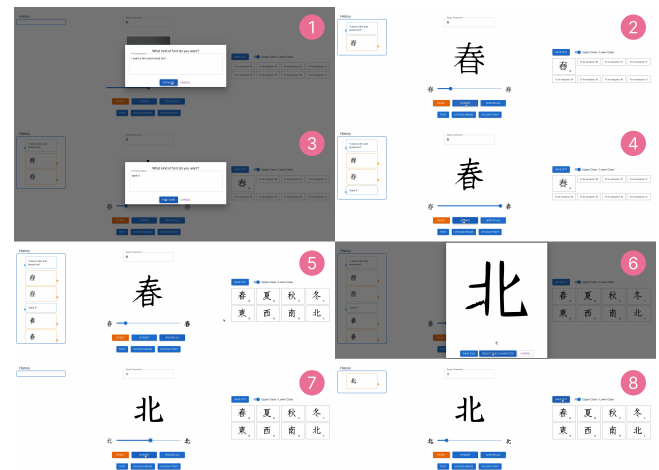
Figure 12: Screenshots of CJK font character design. The demonstration of CJK character designs using our system. The order is displayed in the upper right corner of each screenshot. See the supplemental material for the video.

7 DISCUSSION, LIMITATIONS AND FUTURE WORK

Professional Designers Interview. In our user study and demonstration, we focused on non-expert users, as our system is designed to help them create fonts without requiring specialized knowledge. However, feedback from professional designers (D1 and D2) is also crucial for identifying areas for improvement and enhancing the versatility of FONTCRAFT. To gather feedback, we interviewed two professional designers from an advertising company, demonstrating our system and its outputs while asking two main questions: 1. *"Is our system practical for designing advertisements in real-world scenarios?"* 2. *"Are there any areas in our method that could be improved or features that should be added?"*

For the first question, both designers said they often need to create or customize unique characters for advertisement posters and logos. They acknowledged that our system is suitable for such tasks, validating its use in designing characters for posters and conference logos. For the second question, they pointed out that the distortion in the generated characters should be fixed for use in real-world scenarios. Additionally, they expressed that a function allowing users to edit the generated characters is desirable. Moreover, D1 noted that, for logo design, she sometimes needs to create each unique character in different styles; thus, the style propagation feature is not needed in such cases. This feedback suggests that offering an option to disable style propagation could enhance user flexibility. D2 mentioned that it would be beneficial to save all generated characters, visualize them in the UI, and allow users to revisit any previous points. This implies that developing a more comprehensive history visualization feature could be a potential future work.

Limited Quality of the Generate Characters. As noted by several participants in Section 5.2 and Section 6, generated characters exhibit distortions and visual artifacts. Additionally, despite our system's ability to help users design style-consistent characters, some style inconsistencies persist. For instance, during the demonstration of conference logo design in Section 6, P11 mentioned the difficulty in applying the fading effect seen in "H" and "5" to "2". These issues stem from the limitations of the font generative model [37] and the vectorization method [24] used in our system. Our system requires a font generative model that can generate characters rapidly while manipulating the slider. Therefore, even though some existing font generative models [8, 10, 19, 20, 29] can produce characters with higher-quality and in more consistent styles, we cannot use them since they take longer time to generate. For example, *Vec-Fusion* [29] takes 10 seconds to generate one glyph using A100, which is not feasible for interactive applications. Meanwhile, this limitation also makes it harder for users to use our system to design style-consistent fonts containing many characters (e.g., 52). However, we believe that with the progress of font generative models, our system can utilize the latest models and generate fonts in higher quality. To tackle these issues, we plan to explore how to optimize both the speed and quality of font generation. This includes integrating more advanced vectorization techniques or improving the efficiency of recent generative models without sacrificing real-time performance.

Broader Search Subspace for Multimodal References. The current *multimodal-guided subspace* is constructed by directly connecting a single encoded multimodal reference and previous user preference. Although this approach enables users to explore the latent space around the multimodal reference, it may restrict other font style variations that are similar to the multimodal reference. In the future, we plan to investigate how to construct a search subspace by connecting one or multiple multimodal references to an area in the style latent space that contains potential samples matching the desired styles similar to [16].

Direct Character Editing. As described in Section 6, some users had difficulty create desired characters due to character detail artifacts introduced by the font generative model and vectorization process. In the future, to mitigate these artifacts, we plan to introduce painting tools that will allow users to edit the generated characters directly.

UI for Additional Typographical Support. To design a font ready for production, it is crucial to include features that align characters with baseline, mean line, and other typographical guidelines including kerning. These are essential for improving the overall appearance, alignment, and readability of the text. We next plan to investigate how to integrate different user interfaces to facilitate font design that takes these typographical guidelines into consider.

8 CONCLUSION

In this paper, we introduced FONTCRAFT, a new font design system that enables non-expert users to create fonts of any writing system without the need for predesigned characters. Our system makes two main technical contributions: multimodal-guided subspace and retractable preference modeling, which address the two key limitations in existing human-in-the-loop PBO. Additionally, we incorporated an iterative style propagation and refinement process, enabling users to design style consistent font. Through a study, we demonstrated that non-expert users can efficiently design Roman characters using our system, supported by both quantitative and qualitative analysis. Furthermore, we showcased how users could create Roman and CJK characters in realistic scenarios and achieve satisfying results. FONTCRAFT is independent of any specific font generative model, making it adaptable to various models beyond *DG-Font* used in this paper. This flexibility ensures FONTCRAFT can evolve with future font generation technology advancements.

ACKNOWLEDGMENTS

REFERENCES

[1] Eric Brochu, Tyson Brochu, and Nando De Freitas. 2010. A Bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 103–112.

[2] Eric Brochu, Vlad M Cora, and Nando De Freitas. 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599* (2010).

[3] Eric Brochu, Nando de Freitas, and Abhijeet Ghosh. 2007. Active preference learning with discrete choice data. *Advances in neural information processing systems* 20 (2007).

[4] Neill DF Campbell and Jan Kautz. 2014. Learning a manifold of fonts. *ACM Transactions on Graphics (ToG)* 33, 4 (2014), 1–11.

[5] Junbum Cha, Sanghyuk Chun, Gayoung Lee, Bado Lee, Seonghyeon Kim, and Hwalsuk Lee. 2020. Few-shot compositional font generation with dual memory. In *European Conference on Computer Vision*. Springer, 735–751.

[6] Liwei Chan, Yi-Chi Liao, George B Mo, John J Dudley, Chun-Lien Cheng, Per Ola Kristensson, and Antti Oulasvirta. 2022. Investigating Positive and Negative Qualities of Human-in-the-Loop Optimization for Designing Interaction Techniques. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI '22)*. Article 112. https://doi.org/10.1145/3491102.3501850

[7] Toby Chong, I-Chao Shen, Issei Sato, and Takeo Igarashi. 2021. Interactive Optimization of Generative Image Modelling using Sequential Subspace Search and Content-based Guidance. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 279–292.

[8] Bin Fu, Fanghua Yu, Anran Liu, Zixuan Wang, Jie Wen, Junjun He, and Yu Qiao. 2024. Generate Like Experts: Multi-Stage Font Generation by Incorporating Font Transfer Process into Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6892–6901.

[9] Stephanie Fu, Netanel Y Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. 2023. DreamSim: learning new dimensions of human visual similarity using synthetic data. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*. 50742–50768.

[10] Haibin He, Xinyuan Chen, Chaoyue Wang, Juhua Liu, Bo Du, Dacheng Tao, and Qiao Yu. 2024. Diff-font: Diffusion model for robust one-shot font generation. *International Journal of Computer Vision* (2024).

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[12] Xun Huang and Serge Belongie. 2017. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*. 1501–1510.

[13] Yue Jiang, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. 2017. DCFont: an end-to-end deep Chinese font generation system. In *SIGGRAPH Asia 2017 Technical Briefs*. 1–4.

[14] Florian Kadner, Yannik Keller, and Constantin Rothkopf. 2021. Adaptifont: Increasing individuals' reading speed with a generative font model and bayesian optimization. In *Proceedings of the 2021 chi conference on human factors in computing systems*. 1–11.

[15] Yuki Koyama, Toby Chong, and Takeo Igarashi. 2022. Preferential Bayesian Optimisation for Visual Design. In *Bayesian Methods for Interaction and Design*, John H Williamson, Antti Oulasvirta, Per Ola Kristensson, and Nikola Banovic (Eds.). Cambridge University Press, Chapter 8, 239–258.

[16] Yuki Koyama, Issei Sato, and Masataka Goto. 2020. Sequential gallery for interactive visual design optimization. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 88–1.

[17] Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. 2017. Sequential line search for efficient visual design optimization by crowds. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.

[18] Zhouhui Lian, Bo Zhao, Xudong Chen, and Jianguo Xiao. 2018. EasyFont: a style learning-based system to easily build your large-scale handwriting fonts. *ACM Transactions on Graphics (TOG)* 38, 1 (2018), 1–18.

[19] Yitian Liu and Zhouhui Lian. 2024. QT-Font: High-efficiency Font Synthesis via Quadtree-based Diffusion Models. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

[20] Ying-Tian Liu, Zhifei Zhang, Yuan-Chen Guo, Matthew Fisher, Zhaowen Wang, and Song-Hai Zhang. 2023. Dualvector: Unsupervised vector font synthesis with dual-part representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14193–14202.

[21] George Mo, John Dudley, Liwei Chan, Yi-Chi Liao, Antti Oulasvirta, and Per Ola Kristensson. 2024. Cooperative Multi-Objective Bayesian Design Optimization. *ACM Trans. Interact. Intell. Syst.* 14, 2 (2024). https://doi.org/10.1145/3657643

[22] Peter O'Donovan, Jundefinednis Lundefinedbeks, Aseem Agarwala, and Aaron Hertzmann. 2014. Exploratory Font Selection Using Crowdsourced Attributes. *ACM Transactions on Graphics* 33, 4, Article 92 (2014), 9 pages. https://doi.org/10.1145/2601097.2601110

[23] Song Park, Sanghyuk Chun, Junbum Cha, Bado Lee, and Hyunjung Shim. 2021. Multiple heads are better than one: Few-shot font generation with multiple localized experts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 13900–13909.

[24] Peter Selinger. 2003. Potrace: a polygon-based tracing algorithm.

[25] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.

[26] Danyang Sun, Tongzheng Ren, Chongxun Li, Hang Su, and Jun Zhu. 2017. Learning to write stylized chinese characters by reading a handful of examples. *arXiv preprint arXiv:1712.06424* (2017).

[27] Rapee Suveeranont and Takeo Igarashi. 2010. Example-based automatic font generation. In *International Symposium on Smart Graphics*. Springer, 127–138.

[28] Yuki Tatsukawa, I-Chao Shen, Anran Qi, Yuki Koyama, Takeo Igarashi, and Ariel Shamir. 2024. FontCLIP: A Semantic Typography Visual-Language Model for Multilingual Font Applications. *Computer Graphics Forum* (2024).

[29] Vikas Thamizharasan, Difan Liu, Shantanu Agarwal, Matthew Fisher, Michaël Gharbi, Oliver Wang, Alec Jacobson, and Evangelos Kalogerakis. 2024. VecFusion: Vector Font Generation with Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7943–7952.

[30] Yuchen Tian. 2016. Rewrite: Neural Style Transfer For Chinese Fonts. (2016).

[31] Yuchen Tian. 2017. zi2zi: Master Chinese Calligraphy with Conditional Adversarial Networks. (2017).

[32] Paul Upchurch, Noah Snavely, and Kavita Bala. 2016. From A to Z: supervised transfer of style and content using deep neural network generators. *arXiv preprint arXiv:1603.02003* (2016).

[33] Yizhi Wang, Yue Gao, and Zhouhui Lian. 2020. Attribute2font: Creating fonts you want from attributes. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 69–1.

[34] Yizhi Wang and Zhouhui Lian. 2021. Deepvecfont: synthesizing high-quality vector fonts via dual-modality learning. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–15.

[35] Yuqing Wang, Yizhi Wang, Longhui Yu, Yuesheng Zhu, and Zhouhui Lian. 2023. Deepvecfont-v2: Exploiting transformers to synthesize vector fonts with higher quality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18320–18328.

[36] Zeqing Xia, Bojun Xiong, and Zhouhui Lian. 2023. Vecfontsdf: Learning to reconstruct and synthesize high-quality vector fonts via signed distance functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1848–1857.

[37] Yangchen Xie, Xinyuan Chen, Li Sun, and Yue Lu. 2021. Dg-font: Deformable generative networks for unsupervised font generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5130–5140.

[38] Songhua Xu, Tao Jin, Hao Jiang, and Francis CM Lau. 2009. Automatic generation of personal chinese handwriting by capturing the characteristics of personal handwriting. In *Twenty-First IAAI Conference*.

[39] Kenta Yamamoto, Yuki Koyama, and Yoichi Ochiai. 2022. Photographic Lighting Design with Photographer-in-the-Loop Bayesian Optimization *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 92, 11 pages. https://doi.org/10.1145/3526113.3545690

[40] Zhenhua Yang, Dezhi Peng, Yuxin Kong, Yuyi Zhang, Cong Yao, and Lianwen Jin. 2024. Fontdiffuser: One-shot font generation via denoising diffusion with multi-scale content aggregation and style contrastive learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 38. 6603–6611.

[41] Yexun Zhang, Ya Zhang, and Wenbin Cai. 2018. Separating Style and Content for Generalized Style Transfer. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8447–8455.

[42] Baoyao Zhou, Weihong Wang, and Zhanghui Chen. 2011. Easy generation of personal Chinese handwritten fonts. In *2011 IEEE international conference on multimedia and expo*. IEEE, 1–6.

[43] Yijun Zhou, Yuki Koyama, Masataka Goto, and Takeo Igarashi. 2020. Generative melody composition with human-in-the-loop bayesian optimization. *arXiv preprint arXiv:2010.03190* (2020).

[44] Alfred Zong and Yuke Zhu. 2014. Strokebank: Automating personalized chinese handwriting generation. In *Twenty-Sixth IAAI Conference*.