Color Maker: a Mixed-Initiative Approach to Creating Accessible Color Maps

Amey Salvi
Indiana University Indianapolis
Indianapolis, IN, USA
amsalvi@iu.edu

Kecheng Lu
Shandong Univeristy
Qingdao, Shandong, China
lukecheng0407@gmail.com

Michael E. Papka
Argonne National Laboratory
Lemont, IL, USA
University of Illinois Chicago
Chicago, IL, USA
papka@anl.gov

Yunhai Wang
Renmin University of China
Beijing, China
wang.yh@ruc.edu.cn

Khairi Reda*
Indiana University Indianapolis
Indianapolis, IN, USA
redak@iu.edu

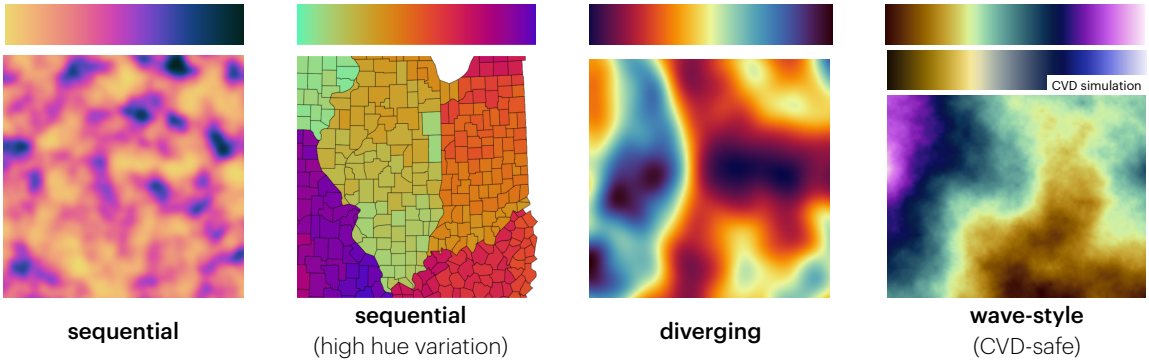

Figure 1: ColorMaker enables users to create and customize continuous colormaps in a variety of styles, including sequential, diverging, and 'wave'-style. Beyond optimizing for fundamental perceptual properties like uniformity, ColorMaker can also ideate and generate designs that are safe for individuals with color vision deficiency (CVD).

ABSTRACT

Quantitative data is frequently represented using color, yet designing effective color mappings is a challenging task, requiring one to balance perceptual standards with personal color preference. Current design tools either overwhelm novices with complexity or offer limited customization options. We present ColorMaker, a mixed-initiative approach for creating colormaps. ColorMaker combines fluid user interaction with real-time optimization to generate smooth, continuous color ramps. Users specify their loose color preferences while leaving the algorithm to generate precise color sequences, meeting both designer needs and established guidelines. ColorMaker can create new colormaps, including designs accessible for people with color-vision deficiencies, starting from scratch or with only partial input, thus supporting ideation and iterative refinement. We show that our approach can generate designs with similar or superior perceptual characteristics to standard colormaps. A user study demonstrates how designers of varying skill levels can use this tool to create custom, high-quality colormaps. ColorMaker is available at: **colormaker.org**

CCS CONCEPTS

• **Human-centered computing** → **Visualization toolkits**; *Scientific visualization.*

KEYWORDS

Mixed-initiative systems, color design, colormaps, simulated annealing.

ACM Reference Format:
Amey Salvi, Kecheng Lu, Michael E. Papka, Yunhai Wang, and Khairi Reda*. 2024. Color Maker: a Mixed-Initiative Approach to Creating Accessible Color Maps. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24), May 11–16, 2024, Honolulu, HI, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3613904.3642265

∗ Work performed in part while on sabbatical at Argonne National Laboratory.

CHI '24, May 11–16, 2024, Honolulu, HI, USA

ACM ISBN 979-8-4007-0330-0/24/05…$15.00
https://doi.org/10.1145/3613904.3642265

1 INTRODUCTION

Color is one of the most evocative and readily apparent properties in a visualization. Used effectively, color can help communicate complex quantitative information, such as a hurricane's projected track, the cosmic microwave radiation left over from the Big Bang, or the change in global temperatures over millennia. Despite the importance of this channel to visualization, designing quantitative colormaps is challenging. Visualization designers need to carefully balance various factors, such as perceptual uniformity [63], color nameability [46], and personal preferences. Given the complexity of the task, most color advice tools provide a limited set of manually crafted color scales to select from. Novice users are often cautioned against the temptation to use intuitional defaults like the rainbow [6], and instead encouraged to choose from an existing selection of expertly designed colormaps [22]. While acceptable in many cases, pre-constructed colormaps lack flexibility, restricting their usage in scenarios that dictate specific design needs. For instance, a visualization designer may desire specific colors at certain scale intervals to meet conventions (e.g., green for plain-level elevations and white for mountain peaks). Meeting these diverse requirements using only predefined colormaps is challenging.

Several techniques exist for generating categorical [21, 31, 32] or ordered palettes [42], but only a few explicitly support the creation of continuous colormaps. For example, Color Crafter allows users to select a preferred 'seed' color [55], which is then fitted to a suitable colormap curve. Although intuitive, curve fitting provides only limited control parameters. Other tools, like CCC [40], afford direct control over the colormap curve, providing greater design flexibility. However, this approach assumes that users are comfortable specifying their design requirements as a series of control points in a perceptual color space – a model that may be unintuitive for many, except for power users.

Hence, there is a need for design tools that provide a high level of design customizability without requiring precision specification or expertise in color perception. Moreover, since crafting colormaps for visualization is an iterative process, tools should ideally support users in all design stages, from ideation and evaluation of candidate solutions, to selection and refinement of designs. To address these needs, we introduce *ColorMaker*, a mixed-initiative approach to generating continuous colormaps. ColorMaker formulates the design process as an optimization-based search for a solution that balances user preference, established perceptual guidelines, and visualization accessibility needs. The interface employs familiar drag-and-drop interactions for selecting desired colors. It accommodates approximate or partial design requirements. These specifications are used to build a probabilistic model of user preference, which is then translated into a set of biases and soft constraints onto a simulated annealing optimization. This approach enables interactive generation of colormaps from scratch or with limited input, thus supporting design ideation and generative thinking. ColorMaker also enables the refinement of generated colormaps, automatically operationalizing user edits into additional optimizations to precipitate improved designs. In addition to modeling standard perceptual rules, ColorMaker can also account for color vision deficiency (CVD), generating colormaps that are interpretable by both normal-vision and CVD viewers.

To evaluate our approach, we compare a large sample of generated colormaps to commonly used designs. We find that ColorMaker generates colormaps with similar or superior perceptual characteristics to established best-practice designs. CVD simulation of generated colormaps also indicates high perceptual discriminability for viewers with CVD. Finally, we conducted a study with visualization designers, demonstrating that ColorMaker is intuitive and provides expressive power for users of all expertise levels.

Our **primary contribution** is a mixed-initiative approach for creating continuous colormaps. Our approach comprises a novel algorithm for colormap generation and optimization, closely integrated with a user-friendly design interface to support iterative colormap development. This integration promotes creative collaboration between the designer and the system, enabling the creation of new, accessible colormaps either from scratch or through partial user specification. Evaluation demonstrates the approach's intuitiveness and ability to produce high-quality colormaps. We have implemented this technique as an open-source[1], web-accessible tool: **https://colormaker.org**.

2 RELATED WORK

Various tools have emerged to facilitate the creation and analysis of colormaps. We situate our work in this landscape and discuss related approaches for colormap optimization, modeling, and generation.

2.1 Color Design Tools

For web designers and digital artists, a variety of tools exist, such as Adobe Color CC [2], Coolors [5], Colormind [16], ColorMagic [15], and Canva [13]. These tools offer a range of features, including the automatic generation of harmonious color palettes, sometimes based on an input image. However, these tools cater to general design needs and might not address specific concerns of data visualization, such as perceptual uniformity or discriminability.

Several tools have been designed to scrutinize and evaluate the perceptual properties of colormaps for visualization. For instance, ColorMeasures offers a comprehensive analysis of colormaps, exploring factors such as uniformity, discriminative power, and order in multiple color spaces [12]. However, it primarily focuses on analysis rather than colormap creation. Similarly, cols4all and VizPalette enable users to analyze and compare palettes but lack the capability to generate new designs [33, 56].

Certain tools provide data- or task-driven colormap recommendations. For example, PRAVDAColor suggests suitable colormaps based on data type, spatial frequency, and representation goal, aligning its recommendation with the perceptual theory of the time [51]. One of the most popular color selection tools is ColorBrewer, which provides high-quality, hand-crafted colormaps that meet design and perceptual standards, but offer limited modification options [9]. The CCC tool allows users to create and optimize continuous colormaps [40]. It provides a higher-level specification format that is meant to reduce the number of control points needed for colormap manipulation. Although the CCC tool provides high design flexibility, users are expected to edit and specify control points to realize new designs. This paradigm is arguably challenging for most, save for experienced users. By contrast, ColorMaker separates colormap

[1]The source code is available at: https://osf.io/4ugf6

specification from its user preference model, making it more user-friendly. Similarly, ColorMoves employs a straightforward interface that allows users to build elaborate colormaps by recombining a set of predefined 'inserts' [53]. However, unlike ColorMaker, this tool does not provide functionalities to computationally evaluate or optimize designs. Paraview integrates a colormap design dialog into a general-purpose scientific visualization suite, albeit without providing evaluation or optimization options [4].

ColorMaker strives to provide greater flexibility than existing color design tools, while still being user-friendly. We take inspiration from recent advancements in visualization authoring environments, including tools like Data Illustrator [30], Charticulator [48], and Lyra [54]. Similar to these tools, ColorMaker leverages intuitive drag-and-drop interactions to facilitate creative expression in colormap design. Given the need for precision specification in quantitative colormaps, we combine these user-friendly design features with an algorithm for generative modeling and optimization. Using principles of creative mixed-initiative interfaces [18, 23, 27], we enable the user and the system to co-create perceptual colormaps that cater to a wide range of styles and preferences.

2.2 Computational Colormap Generation and Optimization

A number of tools employ modeling and optimization-based methods to generate color encodings. ColorCrafter uses clustering techniques to model existing design practices, allowing the latter to be used as the basis for generating new colormaps [55]. However, this approach is limited in expressiveness, as users can only select a single 'seed' color to customize designs. Similarly, Wijffelaars et al's approach, although capable of replicating ColorBrewer's designs, allows only for a few control parameters, potentially limiting the set of designs that can be expressed [61]. Colorgorical employs a model-driven approach to generate categorical color palettes with adjustable scoring functions, providing good customizability but is limited to generating categorical palettes [21]. Similarly, Palettailor optimizes categorical palettes using simulated annealing and based on the underlying data characteristics [31]. However, Palettailor too is limited to qualitative palettes. Nardini et al. propose a set of algorithms to optimize continuous colormaps [39], which can improve perceptual characteristics, including uniformity, smoothness, and intuitive order. However, these algorithms are meant to improve a given colormap by making small adjustments to the control points. By contrast, ColorMaker employs a stochastic optimization process to explore a larger swath of the design space, enabling it to ideate new colormaps on its own while also conforming to user preferences when the latter is provided. ColorCAT is another tool that enables users to create task-driven colormaps, using design heuristics to match color-mapping strategies with certain classes of tasks [36]. However, compared to ColorMaker, which allows users to incorporate arbitrary hue preferences, ColorCAT offers limited customization options. Uniquely among existing color design tools, our approach tightly integrates user preferences and interactions with algorithmic optimization, enabling a collaborative human-system colormap design experience. This affords design agency while ensuring adherence to perceptual standards.

2.3 Addressing Color Vision Deficiency

Several color mapping tools notably address CVD, which affects approximately 4% of the population. *cmasher* offers CVD-friendly scientific colormaps [57], while VizPalette allows users to simulate the effects of different CVD conditions on palettes [33]. Similarly, Chromaticity [20] and cols4all [56] help evaluate color palette discriminability for CVD. ColorBrewer includes CVD-safe options among its colormaps [10]. Most of these tools either provide a set of pre-crafted colormaps considered to be CVD-accessible or allow users to evaluate their existing designs. Manual generation and modification are still needed to realize new, CVD-accessible colormaps with these tools.

Exceptions include IWantHue, which allows for generating CVD-safe categorical palettes [28]. Nuñez et al's Python module takes an arbitrary colormap and outputs a CVD-safe version [41]. This allows for converting problematic designs (e.g., rainbow) to a more accessible form. The technique works by limiting colormaps to a CVD-safe truncated color space. The resulting designs, however, tend to exhibit dull blue to yellow tones, which may be unappealing for normal-color vision viewers. Instead of truncating the color space, ColorMaker allows selection from the entire displayable gamut but limits the design to color *combinations* that cannot be easily confused. This results in a vibrant colormap that retains discriminability for viewers with CVD.

2.4 Perceptual Standards for Colormaps

A multitude of guidelines have emerged to aid in the design of effective colormaps [63]. Among these, the principle of smoothness has been consistently advocated to prevent abrupt transitions (or color banding) in the color sequence [6, 38]. Perceptual uniformity is another pivotal factor in colormap design [26]. A perceptually uniform color sequence ensures that observers can easily distinguish differences in data values as represented by adjacent colors [12]. Perceptual 'order' is another commonly cited factor in making intuitive colormaps [6]. The latter implies that a viewer should be able to deduce the relative ordering of colors without consulting a legend. ColorMaker explicitly models perceptual uniformity and smoothness. While we do not attempt to optimize for order, the latter is helped by ensuring locally monotonic changes in luminance and by limiting the curvature of the colormap. Although not guaranteeing a perceptually orderable colormap in the absolute sense [11], together these two factors appear to yield qualitatively good color ordering. Luminance control is often emphasized as a crucial factor for continuous colormaps [44, 49, 50, 59], hence we model the latter as a hard constraint, in line with the approach used in Matplotlib [58]. Within this constraint, we allow the user to dictate the luminance profile (e.g., sequential, diverging, or wavy [52]).

While crucial to effective colormap design, reconciling these guidelines with user preference is non-trivial. ColorMaker encodes these requirements as a set of scoring functions and soft constraints. It then uses simulated annealing to search for a satisfying solution.

3 DESIGN REQUIREMENTS

In developing ColorMaker, our aim was to create a tool that supports color mapping by visualization designers of all levels of expertise. We wanted to strike a balance between providing designers

with creative control and leveraging computational optimization techniques, so as to ensure that the resulting colormaps adhere to fundamental principles of perceptual color encoding. We discuss the key design requirements we sought to fulfill.

3.1 Balancing Designer Creativity and Automation

Effective colormap design often involves reconciling the personal preferences of designers with established perceptual standards, such as ensuring luminance monotonicity and perceptual uniformity. Balancing user color preferences with perceptual standards can be daunting. Our fundamental hypothesis is that by integrating color selection interfaces with interactive user-driven optimization, we can empower designers with significant control while automating the majority of the colormap generation process. With ColorMaker, designers can specify the hues and colors they wish to incorporate, along with their approximate order. Tedious tasks, such as meticulously positioning control points to achieve a smooth and uniform color gradient, are handled by the algorithm. Beyond automating laborious aspects of colormap development, the algorithm exhibits creative agency by exploring the space of possible solutions, 'filling in' gaps within the user's specification, and responding appropriately to user edits.

This approach borrows from principles of creative, mixed-initiative interfaces [18, 62], leveraging human creativity and computational techniques to support complex design tasks. Taking inspiration from this philosophy, our aim was to create a system where a human designer and the computer take turns to collaboratively refine the design, constrain the space of possible solutions, and ultimately ensure that resulting colormaps align with user preferences while meeting key perceptual metrics for effective visualization.

3.2 Minimal Expertise Required

ColorMaker is designed to be accessible to non-expert users, enabling them to create customized and effective color encodings. The user interface incorporates familiar user interface elements, such as standard color pickers and drag-and-drop interactions. Instead of demanding precise color specifications, ColorMaker was designed to accept uncertain and imprecise user input. For example, a user can specify an approximate red hue at the desired position in a diverging colormap. The optimization algorithm then calculates the best approximation of this preferred color while ensuring a perceptually sound colormap. Importantly, ColorMaker is forgiving of potentially suboptimal or even problematic input. For instance, if a user prefers two opponent hues in close proximity, which would typically result in a sharp visible boundary, ColorMaker will select a midpoint color instead, or steer the design to incorporate one of the two hues selectively. Similarly, when optimizing for color vision deficiency (CVD), the algorithm self-adjusts problematic color pairs (e.g., greens and reds) by selecting similar hues that can still be distinguished by individuals with CVD.

3.3 Scaffolding Ideation and Iterative Design

Creating colormaps is an iterative and creative process. Designers might start with a clear vision or have only a vague concept. To support the exploration of the design space, ColorMaker generates colormap suggestions from the ground up by leveraging its stochastic optimization. Multiple distinctive designs can be synthesized and presented simultaneously to stimulate generative thinking. The tool also accommodates partial input, allowing users to indicate preferences for specific hues at different parts of the color scale. Once an initial solution is generated, ColorMaker provides various interactions for refinement. Users can adjust hue or chroma at arbitrary points in the colormap through lightweight editing functionalities. They can also revise their preferences by repositioning desired colors or adjusting intervals. ColorMaker incorporates these refinements into its user preference model and schedules additional optimization runs to interactively update the design.

While ColorMaker does not assume specific expertise in color perception, it offers optional features for fine-grained editing and colormap evaluation. Users can view perceptual characteristics of colormaps from within the interface, including changes in luminance, color distance, and smoothness. Advanced users can also view and adjust control points to fine-tune the colormap curve, as needed. These adjustments can be performed through direct drag-and-drop interactions, and are similarly incorporated into subsequent optimization runs.

3.4 Fostering Accessible Visualization

A significant portion of visualizations employs color schemes that are difficult to interpret by individuals with color vision deficiency (CVD) [3]. This issue arises in part because balancing designer preferences with accessibility needs is challenging. ColorMaker addresses this challenge by considering CVD effects during colormap generation. The optimization process penalizes designs that incorporate potentially confusing color combinations (e.g., isoluminant reds and greens), favoring CVD-safe solutions. ColorMaker automatically handles these constraints, allowing designers to focus on articulating their needs while the algorithm takes care of CVD modeling and optimization.

4 THE COLOR MAKER INTERFACE

ColorMaker is a single, self-contained web application (see Figure 2). We discuss the user interface components and illustrate how they work together with example usage scenarios.

4.1 Incorporating Designer Preference

The ColorMaker interface includes a color picker to facilitate the selection of desired colors (Figure 2-Ⓐ). By default, this picker allows color selection within the *CIE LCh* color space (luminance, chroma, and hue) but also supports several other perceptual (and non-perceptual) color spaces, including CIE Lab, CAM02-UCS, and RGB. The picker cuts through one color dimension (e.g., L^*) via a slider, presenting a 2D slice of the other two dimensions for easy selection. Once a desired color is chosen, users can simply drag and drop it into a 'preference shelf' Ⓑ at the approximate position where they want the color to appear. Each color added to the shelf is represented as a movable color block. Users can add multiple color blocks to convey several preferences simultaneously, such as various hues at different positions within the scale. Furthermore, users can adjust the size of these color blocks to indicate the desired extent within the colormap, as illustrated in Figure 3.

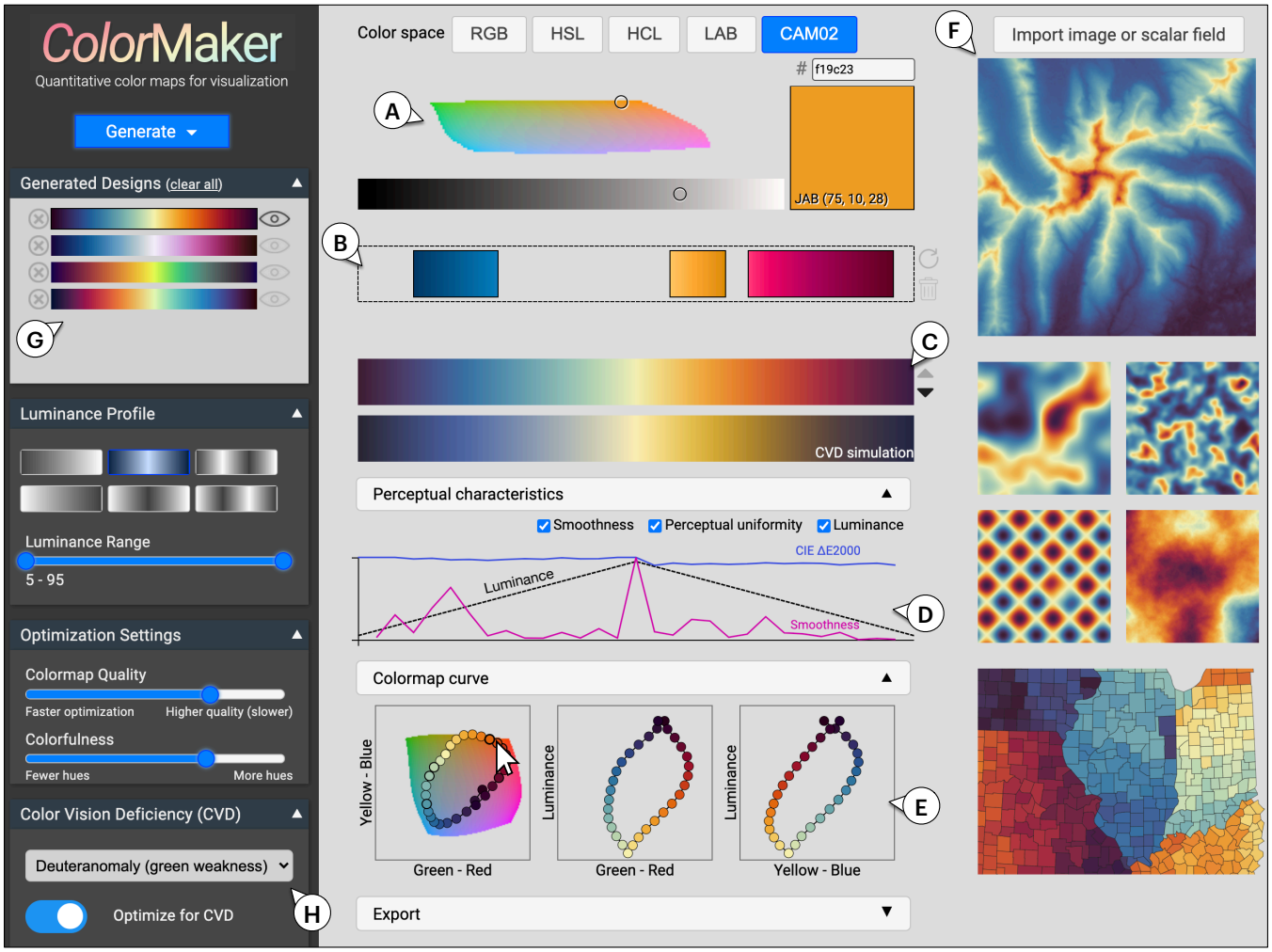

Figure 2: The ColorMaker interface consists of several key elements: (A) A configurable color picker that supports selection in multiple color spaces. (B) A 'preference shelf' that enables users to indicate their preferred colors at their approximate desired positions. (C) The generated colormaps are displayed as continuous scales, accompanied by CVD simulation for accessibility considerations. (D) The user can access information about the underlying perceptual characteristics of the colormap. (E) Users have the option to directly edit the colormap curve if desired, providing fine-grained control. (F) The colormap is applied to a selection of visualizations for quick evaluation. Users can also import their own images or scalar fields for further testing. (G) A sidebar keeps a history of all generated colormaps, allowing the user to revisit and fine-tune previous design variations. (H) Optimization parameters, including the option to simulate CVD effects, can be adjusted to provide additional control over the colormap generation process.

The preference shelf Ⓑ serves as the foundation for a user preference model (explained in §5.7). Importantly, ColorMaker intentionally separates this user model from the colormap itself, or more specifically, from the series of control points that define the colormap curve. This separation accommodates fluid preferences, which can be approximate, incomplete, or even capricious. For instance, when adding a preferred color, its position can be chosen arbitrarily. Similarly, the extent of color blocks can be specified flexibly, potentially with multiple colors overlapping. This fluidity is meant to empower designers to think creatively. It is during the optimization process that ColorMaker seeks to find a balance between what the designer wants and what is feasible within perceptual constraints and device limitations (e.g., the color gamut of the display). Whenever user preferences change, such as by moving, adding, or removing preference colors, ColorMaker automatically updates its user model and triggers additional optimization runs to precipitate new designs based on updated preferences. The resulting colormap is displayed Ⓒ along with its perceptual characteristics Ⓓ and applied to sample visualizations and test patterns. For additional evaluation, the user can visualize the colormap with a custom dataset Ⓕ. The latter can be uploaded as an image or a scalar field.

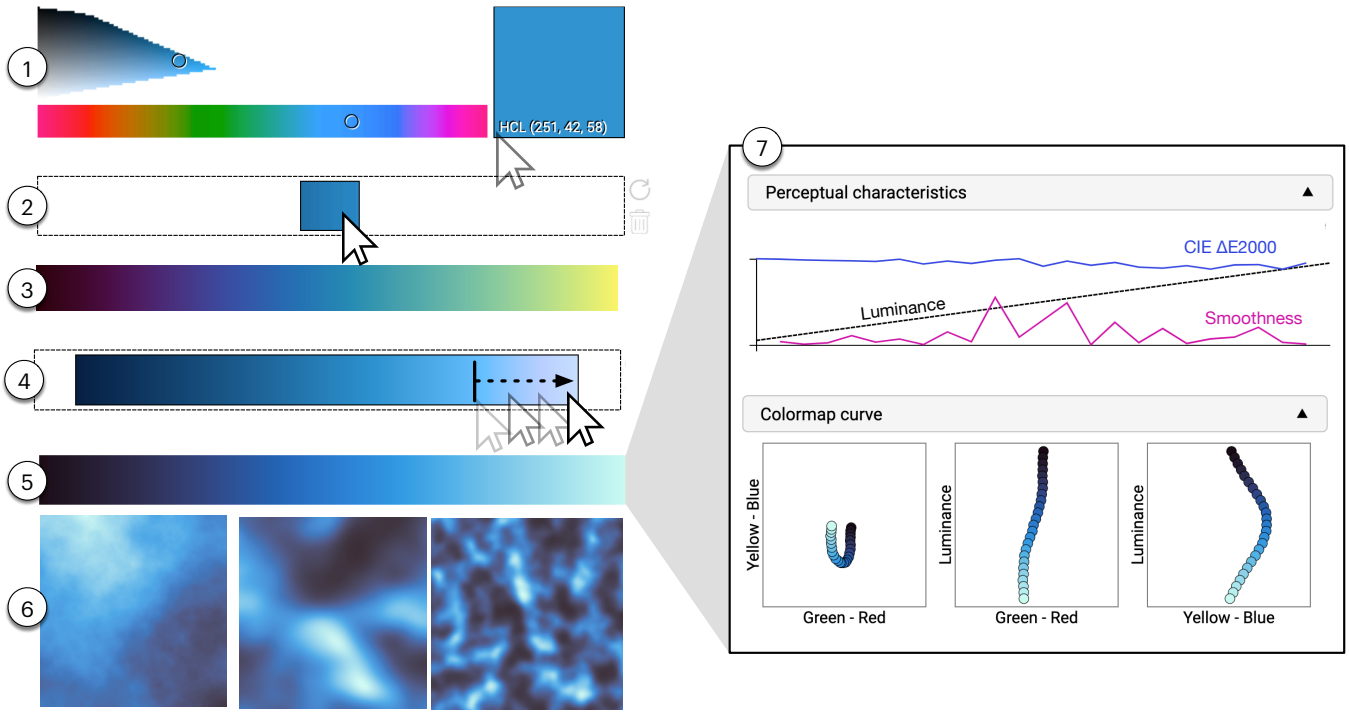

Figure 3: Steps for creating a sequential, single-hue colormap for an ocean bathymetry dataset. A blue color (1) is added to the preference shelf (2), leading to the generation of a multi-hue colormap (3). Stretching of the blue block indicates a preference for a predominantly single-hued design (4), resulting in a second solution that meets the requirement (5). The colormap is checked against example scalar fields (6) and evaluated for its perceptual characteristics (7).

4.2 Refining Generated Designs

To refine the generated design, ColorMaker offers two options. First, designers can directly edit the color scale: When users hover over the generated scale, ColorMaker presents a palette of suggested options to adjust the color at the corresponding point, as shown in Figure 4-④. This palette specifically allows users to modify the chroma and hue of a selected scale point. Users can either increase or decrease the chroma (i.e., saturate or desaturate the color) or choose to rotate the hue. This palette is intentionally limited to a few options that represent localized hue/chroma adjustments for quick refinement. For more precise control, ColorMaker allows users to directly edit the colormap curve by adjusting the position of its control points. For instance, when the CIE Lab space is selected, ColorMaker displays three projections: A-B, L^*-A, and L^*-B (see Figure 2-Ⓔ). From these projections, users can drag any control point to assign a different color. For example, moving control points in the A-B slice changes the hue and chroma while keeping luminance constant, whereas RGB projections enable adjustments in two of the three color channels while keeping the third constant. To visualize the range of possible color options for a control point, ColorMaker displays a slice of the color space within each projection during user interactions.

As users refine the colormap using either of the two interactions above, ColorMaker automatically incorporates user edits as additional preferences, integrating them as constraints into the optimization process.

4.3 Optimization Controls

In the sidebar, users can select from a variety of luminance profiles, including linear, diverging, and wave-style (seesaw luminance pattern [52]), as well as their inverses. Users can also adjust the desired luminance (L^*) range, which is set to a default of $L^* \in [5, 95]$. The chosen luminance profile is enforced as a strict constraint during the optimization process, ensuring that all generated colormaps adhere to it. Additionally, users can fine-tune other optimization parameters through the sidebar (Figure 2-Ⓗ). This includes a 'colorfulness' slider that controls the penalty for smoothness; a higher setting results in a colormap with more curvature, and thus higher hue variation. Users can also adjust the 'colormap quality', which determines the number of iterations in the optimization; higher quality requires more iterations for longer optimization times.

4.4 Usage Scenarios

To illustrate how ColorMaker supports colormap ideation and design, we describe three example usage scenarios.

Scenario 1 — Creating a Single-Hue Colormap: Meet Jane, an Earth scientist working on an oceanography project. She's dealing with a bathymetry dataset, representing ocean bed depth, which naturally progresses from sea level to deeper topographic features. To visualize this dataset, Jane wants a sequential colormap that predominantly uses shades of blue. Opening ColorMaker (Figure 3),

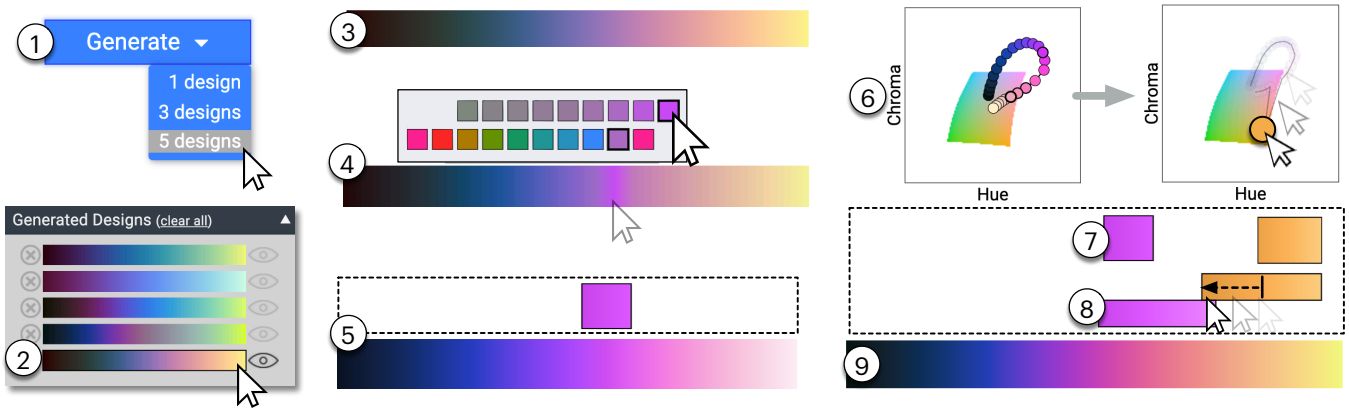

Figure 4: A multi-hue design created by first ideating a random set of solutions (1–2) and selecting one of five generated colormaps to work with (3). The user locally edits the colormap to saturate the purple (4). This adjustment causes the optimization to update the preference model, although it unexpectedly removes the orange-yellow hue previously visible at the end of the scale (5). The user reincorporates that color by editing the curve in the CIE LCh color space (6), and manually repositions the color blocks to indicate a desired blending between purple and orange (7–8), yielding the final colormap (9).

she selects a blue color from the CIE LCh picker ①, roughly matching her intended theme. Initially, she places this color somewhat arbitrarily in the middle of the preference shelf ②. In response, ColorMaker generates an initial colormap ③ that incorporates this particular blue. However, Jane realizes that this initial solution includes additional hues she is not interested in having. To express her preference for a single hue, she stretches the blue color block ④. This action signals that Jane prefers blue throughout the colormap. ColorMaker reacts by updating its user model and scheduling another optimization. This time, the resulting colormap ⑤ aligns with Jane's requirements. She checks the example scalar fields ⑥ provided by ColorMaker, and confirms that this colormap suits her needs. Jane then examines the colormap's properties by opening the 'perceptual characteristics' and 'colormap curve' panes ⑦. She verifies that the colormap indeed has a monotonically increasing luminance profile with even perceptual distances. A glance over the colormap curve in the *CIE Lab* space further confirms that the solution is smooth. Finally, she exports the colormap as an array of control points in the RGB space for convenience.

Scenario 2 – Ideating and Refining a Multi-Hue Colormap: Eduardo is a data scientist who typically uses the default *viridis* colormap in Matplotlib. This time, he is interested in creating a new colormap from scratch. Lacking a clear idea, he utilizes ColorMaker's ideation feature to generate five solutions (Figure 4-①) without providing any input. ColorMaker responds with five suggested solutions ② and displays a thumbnail for each. Eduardo previews the designs by hovering over the thumbnails and selects one that intrigues him ③. While he likes the combination of blue, purple, and orange, he finds the colors somewhat muted. Upon hovering over the scale, ColorMaker suggests alternative chroma and hue values for purple ④. Eduardo chooses a more saturated color, and ColorMaker responds by adding this selection to the preference shelf ⑤, and performing another optimization to fine-tune the design. The new solution predominantly includes the new purple shade, but to Eduardo's surprise, orange has disappeared from the scale. Examining the colormap in the *CIE LCh* space, he notices that the curve has moved up to a more neutral white. To reintroduce orange, he selects a control point near the end of the curve and drags it down ⑥, forcing the curve to pass through orange again. This particular orange is automatically added to the preference shelf ⑦ by ColorMaker. Eduardo further adjusts the preferences shelf ⑧, ensuring that the orange and purple blocks overlap slightly for a smooth gradient. Another optimization run accounts for the updated preference model, yielding a colormap ⑨ that retains the original design features but with more vibrant colors.

Scenario 3 – Creating a CVD-Friendly Diverging Colormap: Sue is a climate scientist who is interested in visualizing results from climate simulation. Her model gives projections of temperature fluctuations over the next 100 years relative to a baseline year. To represent temperature differences effectively, she chooses a diverging colormap. However, rather than using the common cool-warm colormap that is typical of climate visualizations [19], she is interested in creating a new colormap. To ensure accessibility for colorblind individuals, she activates the 'Optimize for CVD' option, specifying deuteranomaly (a common form of color deficiency). She then uses ColorMaker's ideation feature to generate five initial designs (Figure 5-①). Previewing the suggested designs, Sue selects the orange-to-violet design as a favorite. To match the data semantics (warmer colors for higher temperatures), she flips the colormap horizontally, resulting in a final design ②.

Sue also explores another suggested design with a dull brown-to-purple profile ③. To make the colors more vivid, she adjusts purple to a saturated red-rose using ColorMaker's localized suggestion palette ④. She also adds green ⑤ to cover the lower arm of the scale. In combination, these two colors can be easily confused by viewers with green-red deficiency. To account for these CVD effects, ColorMaker optimizes by adjusting green to a more neutral beige

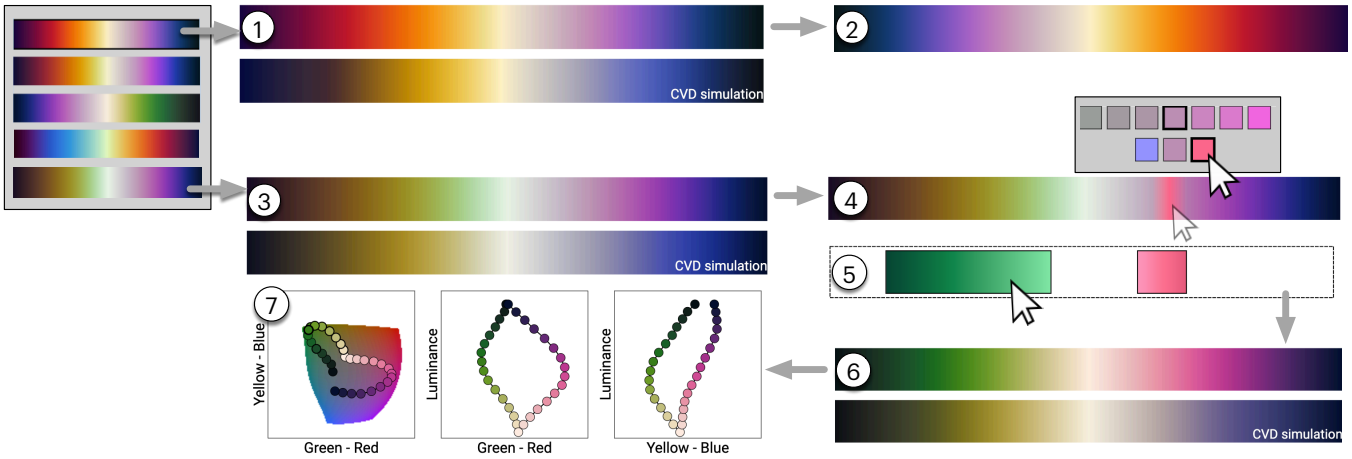

Figure 5: Two diverging colormaps generated for CVD accessibility. Starting from a set of five suggested designs, the user opts for the orange-to-lavender design (1), which is flipped to accommodate desired data semantics (2). As an alternative, the user further selects a second, brown-to-purple colormap (3), refining it to incorporate brighter rose (4) and green (5). Although these colors may be problematic for people with green-red blindness, ColorMaker finds an appropriate solution that balances this preference while maintaining a minimum level of color separability for CVD viewers (6).

near the middle of the scale ⑥, while retaining a vibrant green in the mid-to-low ranges. Inspecting the colormap curve in the CIE CAM02-UCS space reveals the algorithm has shied away from pure reds, instead incorporating blue-to-purple tones, in effect reducing the likelihood of red-green confusion. While this second solution is arguably less CVD-robust compared to the previous violet-orange design, it demonstrates how ColorMaker can balance competing constraints, finding an acceptable middle-ground solution.

In addition to these scenarios, Figure 1 showcases colormaps generated by ColorMaker without specific user preference.

5 METHOD

Having illustrated the user interface and interactions, we now describe the underlying mechanisms of ColorMaker. ColorMaker uses a custom, simulated annealing algorithm [1] to search the design space. At each iteration, the algorithm randomly perturbs the current solution and evaluates the colormap quality. We score candidate solutions based on their perceptual uniformity, smoothness, and the degree to which they can be considered CVD-accessible. We discuss these scoring functions and the algorithm. We then describe how we incorporate user preferences into the optimization.

5.1 Assumptions

A continuous colormap can be described by a function $f : [a, b] \subset \mathbb{R} \mapsto \mathbb{C}$, which maps a scalar x_i to the color c_i in a color space $\mathbb{C}$. This mapping is defined by an increasing sequence of values $\langle x_1, ..., x_n \rangle \in [a, b]$ and corresponding colors $\langle c_1, ..., c_n \rangle \in \mathbb{C}$, with $f(x_i) = c_i$. For simplicity, we assume that the sampling points x_i are equally spaced in the data space (although their associated colors need not be). ColorMaker uses the *CIE Lab* as the underlying color space $\mathbb{C}$. We refer to the sequence of colors C=$\langle c_i \rangle$ as *control points*, and assume that the colormap gradient will pass directly through these colors. We also assume Euclidean, linear interpolation in the CIE Lab space for mapping values x that lie in between two control points $x_i < x < x_{i+1}$, such that $f(x) = \alpha.\vec{v}_{i+1,i} + \vec{c}_i$ where $\vec{v_{j,i}} = \vec{c_j} - \vec{c_i}$ and $\alpha \in (0, 1)$. This in turn allows us to efficiently resample a colormap to m control points $C'_{(m)} = \langle c'_1, ..., c'_m \rangle$. The number of control points n, which dictates the degrees of freedom for the optimization, can be varied by ColorMaker depending on the sought luminance profile, with more complex designs (e.g., diverging and wave-style) requiring more control points.

5.2 Optimization

We frame colormap generation as an optimization problem to balance three key factors: **perceptual uniformity**, **smoothness**, and **accessibility** for viewers with CVD. These three elements are fundamental to ensuring a quality colormap that can be interpreted by most viewers [17, 50]. While other factors, like perceptual order [6], are discussed in the literature, we focus on these three as essential criteria for continuous colormaps. The essence of this optimization is to determine a color sequence $C^* = \langle c_i \rangle$ that minimizes the following objective cost function $E(C)$:

$$C^* = arg\min_{C} \mathbf{E}(C), \text{ where}$$

$$\mathbf{E}(C) = \omega_u \mathbf{E}_{Uniformity}(C) + \omega_s \mathbf{E}_{Smooth}(C) + \omega_c \mathbf{E}_{CVD}(C) \quad (1)$$

Here, the three terms represent scoring functions that penalize the colormap for deficiencies in perceptual uniformity ($\mathbf{E}_{Uniformity}$), for lack of smoothness ($\mathbf{E}_{Smoothness}$), and for the potential of misinterpretation by viewers with CVD ($\mathbf{E}_{CVD}$). The ω terms determine the relative importance of these three factors in the optimization. By default, we set $\omega_u = 0.85$, $\omega_s = 1$, and $\omega_c = 2$. Slightly reducing the weight of perceptual uniformity enhances design diversity while still allowing for uniform designs to emerge (see §6.1 for

results). We assign a higher weight to the CVD factor to increase the likelihood of generating an accessible design. We will further detail each of these three cost functions and discuss the rationale behind their formulation.

5.3 Perceptual Uniformity

A perceptually uniform colormap yields a perceived change in color that is proportional to the amount of increase (or decrease) in data values. Thus, a constant change in data should produce a proportional local change in color distance:

$$\forall i \in \{2, ..., n-1\} : d_{i,i+1} = d_{i,i-1} \tag{2}$$

Where $d_{i,j} = \Delta E^{00}(c_i, c_j)$ is the perceived distance between two colors c_i and c_j. Here, we use the *CIE* $\Delta E2000$ metric, as it is more accurate than Euclidean Lab distance and can still be computed efficiently. Following Bujack et al. [12], we quantify the drift from ideal perceptual uniformity by measuring the standard deviation (σ) of adjacent color distances $d_{i,i+1}$. Because these distances (and their standard deviation) vary considerably across colormap designs (e.g., diverging colormaps exhibit larger deviations than linear), we normalize σ, dividing by the mean color distance $\bar{d}$. The resulting coefficient of variation (CV) represents our normalized perceptual uniformity cost term:

$$\mathbf{E}_{Uniformity}(C) = \frac{\sigma}{\bar{d}} = \frac{1}{\bar{d}}\sqrt{\frac{1}{n-2}\sum_{i=1}^{n-1}(d_{i,i+1} - \bar{d})^2} \tag{3}$$

$$\text{where } \bar{d} = \frac{1}{n-1}\sum_{i=1}^{n-1} d_{i,i+1} \tag{4}$$

5.4 Smoothness

A smooth colormap transitions seamlessly without abrupt changes in the color gradient. Minimizing color "banding" is crucial to prevent viewers from misconstruing false data changes [6]. That said, defining a single metric to predict color banding is challenging as there are multiple factors involved, including variations in color names [47] and local perceptual distances [12]. However, in a perceptual color space, one indicator of smoothness is the curvature of the colormap curve. A relatively linear colormap tends to incorporate fewer hues while maintaining a nearly constant change in local color distance, resulting in a smoother gradient. Conversely, colormaps with high curvature introduce more hue and chromatic variations, giving rise to visual discontinuities. Additionally, sharp bends in the curve can lead to a reduced perception of color 'speed', which manifests as visible bands in the scale.

We therefore assess colormap smoothness by approximating its curvature. An accurate estimate of the latter could be derived from a higher-order representation of the curve, but such computation can be prohibitively costly. Instead, we seek a more efficient approximation by measuring the angle θ_i between displacement vectors at the control points. Specifically, we approximate colormap curvature as follows:

$$\text{Curvature}(C) = \frac{1}{n-2}\sum(\frac{1-\cos\theta_i}{2}) \tag{5}$$

$$= \frac{1}{2(n-2)}\sum_{i=2}^{n-1}(1 - \frac{\vec{v}_{i,i-1}\cdot\vec{v}_{i+1,i}}{\|\vec{v}_{i,i-1}\|.\|\vec{v}_{i+1,i}\|}) \tag{6}$$

$$\text{where } \vec{v}_{j,i} = \vec{c}_j - \vec{c}_i \tag{7}$$

Normalization of the cosine term is meant to limit the range of the cost function to $[0, 1]$: 0 indicating a perfectly straight segment (i.e., no penalty), and 1 for a complete reversal of direction (i.e., a 180° angle). It is important to note that some curvature is desired so as to allow the colormap to incorporate multiple hues. At times, the designer may wish to limit the number of hues appearing, and hence a higher smoothness penalty may be desired. To meet these needs, we allow the user to partially control the penalty weight, with the full cost function for smoothness defined as:

$$\mathbf{E}_{Smoothness}(C_{(n)}) =$$
$$\omega_{s,1}\text{Curvature}(C_{(n)}) + \omega_{s,2}\text{Curvature}(C'_{(\lfloor n/2 \rfloor)}) \tag{8}$$

The first term computes the penalty at the original resolution of n control points with a fixed weight of $\omega_{s,1} = 1$. The second term computes the smoothness penalty over a resampled colormap C' with $\lfloor \frac{n}{2} \rfloor$ control points. The weight for the latter term $\omega_{s,2} \in [0, 1]$ can be controlled by the user. The rationale for the first term is that it is important to ensure a mostly straight line between consecutive colormap segments so as to prevent high-frequency bends. At a lower sampling frequency, the curve can be allowed to deviate from a straight line depending on designer preference (the higher the curvature, the more colorful the design), hence the second, variable-weight term.

5.5 CVD Accessibility

To optimize for accessibility, we penalize the colormap for incorporating potentially confusable colors. The latter includes certain color pairs that share the same luminance level (e.g., isoluminant green and red tones), which some viewers have difficulty telling apart. To detect problematic colors, we test every pair of equally luminant colors, simulating their appearance to someone with a CVD condition (e.g., deuteranopia). We employ the model proposed by Machado et al. to transform colors from normal-vision coordinates to their simulated CVD appearance [34]. We then measure the distance between the CVD-transformed pairs, and exact a penalty when that distance is lower than a threshold $\gamma(K)$. Specifically:

$$\mathbf{E}_{CVD}(C) = \frac{1}{|\Psi|}\sum_{\langle i,j\rangle\in\Psi}\phi(\hat{c_i}, \hat{c_j}) \tag{9}$$

$$\text{where } \hat{c_i} = T_{CVD}.c_i \tag{10}$$

$$\phi(\hat{c_i}, \hat{c_j}) = \begin{cases} 0, & \text{if } \Delta E(\hat{c_i}, \hat{c_j}) >= K_{ij} \\ 1 - \dfrac{\Delta E(\hat{c_i}, \hat{c_j})}{K_{ij}}, & \text{if } \Delta E(\hat{c_i}, \hat{c_j}) < K_{ij} \end{cases} \tag{11}$$

$$K_{ij} = \gamma(K, i, j) = K.\frac{e^{|i-j|/(n-1)} - 1}{e - 1} \tag{12}$$

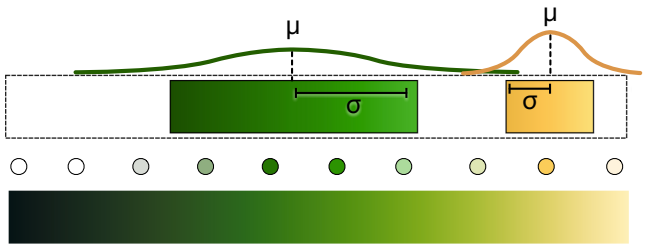

Figure 6: Illustration of how ColorMaker incorporates chromatic bias towards preference colors (green and yellow, in this example). Each color block exerts a certain amount of bias on every control point (circles). The magnitude of the bias follows a normal distribution centered at the middle of the block. In this example, green exerts a more diffused bias impacting several control points, whereas the bias to yellow is focused on the second control point from the right. The cumulative, averaged bias vectors are blended with random offset vectors during the optimization.

Ψ is a set of control point pairs that lie on the same L^* slice in the CIE Lab space. We obtain a CVD simulated color $\hat{c_i}$ by transforming c_i with a matrix T_{CVD}. ColorMaker allows the user to select which CVD condition to simulate and optimize for, with deuteranomaly (the most common) being the default. The constant K dictates the minimum distance threshold in CIE Lab units. By default, we use a value of $K = 70$ which we found to be a sweet spot. The cost function applies a penalty if the distance between the simulated color pairs, $\hat{c_i}$ and $\hat{c_j}$, is less than $K_{ij} = \gamma(K, i, j)$. The attenuation function γ serves to reduce the minimum required threshold K when the two colors i and j are in close proximity within the sequence. This attenuation is important so as not to penalize neighboring colors that should, by necessity, be perceptually similar. The above cost function ensures continuous and automatic steering of colormaps towards accessible color combinations. That said, CVD modeling is optional and can be disabled by setting T_{CVD} to an identity matrix I_3. In this case, we still retain the cost function to ensure sufficient distance between the control points, effectively preventing the solution from looping and 'reusing' the same colors in different parts of the scale (e.g., using the same blue twice in the two arms of a diverging colormap).

One limitation in the current implementation is that it only considers one CVD condition at a time, although the user can specify which condition to model. It is possible to extend Equation 9 to multiple conditions by summing up the penalties from three CVD matrices, for instance. Doing so, however, severely limits the color space, leading to dull designs. Our approach is to instead model the most prevalent and severe CVD conditions by default while still allowing optimization for less common conditions when needed.

5.6 Algorithm

We employ a custom, simulated annealing algorithm to optimize the objective cost function. The algorithm starts at a random solution with all control points initialized to random colors chosen from the displayable CIE Lab gamut. Specifically, we set the luminance of the control points to the requested luminance profile (e.g., monotonically increasing or diverging), while randomly initializing the A and B (i.e., chromatic) components of the color. The algorithm begins at a high initial 'temperature', which is then progressively cooled. At every iteration, the algorithm perturbs the current solution by offsetting a randomly selected control point by a small amount, in the direction of a random vector $\vec{r}$. We only offset the A and B components of the color, leaving the L^* channel unchanged.

Algorithm 1 Simulated annealing in ColorMaker

function computeUserPrefBias($index$, C_j)
  Set P to the array of preference colors (§5.7)
  $\vec{u} \leftarrow [0, 0, 0]$
  for $i \leftarrow 1$ to $length(P)$ **do**
    $\mu \leftarrow P_i.center$; $\sigma \leftarrow \frac{1}{2} P_i.size$
    $\vec{b} \leftarrow [0, P_i.A - C_j.A, P_i.B - C_j.B]$
    $\vec{u} \leftarrow \vec{u} + \text{DNORM}(index, \mu, \sigma)\vec{b}$
  end for
  return $normalize(\vec{u})$
end function

Set n to the number of control points
Set C to an array of n CIE Lab colors
Set initial T_{init} and final T_{end} temperatures, $alpha$ as the cooling rate, and $iterCount$ as the number of iterations

for $i \leftarrow 1$ to n **do** ▷ Initialize C_i to a random Lab color
  $C_i.L^* \leftarrow L^*$ of desired luminance profile at index i
  repeat
    Set $C_i.A$ and $C_i.B$ to random chromatic values
  until C_i is within display gamut
end for

$T \leftarrow T_{init}$
$C_{best} \leftarrow C; E_{best} = E(C)$ ▷ Set current best solution
while $(T \geq T_{end})$ **do**
  for $i \leftarrow 1$ to $iterCount$ **do**
    $j \leftarrow \lfloor random(1, n+1) \rfloor$ ▷ Select an index to perturb
    $\vec{u_j} \leftarrow \text{computeUserPrefBias}(j, C_j)$
    repeat
      $\vec{r} \leftarrow normalize(\ [0, random(0,1), random(0,1)]\ )$
      $\vec{o} \leftarrow normalize(\ 0.4\vec{r} + 0.6\vec{u_j}\ )$
      offset C_j by a small amount in the direction of $\vec{o}$
    until C_j is within display gamut
    $E_{new} \leftarrow E(C)$ ▷ Cost of the new solution
    $\Delta \leftarrow E_{new} - E_{best}$
    if $\Delta \leq 0$ **OR** $random(0,1) < 1/(1 + e^{\frac{\Delta}{T}})$ **then**
      $C_{best} \leftarrow C$ ▷ Adopt new as current best
      $E_{best} \leftarrow E_{new}$
    else
      $C \leftarrow C_{best}$ ▷ Revert to last best solution
    end if
  end for
  $T \leftarrow T * \alpha$
end while

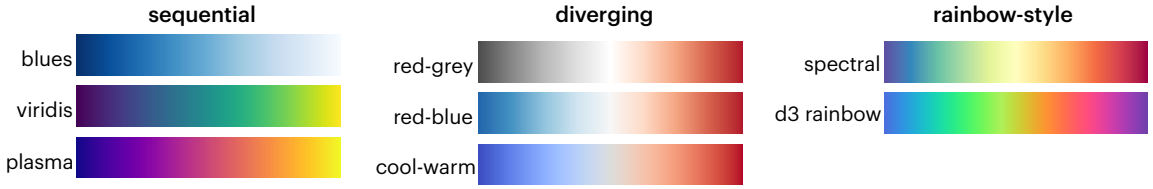

Figure 7: We selected eight commonly used colormaps as benchmarks to compare against. This selection covers three typical design styles: sequential, diverging, and rainbow.

The resulting colormap C is evaluated using the cost function in Equation 1. If the new solution yields a lower cost, it is adopted. If the new solution is worse, it may still be adopted with a probability:

$$p = 1/(1 + e^{\frac{E(C)-E(C_{best})}{T}}) \tag{13}$$

Where $E(C)-E(C_{best})$ is the difference in cost between the new and the current best solution, and T is the temperature. The scheduling function above, which is commonly used in simulated annealing problems [25], ensures a certain probability of accepting a *worse* solution. This behavior allows the algorithm to explore a wider solution space, especially in the earlier, high-temperature cycles of the optimization. Based on prior work [31, 32], we choose an initial temperature value of $T_{init} = 1$ and final temperature $T_{end} = 0.0001$, with a cooling coefficient $\alpha = 0.925$. We set the number of iterations at each temperature level to $iterCount$ = 5,500. This number can be increased or decreased by a slider in the user interface, allowing for either faster or slower (and thus higher quality) optimization, as needed. The pseudo-code is shown in Algorithm 1.

5.7 Operationalizing User Edits and Preferences

The optimization iteratively perturbs the colormap, one color at a time until it converges to a good solution. The perturbation is, by design, stochastic. However, to incorporate user preferences, we can bias the algorithm to favor colors specified by the user. This is done by blending the random vector $\vec{r}$ with a bias vector $\vec{u}$ pointing in the direction of color(s) included in the preference shelf (see §4.1). Specifically, for a control point c_j selected for perturbation, the vector $\vec{u_j}$ will point toward the color (or average of multiple colors) indicated by the user as preference. The magnitude of bias is dictated by the extent of a color block 2σ, and the distance between the control point and the midpoint of the color block μ: the closer j is to the center, the stronger the bias (see Figure 6). Bias magnitude is also modulated by the extent of the preference: a narrower color block exerts a more focused bias concentrated at its center, whereas a wider block conveys a more diffused bias that impacts more control points although to a lesser extent. The overall bias vector $\vec{u_j}$ is an accumulation of weighted biases, which are then blended with $\vec{r}$ at a 60-40% ratio, respectively (line #31 in Algorithm 1). In effect, we allow the algorithm to favor colors indicated by the designer while still leaving a sufficient level of randomness in the optimization.

The above model of *designer preference as bias* enables ColorMaker to operationalize initial preferences and post-optimization refinement in a uniform manner. Specifically, edits, such as adjusting the chroma or hue (see §4.2), are incorporated into the preference shelf and factored into the optimization according to the model above. One difference is that, when re-executing the optimization as a result of edits, we use the previous solution as a starting point, in lieu of a random initial solution. Additionally, we also initialize the algorithm to start at a lower temperature. This enables us to maintain the global structure of the last solution while still responding to local edits. The lower starting temperature also allows for a faster, more interactive optimization. That said, the user can choose to run a full optimization cycle by clicking a 're-run' button.

6 EVALUATION

We evaluate ColorMaker in two ways: First, by analyzing the perceptual characteristics of a representative sample of generated design, and second, through a user study conducted with visualization designers and researchers.

6.1 Analysis of Perceptual Characteristics

In line with previous work [55, 61], we evaluate the quality of ColorMaker's algorithm by comparing its output to popular colormaps. We specifically test if our algorithm can generate colormaps that are perceptually similar to established, best-practice designs. To do this, we generate a large sample of colormaps for each of our test cases. We vary the generation parameters, testing sequential and diverging profiles, low vs. unconstrained hue variation, and CVD-optimization vs. non-optimized generation. We evaluate three characteristics: perceptual uniformity (computed according to Equation 3), smoothness (Equation 6), and perceptual discriminability. The latter is a measure of how distinctive individual colors are relative to all other colors in the scale [12], computed using the following equation:

$$\text{Perceptual discriminability}(C) = \frac{2}{n(n-1)} \sum_{i<j\leq n} \Delta E^{00}(c_i, c_j) \tag{14}$$

This metric measures whether the scale contains colors that are potentially confusable, making it useful to gauge colormap accessibility through simulation, in addition to normal-color vision discriminability.

We selected eight benchmark colormaps (see Figure 7) from a range of reputable sources, including Matplotlib [58], ColorBrewer [22], D3 [7], and one colormap due to Moreland [37]. This

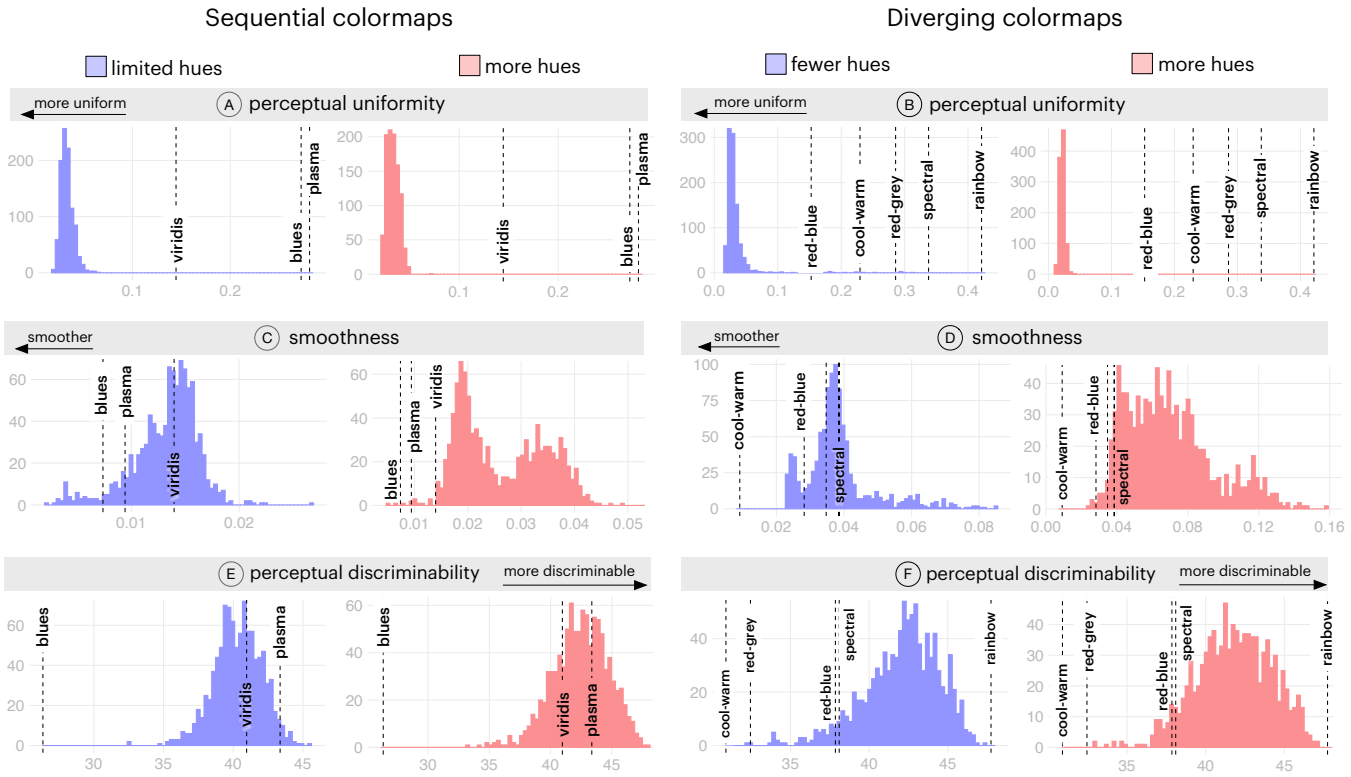

Figure 8: Comparison of perceptual characteristics for generated designs vs. comparable benchmark colormaps. Histograms represent a sample of 1,000 colormaps generated either as sequential (left) or diverging (right). Blue and red represent optimization for either limited or unconstrained hue variation.

selection represents designs often considered by the visualization community to embody good perceptual characteristics. We added D3's rainbow as an upper limit in terms of colorfulness. The latter exhibits a notably smoother profile than traditional rainbows.

We investigate ColorMaker's generation capability focusing on two designs: sequential and diverging colormaps, both commonly used in quantitative visualizations. We also explore high and low colorfulness settings, which directly affect the smoothness penalties detailed in §5.4. To illustrate the impact of this user-controlled parameter, we show two distributions: one with low colorfulness for limited hue variation ($w_{s,2} = 0.9$) and another with more varied colors ($w_{s,2} = 0.25$, ColorMaker's default value). Additionally, we introduce slight randomness in luminance range $[L_0^*, L_1^*]$ to promote design diversity. The range was randomized from $L_0^* = \text{random}(5, 15)$ to $L_1^* = \text{random}(85, 95)$, similar to the luminance profiles found in the benchmarks.

6.1.1 Results. Figure 8 displays the results, featuring histograms of 1,000 generated designs each, obtained under the default optimization setting of 5,500 iterations. Red histograms illustrate high colorfulness (greater hue variation), while blue represents lower colorfulness (fewer hues). We include comparable benchmarks in each plot to contrast their perceptual characteristics against ColorMaker's. Benchmark scores were calculated using the same equations, sampling the scale at n equidistant points ($n = 25$ for sequential and $n = 31$ for diverging, as in ColorMaker's algorithm).

Perceptual Uniformity: For all colormap types (Figure 8-A & B), perceptual uniformity scores for ColorMaker are noticeably better than the benchmarks (lower score is better, as it implies smaller deviation from expected color distance). This is possibly due to the benchmarks having been designed using different distance metrics. For instance, *blues* and *red-grey* were made with equal Euclidean CIE Lab steps, whereas *cool-warm* was created using a polar transformation of the CIE Lab space. That said, the narrow distributions suggest the algorithm can effectively optimize and maintain high perceptual uniformity across a variety of design styles.

Smoothness: As expected, the smoothness scores exhibit variations based on ColorMaker's generation parameters. For instance, in sequential colormaps, restricting hue variation results in distributions that largely overlap with the benchmark colormaps (Figure 8-C, blue). This is particularly true for colormaps like *plasma* and *viridis*. Interestingly, *blues* appear smoother than the majority of ColorMaker's designs, possibly due to its straightforward, single-hue design. As the constraint on hue variation is relaxed, the generated scales tend to be less smooth (red histogram). This trend

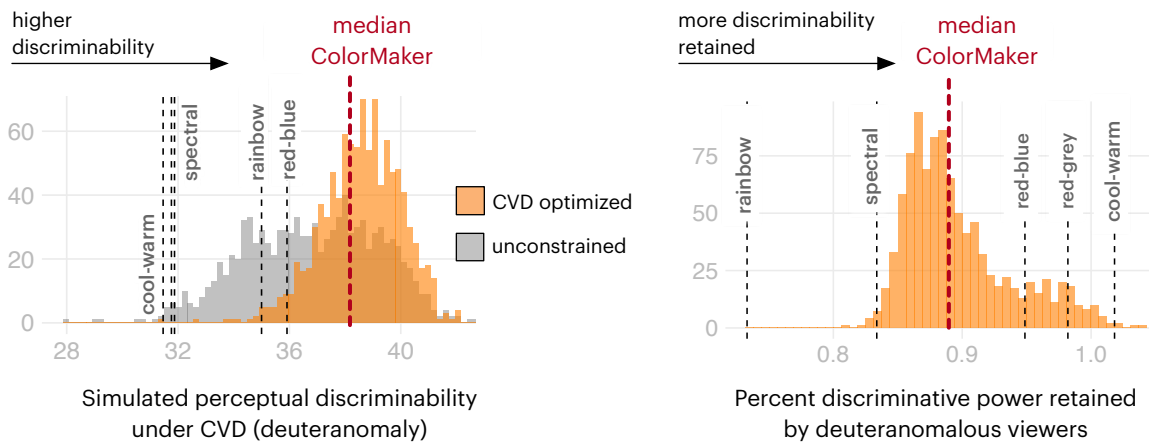

Figure 9: *Left:* Simulated perceptual discriminability for a viewer with deuteranomaly. Colormaps were generated with either CVD-optimization enabled (yellow) or disabled (grey). *Right*: Loss of discriminability due to CVD. A value of 1 indicates no theoretical loss.

is mirrored in diverging colormaps (Figure 8-D), with the red distribution indicating lower smoothness compared to the benchmark colormaps. This suggests that, when hue variation is left unconstrained, the algorithm tends to favor more intricate and vibrant colormaps.

It is worth noting that, when the smoothness penalty is set to nearly maximum, we observe a long-tailed distribution in diverging scales (Figure 8-D, blue histogram). This might reflect occasional difficulty by the algorithm in maintaining fewer hues when indicated by the colorfulness slider. However, this issue is relatively limited: out of 1,000 designs optimized for restricted hues, only 121 (12.1%) scored 0.05 or worse in smoothness (for reference, Brewer's *spectral*[8] scores approximately 0.04). Furthermore, ColorMaker offers an alternative mechanism for constraining hues through the preference shelf (see Figure 3 for an example). Nevertheless, the majority of generated colormaps generally fall within the benchmarks for smoothness, with perhaps the exception of Moreland's *cool-warm* – the latter has been designed to be extraordinarily smooth [37], by way of softening the sharp, midpoint transition that is characteristic of many diverging colormaps (see Figure 7).

The results suggest that ColorMaker can generate a diverse range of single-hue, multi-hue, and diverging colormaps. Depending on the user-controlled colorfulness parameter, the generated colormaps can either exhibit more vibrant colors or attain similar smoothness to comparable benchmarks. Notably, the algorithm appears capable of maintaining good perceptual uniformity even when set to favor high hue variation.

Perceptual Discriminability: Sequential colormaps generated by ColorMaker demonstrate comparable discriminability to multi-hue benchmarks (Figure 8-E). In optimizations favoring hue variation (red), the distribution slightly shifts to the right, resulting in even higher discriminability. However, many of the generated colormaps still fall within the range of well-known multi-hue designs like *viridis* and *plasma*.

In contrast, the generated diverging scales consistently exhibit higher discriminability than the benchmarks (Figure 8-F). This suggests more vibrant designs, at times approaching the characteristics of rainbow scales. One potential reason for this difference is that traditional diverging scales often follow a conservative design, commonly concatenating two single-hued ramps [61]. Although the user can replicate this design by choosing two hues from the preference shelf and stretching them to overlap the two luminance arms, ColorMaker appears to generate more diverse and colorful designs when unconstrained by this strategy, resulting in higher discriminative power.

Discriminability for CVD Viewers: Lastly, we examined the perceptual discriminability of diverging colormaps under CVD simulation. To conduct this analysis, we utilized equation 14, replacing every color pair with their CVD-simulated appearance, as computed using the transform detailed in §5.5. Figure 9-left plots perceptual discriminability as experienced by a viewer with deuteranomaly. Compared to unconstrained generation, CVD optimization (yellow) shifts the distribution to the right, enhancing the accessibility of the generated colormaps. Notably, for the majority of optimized colormaps, the discriminative power as perceived by individuals with CVD surpasses that of comparable benchmarks like Brewer's *red-blue*, known for its CVD-friendliness [22].

Another approach to evaluating the impact of CVD on colormaps is to assess the theoretical loss in discriminative power (Figure 9-right). Typical colormaps generated by ColorMaker retain approximately 88.6% of their original color discriminability, which is higher than problematic colormaps like *rainbow* (73.3%) and *spectral* (83.4%), but lower than hand-crafted designs such as Brewer's *red-blue* and *red-grey* (94.9% and 98.2% respectively), and *cool-warm*, which experiences virtually no loss (101%). This gap reflects a difference in strategy between ColorMaker and other tools, such as ColorBrewer, when it comes to ensuring CVD accessibility. While ColorBrewer follows a heuristic of avoiding large portions of the color space at a time (e.g., green tones [8]), ColorMaker's

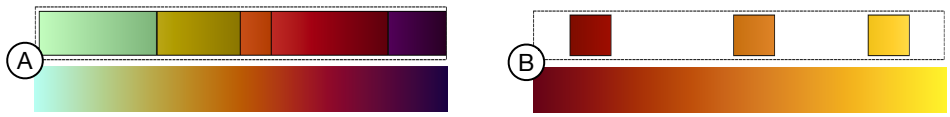

Figure 10: Participants employed two broad approaches when communicating their color preference: (A) 'packing' the preference shelf to dictate precise expectations, or (B) making partial specification and allowing the algorithm to populate the scale.

optimization seeks a maximally accessible middle-ground solution. This results in more colorful designs that, although lose a larger percentage of their discriminative power, still manage to provide higher discriminability for CVD viewers.

6.1.2 Summary. Our analysis indicates that ColorMaker generates accessible designs with perceptual characteristics similar to known benchmarks. In some cases, the generated colormaps even surpass existing designs, achieving better uniformity and higher discriminative power. The range of scores, particularly for smoothness and discriminability, suggests that the algorithm can produce a diverse array of solutions. However, these results are for non-interactive generation, hence we evaluate the latter in a user study.

6.2 User Study

To evaluate ColorMaker's usability and expressive power, we conducted a user study with visualization practitioners and researchers who have varying levels of expertise in color design. The study included a set of structured and open-ended tasks, which were meant to simulate various color design scenarios.

6.2.1 Participants. We recruited 13 participants by email and by posting announcements in relevant forums, including slack channels frequented by computational scientists and visualization practitioners. We limited enrollment to those who create and/or consume visualizations regularly as part of their work. Participants were also required to have used at least one color-design tool in the past (e.g., ColorBrewer). Of the 13 participants we recruited, seven were student researchers in visualization, distributed systems, and neuroscience, four were visualization practitioners, and two were academic visualization researchers. Participants were compensated with a $50 gift card. The study was approved by the Indiana University IRB.

6.2.2 Procedures. The study was conducted remotely via Zoom. Participants interacted with ColorMaker through a standard web browser and shared their screens. We recorded screen content, video, and audio for analysis. We encouraged participants to think aloud and verbalize their design process. At the beginning of the study, we provided a brief tutorial to introduce the tool and demonstrate its various features. Participants then completed three color design tasks. In the first task, participants were asked to recreate two well-known colormaps (*viridis* [58] and *purple-green* [22]) in two separate trials. Reference color scales were displayed in a side panel. In the second task, participants were asked to design appropriate colormaps for three separate datasets that we provided: bathymetry data (scalar field), a slice from an MRI brain scan (scalar field), and air quality data (choropleth). Those datasets were shown on the side of the interface in lieu of the default example visualizations. The final task included a single open-ended trial, prompting participants to experiment and create a colormap they find pleasing, without specifying a target dataset. The study concluded with a semi-structured exit interview, in which participants described their general color design needs and commented on whether they thought ColorMaker would meet those.

6.2.3 Results. Participants completed the study in 50.58 minutes on average (σ = 15.8). Average trial-completion time was 6.09 minutes in the first task (σ = .9). Tasks 2 and 3 took slightly less time, 5.03 (σ = .48) and 4.95 (σ = .87) minutes on average, respectively. We analyzed the video and audio recordings to understand participants' usage patterns. We also analyzed their comments to understand what features participants were able to utilize effectively, and what challenges they faced while interacting with the tool. Several key themes emerged from the study:

Colormap Generation and Ideation: Several participants expressed satisfaction with the colormaps generated by the algorithm. For example, *P13* remarked that "the optimization under the hood does a great job of smoothing things out, especially when it gets to the final tweaks." Participants appreciated the tool's ability to ideate new designs, noting that the approach opens new design avenues for them. For instance, *P9* said: "Right now, the main way I tweak is to flip through the default [color] maps till I find something close and then adjust them by hand. I see myself using this [ColorMaker] to generate relatively quickly, entire new colormaps." Some remarked that the generated designs are immediately usable in their own visualizations: "I'm pretty happy with the smoothness of the colors it generates. It will be good for heatmap-like data that I would use." (*P3*).

Specifying Preference: Most participants found ColorMaker's model for specifying color preferences to be intuitive. This included the ability to add, move, 'stretch', and remove colors. For example, *P9* commented, "The clicking, dragging, sliding [of color preferences] was all very intuitive. I enjoyed that." *P5* also commented on the ease of eliminating unwanted colors from the scale: "Some things are very easy to use, like if I want to trash this [color], I don't even have to worry about it and just drag it out and it's gone."

There were broadly two different strategies used by participants to influence how the colormap should look. The first is to almost completely 'pack' the preference shelf with all the colors the participant wanted to see (see Figure 10-A). Four of the 13 participants employed this strategy of over-specifying the design. When asked about this approach, *P8* said: "I am thinking of the color in my head like when I am doing the visualization... If I have spaces, the algorithm may generate some different color which I am not imagining." This suggests a desire by some participants to exert fine-grained control over the algorithm.

A second, more prevalent usage pattern was to provide an incomplete specification and let the algorithm "fill in the gaps." Participants most commonly specified two or three desired colors and waited to see the initial optimization result (see Figure 10-B for an example). Those who used this strategy generally thought the algorithm incorporated effective colors to complete the scale. For example, *P1* remarked: "After dragging and dropping a color, the generated color map makes sense and looks really pleasing, nice, smooth." Another participant, *P2*, was pleasantly surprised with the result when, after adding only two colors to the preference shelf, was able to obtain a scale that matched their intuition: "What's interesting is that it [the algorithm] introduced a new color that I was thinking of putting in there anyway."

Refining Generated Designs: Participants used a variety of approaches to refine the generated designs. One frequent strategy was to adjust the preference shelf after an initial optimization run, by adding new colors or by repositioning and resizing existing color blocks. *P13* commented on the ease of editing: "I like the way the picker is set up in terms of being able to edit.... just like being able to tweak and tune." Different participants were also able to achieve similar design outcomes in different ways. For example, while *P5* stretched a blue color to cover the entire scale (similar to the example in Figure 3), *P7* achieved a similar colormap using two hues as input and making adjustments to the colorfulness slider. In both cases, the algorithm responded by generating a single-hue, predominantly blue design.

Several participants performed quick adjustments to the scale using the inline suggestion palette, which allows relative adjustment of hue or chroma at specific point in the scale. *P13* commented on the usefulness of this feature: "I really like the ability to get some suggestions when you're hovering [over the colormap] like this, because there were times where it's like, do I really want to tune out and pick a shade? And it's like no, I can just do this [hover] and get that suggestion. I thought that was really handy when in the drafting mode."

Optimization for CVD: Of the 13 participants, seven used the built-in CVD optimization option to ensure accessibility. *P10* commented on the usefulness of this feature to their visualization practice: "I have to take into account if it is going to be printed and work well for CVD. So having the option when generating colors is nice...". The participant went on to say: "For the software I use, I don't have this [CVD] option, so it was good to have that. It was intuitive to use as well." Others also appreciated the ability to get an immediate CVD simulation: "I really enjoyed the color vision checker because that's something I keep in mind and a lot of tools don't have anything like this." (*P9*) and "[the] CVD panel for this [ColorMaker] has more options. ColorBrewer has only one option: select colorblind safe maps and that's it." (*P8*).

Challenges and Suggestions for Improvement: Despite the largely positive feedback, a few participants encountered challenges with the tool. One particular aspect of ColorMaker that seemed to confuse some is the luminance-as-hard-constraint model. This confusion often manifested during interaction with the preference shelf, and after a participant has added a color, only to see ColorMaker automatically adjust the luminance of that color to conform to the selected profile. Other participants had expected the tool to perform a linear interpolation between two colors by default. However, this was not typically the case, as the algorithm would frequently bend the colormap to improve smoothness while conforming to the user-specified hues. These examples seem to point to a possible gulf between some participants' mental models and the algorithm. The majority of participants appeared to grasp ColorMaker's mixed-initiative paradigm after a few interactions, and ultimately came to appreciate its expressiveness. However, the initial confusion experienced by some suggests more could be done to orient new users to the tool.

One feature that was used sparsely is the colormap curve editor (Figure 2-E), which, for many, seemed too "complex" to "mess with". *P2* said: "I was a little bit intimidated by the curves. I have worked in curves before, but not all colors at once... Those were a bit tough for me to get into, at least at the first go, maybe [with] a bit more experience, I'd understand it better." This suggests a need to redesign this interaction to be more forgiving.

Although ColorMaker was envisioned to fill a gap in design tools for continuous colormaps, several participants opined that they would have liked the inclusion of categorical and discrete, ordered palettes. Moreover, certain design features cannot be easily expressed with ColorMaker, which participants saw as a limitation. This includes color bands which were desired by some: "I would like a sharper contrast between two colors almost like a contour itself, like a white line" (*P3*). Future work could add support for a wider variety of designs, including the ability to create sharp transitions along with support for rainbow-style colormaps (e.g., Turbo [35]).

One potential barrier to iterative refinement is the need to wait on the algorithm. On a laptop computer and with default generation parameters, it can take approximately eight seconds to complete a full optimization cycle (CVD modeling can take longer, though). To maintain interactivity, ColorMaker runs the optimization in the background and renders intermediate results, while also allowing the user to interrupt (e.g., with edits). One participant suggested a "fast updating draft mode" which would allow even faster design iteration. A current workaround involves decreasing the number of optimization iterations, which is controllable through a slider. Future work could focus on improving the efficiency of the algorithm, or on developing a parallelizable optimization and leveraging faster hardware (e.g., GPUs).

7 LIMITATIONS AND FUTURE WORK

Feedback from participants suggests that ColorMaker has successfully met the design requirements outlined in §3. However, there are certain limitations to the current approach that warrant consideration. First, unlike techniques employing deterministic generation models [61], our algorithm relies on a stochastic, simulated annealing process. While the majority of generated colormaps appear to be comparable to benchmarks, there is a possibility of generating suboptimal designs. Therefore, it is essential for designers to evaluate the generated colormaps before adoption, including assessing the results of CVD simulation. The ColorMaker interface is designed to facilitate ideation, evaluation, and swift refinement of designs, but

future enhancements could focus on ensuring more robust generation, possibly through automated post-optimization checks. These checks might involve comparing the generated colormap to known benchmarks (i.e., similar to the analysis in §6.1). Colormaps falling significantly short of benchmarks could be automatically discarded, with the algorithm generating new solutions. Similarly, although we offer the option to generate multiple colormaps simultaneously to promote ideation, the stochastic algorithm does not preclude the generation of repeated or highly similar designs. Future work could address this by measuring design similarity, and by introducing random biases to actively herd the algorithm into generating more diverse solutions.

While ColorMaker strives to meet users' personal preferences, it currently lacks an *empirical* model of color preference. In other words, the optimization does not consider whether a particular colormap is aesthetically pleasing. Aesthetics is essential, especially because designers often base their color choices on visual appeal [14, 29]. Future work could incorporate an empirical aesthetics model into the objective scoring function. Moreover, there is room to optimize for additional cognitive factors, like color nameability [24, 46], thus enabling the generation of designs with more distinct and recognizable color names.

Lastly, the ColorMaker algorithm currently depends on several parameters, including the number of control points (n). These parameters are operationalized into the objective cost functions (e.g., the smoothness penalty function uses n and $\frac{n}{2}$ samples to estimate curvature), and will therefore influence the quality of the results. Future work should attempt to characterize the impact of these parameters on colormap generation and, where possible, reduce their number to a minimum. The current implementation is also limited in the range of luminance profiles (and consequently colormap designs) it can generate. We believe that this limitation can be overcome with minor adaptations. Future implementations could also permit some flexibility in meeting the requested constraints, including minor deviations from a strict luminance profile. Such pliability would allow for the generation of a wider diversity of designs, including rainbows, which have merit in certain cases [43, 45, 60].

8 CONCLUSION

Although numerous tools cater to the creation of categorical color palettes, designing quantitative colormaps for visualization is less supported. This gap led us to introduce ColorMaker, a mixed-initiative approach for the generation and customization of continuous colormaps. ColorMaker employs simulated annealing to produce a diverse range of colormap styles, including CVD-friendly designs. It allows for incorporating user color preferences, enabling iterative colormap refinement. Evaluation confirms that this approach consistently delivers high-quality colormaps, on par with established designs. Furthermore, a user study demonstrates ColorMaker's intuitiveness and its potential for empowering visualization designers to create colormaps for various needs.

ACKNOWLEDGMENTS

This paper is based upon research supported by the National Science Foundation under award 1942429. MEP and KR were also supported in part by the Office of Science, U.S. Department of Energy, under contract DE-AC02-06CH11357. KL and YW are supported in part by NSF China (No. 62132017, 62141217), and the Shandong Provincial Natural Science Foundation (No. ZQ2022JQ32).

REFERENCES

[1] EHL Aarts. 1989. *A stochastic approach to combinatorial optimization and neural computing*. John Wiley, United States.
[2] Adobe. 2014. Adobe Color CC. https://color.adobe.com/de/create/color-wheel/
[3] Katrin Angerbauer, Nils Rodrigues, Rene Cutura, Seyda Öney, Nelusa Pathmanathan, Cristina Morariu, Daniel Weiskopf, and Michael Sedlmair. 2022. Accessibility for color vision deficiencies: Challenges and findings of a large scale study on paper figures. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–23.
[4] Utkarsh Ayachit. 2015. *The paraview guide: a parallel visualization application*. Kitware, Inc., Clifton Park, NY, USA.
[5] Fabrizio Bianchi. 2023. Coolors. https://coolors.co. https://coolors.co [Online; accessed 3-September-2023].
[6] David Borland and Russell M Taylor Ii. 2007. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications* 27, 2 (2007), 14–17.
[7] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2301–2309.
[8] Cynthia Brewer. 1997. Spectral schemes: Controversial color use on maps. *Cartography and Geographic Information Systems* 24, 4 (1997), 203–220.
[9] Cynthia A Brewer. 1994. Color use guidelines for mapping. *Visualization in modern cartography* 1994, 123-148 (1994), 7.
[10] Cynthia A Brewer. 1996. Guidelines for selecting colors for diverging schemes on maps. *The Cartographic Journal* 33, 2 (1996), 79–86.
[11] Roxana Bujack, Terece L Turton, David H Rogers, and James P Ahrens. 2018. Ordering perceptions about perceptual order. In *2018 IEEE Scientific Visualization Conference (SciVis)*. IEEE, IEEE, Berlin, Germany, 32–36.
[12] Roxana Bujack, Terece L Turton, Francesca Samsel, Colin Ware, David H Rogers, and James Ahrens. 2017. The Good, the Bad, and the Ugly: A Theoretical Framework for the Assessment of Continuous Colormaps. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2017), 923–933.
[13] Canva. 2023. Canva. https://www.canva.com/colors/color-palette-generator/. https://www.canva.com/colors/color-palette-generator/ [Online; accessed 3-September-2023].
[14] Nick Cawthon and Andrew Vande Moere. 2007. The effect of aesthetic on the usability of data visualization. In *2007 11th International Conference Information Visualization (IV'07)*. IEEE, IEEE, Zurich, Switzerland, 637–648.
[15] Colormagic. 2023. Colormagic. https://colormagic.app/. https://colormagic.app/ [Online; accessed 3-September-2023].
[16] Colormind. 2023. Colormind. http://colormind.io/. http://colormind.io/ [Online; accessed 3-September-2023].
[17] Fabio Crameri, Grace E Shephard, and Philip J Heron. 2020. The misuse of colour in science communication. *Nature Communications* 11, 1 (2020), 1–10.
[18] Sebastian Deterding, Jonathan Hook, Rebecca Fiebrink, Marco Gillies, Jeremy Gow, Memo Akten, Gillian Smith, Antonios Liapis, and Kate Compton. 2017. Mixed-initiative creative interfaces. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 628–635.
[19] Deborah Dixon. 2022. In the breach: feeling the heat of climate change. *Scottish Geographical Journal* 139, 1-2 (2022), 1–12.
[20] Connor Gramazio. 2017. Chromaticity. https://gramaz.io/chromaticity/. https://gramaz.io/chromaticity/ [Online; accessed 8-September-2023].
[21] Connor C Gramazio, David H Laidlaw, and Karen B Schloss. 2017. Colorgorical: Creating discriminable and preferable color palettes for information visualization. *IEEE Transactions on Visualization and Computer Graphics* 23, 1 (2017), 521–530.
[22] Mark Harrower and Cynthia A Brewer. 2003. ColorBrewer. org: an online tool for selecting colour schemes for maps. *The Cartographic Journal* 40, 1 (2003), 27–37.
[23] Jeffrey Heer. 2019. Agency plus automation: Designing artificial intelligence into interactive systems. *Proceedings of the National Academy of Sciences* 116, 6 (2019), 1844–1850.
[24] Jeffrey Heer and Maureen Stone. 2012. Color naming models for color selection, image editing and palette design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1007–1016.
[25] Darrall Henderson, Sheldon H. Jacobson, and Alan W. Johnson. 2003. *The Theory and Practice of Simulated Annealing*. Springer US, Boston, MA, 287–319. https://doi.org/10.1007/0-306-48056-5_10
[26] GT Herman and H Levkowitz. 1992. Color scales for image data. *IEEE Computer Graphics and Applications* 12, 1 (1992), 72–80.

[27] Eric Horvitz. 1999. Principles of mixed-initiative user interfaces. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 159–166.

[28] Mathieu Jacomy. 2023. I Want Hue. https://medialab.github.io/iwanthue/. https://medialab.github.io/iwanthue/ [Online; accessed 3-September-2023].

[29] Andrea Lau and Andrew Vande Moere. 2007. Towards a model of information aesthetics in information visualization. In *2007 11th International Conference Information Visualization (IV'07)*. IEEE, IEEE, Zurich, Switzerland, 87–92.

[30] Zhicheng Liu, John Thompson, Alan Wilson, Mira Dontcheva, James Delorey, Sam Grigg, Bernard Kerr, and John Stasko. 2018. Data Illustrator: Augmenting vector design tools with lazy data binding for expressive visualization authoring. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–13.

[31] Kecheng Lu, Mi Feng, Xin Chen, Michael Sedlmair, Oliver Deussen, Dani Lischinski, Zhanglin Cheng, and Yunhai Wang. 2020. Palettailor: Discriminable colorization for categorical data. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2020), 475–484.

[32] Kecheng Lu, Khairi Reda, Oliver Deussen, and Yunhai Wang. 2023. Interactive Context-Preserving Color Highlighting for Multiclass Scatterplots. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, 1–15.

[33] Susie Lu and Elijah Meeks. 2023. Viz Palette. https://projects.susielu.com/viz-palette. https://projects.susielu.com/viz-palette [Online; accessed 3-September-2023].

[34] Gustavo M Machado, Manuel M Oliveira, and Leandro AF Fernandes. 2009. A physiologically-based model for simulation of color vision deficiency. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1291–1298.

[35] Anton Mikhailov. 2019. Turbo, An Improved Rainbow Colormap for Visualization. https://ai.googleblog.com/2019/08/turbo-improved-rainbow-colormap-for.html. [Online; accessed 4-December-2020].

[36] Sebastian Mittelstädt, Dominik Jäckle, Florian Stoffel, and Daniel A Keim. 2015. Colorcat: Guided design of colormaps for combined analysis tasks. In *Conference on Visualization (EuroVis)*. The Eurographics Association, Cagliari, 115–119.

[37] Kenneth Moreland. 2009. Diverging color maps for scientific visualization. In *International Symposium on Visual Computing*. Springer, Springer Berlin Heidelberg, Berlin, Heidelberg, 92–103.

[38] Kenneth Moreland. 2016. Why We Use Bad Color Maps and What You Can Do About It. *Electronic Imaging* 2016, 16 (2016), 1–6.

[39] Pascal Nardini, Min Chen, Michael Böttinger, Gerik Scheuermann, and Roxana Bujack. 2021. Automatic Improvement of Continuous Colormaps in Euclidean Colorspaces. *Computer Graphics Forum* 40, 3 (2021), 361–373. https://doi.org/10.1111/cgf.14313

[40] P. Nardini, M. Chen, F. Samsel, R. Bujack, M. Böttinger, and G. Scheuermann. 2019. The Making of Continuous Colormaps. *IEEE Transactions on Visualization and Computer Graphics* 27, 6 (2019), 1–1. https://doi.org/10.1109/TVCG.2019.2961674

[41] Jamie R Nuñez, Christopher R Anderton, and Ryan S Renslow. 2018. Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data. *PLoS ONE* 13, 7 (2018), e0199239.

[42] Matthew A. Petroff. 2021. Accessible Color Sequences for Data Visualization. arXiv:2107.02270 [cs.GR]

[43] Khairi Reda. 2022. Rainbow Colormaps: What are they good and bad for? *IEEE Transactions on Visualization and Computer Graphics* 29, 12 (2022), 5496–5510.

[44] Khairi Reda, Pratik Nalawade, and Kate Ansah-Koi. 2018. Graphical perception of continuous quantitative maps: the effects of spatial frequency and colormap design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, Association for Computing Machinery, New York, NY, USA, 272.

[45] Khairi Reda and Michael E. Papka. 2019. Evaluating Gradient Perception in Color-Coded Scalar Fields. In *2019 IEEE Visualization Conference (VIS)*. IEEE, IEEE, Vancouver, BC, Canada, 271–275.

[46] Khairi Reda, Amey A. Salvi, Jack Gray, and Michael E. Papka. 2021. Color Nameability Predicts Inference Accuracy in Spatial Visualizations. *Computer Graphics Forum* 40, 3 (2021), 49–60. https://doi.org/10.1111/cgf.14288

[47] Khairi Reda and Danielle Albers Szafir. 2021. Rainbows Revisited: Modeling Effective Colormap Design for Graphical Inference. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 1032–1042. https://doi.org/10.1109/TVCG.2020.3030439

[48] Donghao Ren, Bongshin Lee, and Matthew Brehmer. 2018. Charticulator: Interactive construction of bespoke chart layouts. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2018), 789–799.

[49] Bernice E Rogowitz and Alan D Kalvin. 2001. The "Which Blair Project": a quick visual method for evaluating perceptual color maps. In *Visualization, 2001. VIS'01. Proceedings*. IEEE, IEEE, San Diego, CA, USA, 183–556.

[50] Bernice E Rogowitz and Lloyd A Treinish. 1998. Data visualization: the end of the rainbow. *IEEE Spectrum* 35, 12 (1998), 52–59.

[51] Bernice E Rogowitz, Lloyd A Treinish, Steve Bryson, et al. 1996. How not to lie with visualization. *Computers in Physics* 10, 3 (1996), 268–273.

[52] Francesca Samsel. 2022. SciVisColor. https://sciviscolor.org. https://sciviscolor.org [Online; accessed 23-March-2022].

[53] Francesca Samsel, Sebastian Klaassen, and David H Rogers. 2018. Colormoves: Real-time interactive colormap construction for scientific visualization. *IEEE Computer Graphics and Applications* 38, 1 (2018), 20–29.

[54] Arvind Satyanarayan and Jeffrey Heer. 2014. Lyra: An Interactive Visualization Design Environment. *Computer Graphics Forum* 33, 3 (2014), 351–360. https://doi.org/10.1111/cgf.12391

[55] Stephen Smart, Keke Wu, and Danielle Albers Szafir. 2019. Color Crafting: Automating the Construction of Designer Quality Color Ramps. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2019), 1215–1225.

[56] Martijn Tennekes and Marco J. H. Puts. 2023. cols4all: a Color Palette Analysis Tool. In *EuroVis 2023 - Short Papers*. The Eurographics Association, Leipzig, 37–41. https://doi.org/10.2312/evs.20231040

[57] Ellert van der Velden. 2020. CMasher: Scientific colormaps for making accessible, informative and 'cmashing' plots. *arXiv preprint arXiv:2003.01069* 5, 46 (2020), 2004.

[58] Stéfan van der Walt and Nathaniel Smith. 2015. Matplotlib colormaps. https://bids.github.io/colormap/. [Online; accessed 20-April-2020].

[59] C Ware. 1988. Color sequences for univariate maps: Theory, experiments and principles. *IEEE Computer Graphics and Applications* 8, 5 (1988), 41–49.

[60] Colin Ware, Maureen Stone, and Danielle Albers Szafir. 2023. Rainbow Colormaps Are Not All Bad. *IEEE Computer Graphics and Applications* 43, 3 (2023), 88–93.

[61] Martijn Wijffelaars, Roel Vliegen, Jarke J. Van Wijk, and Erik-Jan Van Der Linden. 2008. Generating Color Palettes using Intuitive Parameters. *Computer Graphics Forum* 27, 3 (2008), 743–750. https://doi.org/10.1111/j.1467-8659.2008.01203.x

[62] Georgios N. Yannakakis, Antonios Liapis, and Constantine Alexopoulos. 2014. Mixed-initiative co-creativity. In *9th International Conference on the Foundations of Digital Games*. Foundations of Digital Games, Fort Lauderdale, 1–8.

[63] Liang Zhou and Charles D Hansen. 2016. A survey of colormaps in visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 8 (2016), 2051–2069.