$\mathbb{C}^3$-palette: Co-saliency based Colorization for Comparing Categorical Visualizations

ANONYMOUS AUTHOR(S)

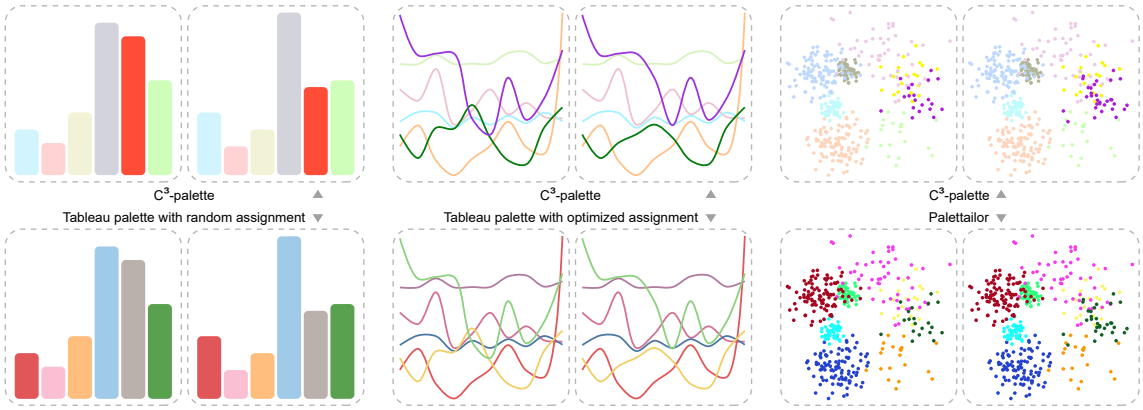

Fig. 1. Results for different types of categorical data visualizations: (left) C^3-palette versus Tableau palette with random assignment; (center) C^3-palette versus Tableau palette with optimal discrimination assignment [46]; (right) C^3-palette versus Palettailor [32]. Our co-saliency methods (top) can highlight the changed classes while maintaining discrimination of classes.

Visual comparison within juxtaposed views is an essential part of interactive data analysis. In this paper, we propose a co-saliency model to characterize the most co-salient features among juxtaposed labeled data visualizations while maintaining class discrimination in the individual visualizations. Based on this model, we present a comparison-driven color design framework, enabling the automatic generation of colors that maximizes co-saliency among juxtaposed visualizations for better identifying items with the largest magnitude change between two data sets. We conducted two online controlled experiments to compare our colorizations of bar charts and scatterplots with results produced by existing single view-based color design methods. We further present an interactive system and conduct a case study to demonstrate the usefulness of our method for comparing juxtaposed line charts. The results show that our approach is able to generate high quality color palettes in support of visual comparisons of juxtaposed categorical visualizations.

CCS Concepts: • **Human-centered computing** → **Information visualization**.

Additional Key Words and Phrases: Color Palette, Visual Comparison, Multi-Class, Juxtaposition

ACM Reference Format:

ANONYMOUS AUTHOR(S). 2018. $\mathbb{C}^3$-palette: Co-saliency based Colorization for Comparing Categorical Visualizations. In . ACM, New York, NY, USA, 22 pages. https://doi.org/10.1145/1122445.1122456

Manuscript submitted to ACM

1 INTRODUCTION

Comparison is an indispensable task in data analysis and visualization. It often involves searching for categories (classes) with large or small changes among multiple categorical datasets. Comparison are usually achieved through juxtaposition of multiple categorical visualizations [15, 33] such as bar charts, line charts or multi-class scatterplots, where each category is commonly encoded by a unique color. While color codings are known to play an important role in helping viewers see differences between juxtaposed views [1, 15, 45], there is no color design scheme that optimizes visual comparisons, especially for the task of identifying the largest differences between charts [37].

A typical scenario would be a market analyst who uses comparisons to investigate the performance of a company across different countries over the last couple years. S/he first would create a scatterplot for each year by showing the annual revenue and profit of each product by colorizing each point of the plot with a country label. After finding the two countries with the largest changes, s/he then examines the annual and monthly profits of various products in these countries with bar and line charts. Using the product name for color encoding, s/he is able to efficiently search the product with the largest differences from side-by-side shown bar and line charts.

The most common way to colorize juxtaposed views is to manually find a color mapping for a selected view while judging how well it fits to the other views. Such a trial and error procedure might converge to a desirable color mapping; however, the needed effort significantly increases with the number of classes and views. Although existing automated color selection approaches [6, 32, 46] allow to alleviate the effort for single view colorizations, the obtained color mapping might not be able to clearly reveal similarities or differences among multiple views. For example, the assignment of the Tableau palette for maximizing class separability (cf. [46]) in Fig. 1 (center bottom) creates a visualization with better class discrimination, but the changed time series (see green and yellow curves in the center bottom) are hard to identify. Although such classes of interest could be highlighted by fading out background classes using alpha blending, this inevitability would introduce visual ambiguities for overlapping classes [3] and potentially lead to poor class separation. As far as we know, few existing visualization-oriented color selection tools (e.g., ColorBrewer [18] or Palettailor [32]) allow for colorizing multi-view visualizations, let alone supporting comparisons in juxtaposed views.

To fill this gap, we propose a comparison-driven color palette generation framework, which automatically generates appropriate color mappings for efficiently searching the largest differences from one categorical visualization to another. To achieve this goal, we propose a co-saliency model to characterize the most salient features among juxtaposed categorical visualizations that are likely to attract visual attention. We borrow the idea from the concept of image co-saliency [22], which was originally designed for summarizing salient differences between two similar natural images. Our co-saliency model allows to easily identify important features (e.g., changed classes) from juxtaposed categorical visualizations while maximizing the visual discrimination of classes in the individual visualizations. It is achieved by fusing two separate goals: class importance between visualizations and class contrast within them. Class contrast is based on perceptual separability between neighboring classes and with the background [46], while class importance is measured by summing up the changes of point positions and point numbers of each class, where the position change is quantified by using the Earth Mover's Distance (EMD) [39], a perceptual distance metric. Classes with large importance and small class separability (strong overlap with other classes) are more co-salient, while classes with small importance or large separability (more compact) are less co-salient.

By integrating our co-saliency model into existing categorical data colorization tools [32], we can automatically generate color mappings that maximize co-saliency among juxtaposed visualizations. The resulting color mapping schemes let classes with large importance pop out from the context and will attract viewers' attention, while at the same

time maximizing the perceptual separability between classes in the individual visualizations. By doing so, the major issue of a juxtaposition, that humans have limited visual memory (see [44]), is greatly alleviated and visual searches can be performed with less cognitive costs [19]. The top of Fig. 1 shows the results generated by our colorization method, where the changed classes pop out and can easier be spotted than the ones in the bottom of Fig. 1. Our results are similar to the ones of alpha blending, but still maintain the separability between classes due to the different hues.

We evaluated our approach through carefully designed bar charts and scatterplots by comparing our colorized results with the ones produced by state-of-the-art palettes (e.g.,Tableau [43] and Palettailor [32]). For bar charts, we replicated the experimental setting of Ondov et al. [37] but only performed the task of identifying a maximum difference from two horizontal juxtaposed bar charts, which is also referred as the *max delta task* by Ondov et al. [37]. Next, we carried out studies with multi-class scatterplots generated by Lu et al. [32], whose counterparts were generated by changing the properties (point number and position) of several randomly selected classes. We first conducted a pilot study to verify the validity of our experimental setting and then ran two online studies: first, we investigated how well our generated palettes help users to identify changed classes of two scatterplots and second, we let them count class numbers in single scatterplots (discrimination task). Lastly, we conducted a case study to demonstrate how our system helps comparing juxtaposed visualizations with multiple line charts. The results show that our approach is able to produce color mappings optimized for supporting comparison and aligned with the state-of-the-art palettes in maximizing perceptual class separability.

A web-based color design tool, $\mathbb{C}^3$-palette[1], named by **C**o-saliency based **C**olorization for **C**omparing multi-class scatterplots, allows to show coordinated views and let users explore the relationship between multiple data sets with different color mapping schemes. The main contributions of this paper are as follows:

- We propose a multi-class data visualization co-saliency model for measuring the importance of each data item shown in juxtaposed visualizations and use this metric to automatically generate color mapping schemes for effective comparisons;
- We evaluate the effectiveness of the resulting color mapping schemes in supporting both, visual comparison and visual discriminability, with three online user studies (Section 4) and a case study (Section 5.1); and
- We provide an interactive tool that demonstrates how our approach can be used for visually comparing multiple juxtaposed visualizations.

2 RELATED WORK

We divide previous works into those related to visual comparisons, color design for visualization, and visual saliency/co-saliency.

2.1 Visual Comparison

Visual comparison is an essential part of interactive data analysis, which is regarded as a high-level "compound task." Gleicher et al. [16] provide a systematic review of techniques developed for supporting comparisons, three basic layout designs for comparative visualization are found: *juxtaposition*, *superposition* and *explicit encoding*. Among them, juxtaposition places different datasets in separate views without changes to the original visualization design due to its simplicity it is used in many applications [1, 31, 36]. However, such a design often creates cognitive burden because

[1] https://c3-palette.github.io/

users need to maintain a mental image of one view for comparing it with another view [33]. Recently, Ondov et al. [37] and Jardine et al. [24] evaluated the perceptual effectiveness of different layouts for the comparison of bar charts with a few low-level tasks. They show that juxtaposition is less effective for tasks like finding the "biggest delta between items". Accordingly, Gleicher et al. [16] and L'Yi et al. [33] both suggested to carefully design visual encoding for improving their effectiveness. Therefore, our method facilitates visual comparison of categorical data by improving visual search using a pop-out effect [11] induced by our proposed color mapping scheme.

2.2 Color Design

For a complete review of color design techniques for visualization, we refer readers to surveys such as [45, 50]. We limit our discussion to techniques related to color design for categorical data visualization and specifically to the optimization of color mappings, color palette generation, and color design for multi-view visualization.

Color Mapping Optimization. Mapping each class to a proper color selected from a given palette is particularly helpful for categorical data visualization, since here no given order can be used. A few factors have been identified for guiding searches within such mappings. For example, Lin et al. [29] proposed to optimize the compatibility between class semantics and the assigned colors. Setlur and Stone [40] produced better results by using co-occurrence measures of color name frequencies. Kim et al. [25] incorporated color aesthetics and contrast into the optimization of color assignment for image segments. Recently, Wang et al. [46] proposed to maximize class discriminability based on color-based class separability, which takes into account spatial relationships between classes and the contrast with the background color. Once an assignment is done, the color of each class can be further optimized to better serve different purposes, such as reducing power consumption of displays [7] , improving the accessibility of visualizations for visually impaired users [34], and better class discrimination [28]. Almost all these methods aim to generate effective visualizations for single data sets, whereas our goal is to efficiently visualize salient class differences across multiple datasets with the same label information. One example is instances of the same dataset over time.

Color Palette Generation. To create an appropriate categorical color palette, the commonly used approach is to select one from a library of carefully designed palettes provided by online tools (e.g. ColorBrewer [18]). Colorgorical [17] further allows users to customize color palettes by generating them based on user-specified discriminability and preference importance. Recently, Palettailor [32] takes a further step by automatically generating categorical palettes for different types of charts, such as scatterplots, line and bar charts. However, all the aforementioned methods deal with single datasets, while our work focuses on visual comparisons within multiple datasets with some changed instances.

Multi-view Color Design. Multi-view visualizations are commonly used in multivariate analysis. Although a few design guidelines [47] have been proposed for constructing multi-view visualizations, few of them are related to color design. Qu et al. [38] recommended a set of color consistency constraints across views. Among them, is a high-level constraint that the same data field should always be encoded in the same way, which is related to our studied comparative visualization. Namely, all juxtaposed views should have the same color mapping scheme and a good scheme is able to help for seeing the differences between views. However, few works have been done for finding such schemes. The only exception is comparing multiple continuous scalar fields [45] with a global color map by merging overlapping value ranges in different datasets. Our work is the first to generate appropriate color mapping for comparing multiple categorical visualizations.

2.3 Visual Saliency & Co-saliency

Here we briefly review visual saliency models developed for visualizations and image co-saliency models.

Saliency for Visualization. The human visual system enables viewers to concentrate on salient regions of an image while ignoring others. This is guided by two major factors [9]: pre-attentive, bottom-up focus based on visual features (e.g., color, intensity and edges) and task-driven, top-down attention based on prior knowledge. Numerous saliency models [4] have been developed to mimic the bottom-up attention mechanism in computer vision. Most of them model image saliency as the contrast of image regions to their surroundings with low level features. Among them, the most influential one is the Itti model [21], which computes image saliency with differences surrounding a central region. Kim et al. [26] tailored this model to increase the visual saliency of selected regions within a volume dataset. Jänicke and Chen [23] employed Itti's model [21] to define a quality metric for evaluating visualizations. Recently, Matzen et al. [35] evaluated a variety of saliency models on a large dataset and explored why these models work poorly for visualizations. One major reason is that visualizations are often created for specific goals, whereas existing models are based on bottom-up attention. To overcome these weaknesses, they proposed a data visualization saliency (DVS) model by incorporating meaningful high-level features into Itti's model. However, this model is not designed on a class-level and cannot be directly used for categorical visualizations.

Image Co-Saliency. Unlike single image based saliency models, the co-saliency model estimates the saliency (importance) of each pixel within the context of related images. Jacobs et al. [22] developed a first co-saliency model for highlighting the most salient differences between two images. Later, this concept was extended for discovering common and salient objects/foregrounds from image collections [49]. Inspired by the original model [22], our work attempts to design an appropriate color mapping for visualizing the most co-salient features among juxtaposed labeled data visualizations. Following their findings that the co-salient features can be effectively characterized by fusing image changes and single image contrast, our co-saliency model relies on two factors: class contrast in the individual views and global features from in-between views (e.g., changes in the class structure).

3 CO-SALIENCY BASED COLOR DESIGN

Given N ($N \geq 2$) categorical visualizations with the same class labels (or a subset thereof), the jth visualization has M classes and n_j data items $\{\mathbf{x}_1^j, \cdots, \mathbf{x}_{n_j}^j\}$, where each $\mathbf{x}_t^j$ has a label $l(\mathbf{x}_t^j)$ and the i-th class (with n_i^j data points) consists of $\{\mathbf{x}_{i,1}^j, \cdots, \mathbf{x}_{i,n_i^j}^j\}, i \in \{1, \cdots, m\}$. For standard bar and line charts, M is equal to n_j but it is often smaller than n_j for scatterplots. All visualizations use the same background color $\mathbf{c}_b$ and the same color mapping scheme $\tau : L \mapsto c$. Our goal is to find the best mapping τ that supports an effective comparison of multiple categorical visualizations.

In line with the design requirements for natural image comparison and categorical data visualization [15, 22, 32], our problem is formulated based on the following three design requirements:

(i) **DR1:** highlighting the most concerned classes between visualizations as much as possible for an efficient comparison;
(ii) **DR2:** maximizing the visual discrimination between classes in the individual visualizations for an efficient exploration of multi-class data; and
(iii) **DR3:** providing flexible interactions for the exploration of relationships among the compared datasets.

Although visual comparison is an essential part of interactive data analysis, most of the existing colorization techniques [17, 32] attempt to meet DR2. The key challenge in meeting DR1 is that we need a proper model to characterize

the most salient features in multiple visualizations. To address this issue, we propose our co-saliency model that calculates the saliency of each data item in the context of other similar visualizations. Integrating this model into the objective of a state-of-the-art color mapping generation framework [32], we can generate proper color mappings that highlight salient differences between juxtaposed categorical visualizations while fostering a better visual discrimination of classes.

3.1 Co-saliency for Multi-class Scatterplots

Following the definition of image co-saliency [22], we model class co-saliency with two factors: class importance between visualizations and class contrast within visualizations. The class importance describes how much each class should stand out from the visualization. Class contrast describes how much each class stands out from neighboring classes and the background, which is similar to perceptual class separability [2, 46]. Hence, we define two types of class contrasts: a local contrast with neighboring classes and a contrast with the background.

Since point-based representations are very general in 2D visualization, we use two ($N = 2$) horizontally juxtaposed scatterplots to illustrate our method. Analogous to bottom-up image co-saliency models [13, 22], the co-saliency of the ith class is defined as the product between the class importance and class contrast scores to emphasize the target class and the co-saliency for M classes:

$$E_{CoS} = \sum_{i}^{M} \left(\sum_{j}^{N} \frac{1}{n_i^j} \left(\lambda \alpha_i^j \exp(\theta_i) + (1-\lambda) \beta_i^j f(\theta_i) \right) \right) \tag{1}$$

where θ_i is the importance of the ith class, n_i^j is the number of points of the ith class in the jth scatterplot, α_i^j is the local contrast with the neighboring classes of the ith class in the jth scatterplot, β_i^j is the contrast of the class to the background, and λ is the weight between them. The weight $1/n_i^j$ is used to alleviate class imbalances so that classes with small numbers of points and large changes can be highlighted.

To better support DR1, we apply an exponential function to enlarge the weight of structural class changes, while using a piecewise function weighting the background contrast:

$$f(\theta_i) = \begin{cases} \exp(\theta_i) & \text{if } \theta_i > \kappa \\ -\exp(\theta_i) & \text{else} \end{cases} \tag{2}$$

κ is a user-specified threshold with a default of zero. The reason for the two different weighting schemes is that classes with less or no changes might be treated as the background by viewers [49]. To suppress the saliency of such classes, we introduce negative importance for them.

Local Contrast. Given the jth scatterplot, we define the local class contrast based on the α-shape based point distinctness [32]. For each data point $\mathbf{x}_t^j$, we define its point distinctness as:

$$\gamma(\mathbf{x}_t^j) = \frac{1}{|\Omega_t^j|} \sum_{\mathbf{x}_p^j \in \Omega_t^j} \frac{\Delta\epsilon(\tau(l(\mathbf{x}_t^j)), \tau(l(\mathbf{x}_p^j)))}{d(\mathbf{x}_t^j, \mathbf{x}_p^j)},$$

where Ω_t^j is set of k nearest neighbors of $\mathbf{x}_t^j$, $\tau(l(\mathbf{x}_p^j))$ is the color of $\mathbf{x}_p^j$, d is the Euclidean distance and $\Delta\epsilon$ is the CIELAB color distance [41]. For the ith class, its local contrast is the sum of all points with the same class label in the scatterplot:

$$\phi_i^j = \frac{1}{n_i^j} \sum_t^{n_j} \gamma(\mathbf{x}_t^j)\delta(l(\mathbf{x}_t^j), i) \tag{3}$$

where $\delta(l(\mathbf{x}_t^j), i)$ is one if the class label $l(\mathbf{x}_t^j)$ is i and else zero.

If a class overlaps with different classes, the local contrast value will be high and the value will be small for a well separated class. Hence, the black class in the two scatterplots shown in Fig. 2(a) has a low contrast value (see Fig. 2(b)) and the cyan class has a large value.

Background Contrast. The contrast to the background is based on the so-called point non-separability $\rho(\mathbf{x}_t^j)$ (rf. [46]), which is defined as the difference between two separation degrees:

$$\rho(\mathbf{x}_t^j) = b(\mathbf{x}_t^j) - a(\mathbf{x}_t^j)\ . \tag{4}$$

where $b(\mathbf{x}_t^j)$ is the between-class separation degree and $a(\mathbf{x}_t^j)$ is the within-class separation degree. The measures are defined as weighted sums of color differences of $\mathbf{x}_t^j$ with its neighborhood from the same and different classes:

$$a(\mathbf{x}_t^j) = \frac{1}{|\Omega_t^j|} \sum_{\mathbf{x}_p^j \in \Omega_t^j} \frac{\delta(l(\mathbf{x}_t^j), l(\mathbf{x}_p^j))\Delta\epsilon(\tau(l(\mathbf{x}_t^j)), \mathbf{c}_b)}{d(\mathbf{x}_t^j, \mathbf{x}_p^j)}, \quad b(\mathbf{x}_t^j) = \frac{1}{|\Omega_t^j|} \sum_{\mathbf{x}_p^j \in \Omega_t^j} \frac{(1 - \delta(l(\mathbf{x}_t^j), l(\mathbf{x}_p^j)))\Delta\epsilon(\tau(l(\mathbf{x}_t^j)), \mathbf{c}_b)}{d(\mathbf{x}_t^j, \mathbf{x}_p^j)}$$

When most neighbor points of $\mathbf{x}_t^j$ have the same label as $\mathbf{x}_t^j$, $\rho(\mathbf{x}_t^j)$ is negative. However, such a negative $\rho(\mathbf{x}_t^j)$ makes the optimization in Eq. 1 meaningless, and the corresponding classes might be highlighted no matter how large the change of this class is. To address this issue, we use an exponential function to let $\rho(\mathbf{x}_t^j)$ always be positive while maintaining the monotonicity of the function. Accordingly, we define the contrast to the background of the ith class as:

$$\beta_i^j = \frac{1}{n_i^j} \sum_t^{n_j} \exp(\rho(\mathbf{x}_t^j))\delta(l(\mathbf{x}_t^j), i). \tag{5}$$

As illustrated in Fig. 2(c), well-separated classes with large color differences from the background have large background contrast (blue and black classes), whereas the pink and cyan classes have relatively large background contrast values with a medium class separation.

Class Importance. Class importance reflects whether a class should be highlighted or not. It can be specified by user or by some measures. In our paper, as a default we use the class change degree to represent the importance of each class. To quantify how users perceive structural changes of classes, we measure the difference between the class distributions in two scatterplots using the Earth Mover's Distance (EMD) [39], a perceptual metric. Suppose the ith class with two representations by two sets of points $\mathbf{X}_i^1 = \{\mathbf{x}_{i,1}^1, \cdots, \mathbf{x}_{i,n_i^1}^1\}$ and $\mathbf{X}_i^2 = \{\mathbf{x}_{i,1}^2, \cdots, \mathbf{x}_{i,n_i^2}^2\}$. Taking the Euclidian distance between points as the cost, we need to minimize the total matching cost

$$H(\mathbf{X}_i^1, \mathbf{X}_i^2) = \min_\chi \sum_t d(\mathbf{x}_{i,t}^1, \mathbf{x}_{i,\chi(t)}^2),$$

which constrains one-to-one mappings χ between points This is the classic bipartite matching problem, which can be solved by the Hungarian method [27]. When the number of points of two sets is not equal, we further take the difference between the number of points into account. In doing so, the class change degree contains positional changes

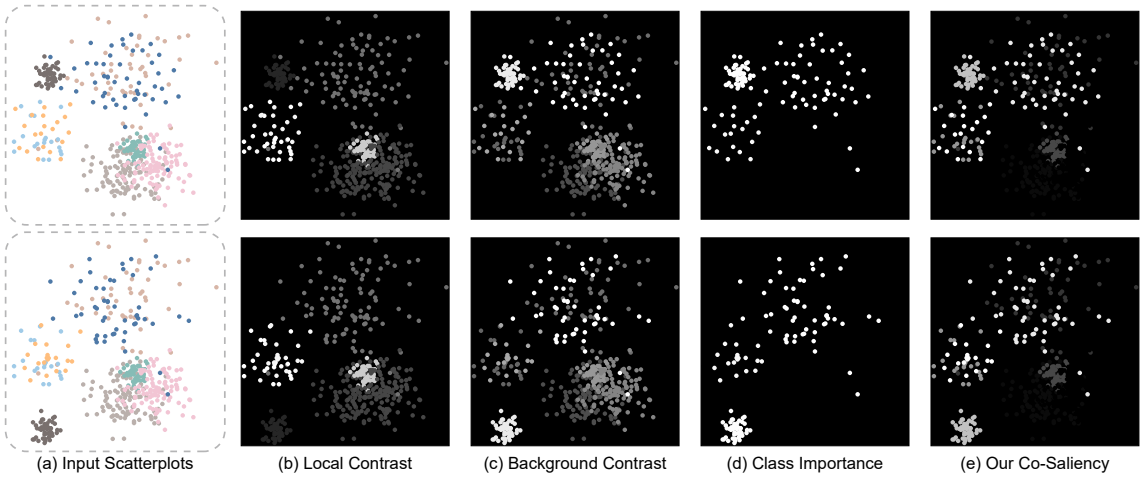

Fig. 2. Main components for computing co-saliency maps: For the two input scatterplots (a), our class-based co-saliency (e) is generated by fusing local contrast (b), background contrast (c), and class change degree (d). Brightness of points denotes value.

and changes of element numbers:

$$\theta_i = \frac{H(\mathbf{X}_i^1, \mathbf{X}_i^2)}{\min\{n_i^1, n_i^2\}} + \nu \frac{||n_i^1 - n_i^2||}{\max\{n_i^1, n_i^2\}} \tag{6}$$

both terms range within [0,1] and ν is 1.0 as the default value.

Fig. 2 shows an example of two 8-class scatterplots with three changing classes (orange, blue and black). Combining the class change degree with the two above-given contrast measures allows us to highlight salient differences and maintain the visual discrimination of the classes (see Fig. 2 (e)).

3.2 Co-Saliency based Palette Generation

On the basis of our co-saliency model we meet DR1 and DR2 by a co-saliency based generation of color palettes. Taking the model in Eq. 1 as the objective within a state-of-the-art optimal discrimination assignment method [46], an optimal color mapping can be obtained from a given good palette. However, there are two major limitations by not taking palette generation itself into account: i) the model requires users to try many palettes for selecting a good one; and ii) the design of most existing palettes is not oriented towards visual comparison so that even the best color assignment cannot provide prominent cues for this task. Fig. 3 shows examples with the Tableau-10 palette and ColorBrewer palette [18]. Both results highlight several classes with minor changes (e.g., the bottom left purple class in Fig. 3 (b)), and make it hard to identify the red class with the largest change even though it is very distinctive. Thus, we prompt users to use our co-saliency based palette generation method.

A recently proposed data-aware palette generation method by Lu et al. [32] automatically generates discriminable and preferable palettes by maximizing the combination of three palette quality measures: point distinctness, name difference, and color discrimination. By replacing the first measure with our co-saliency model, palette generation can be formulated as an optimization problem:

$$\arg\max_{\tau} E(\tau) = \omega_0 E_{CoS} + \omega_1 E_{ND} + \omega_2 E_{CD}. \tag{7}$$

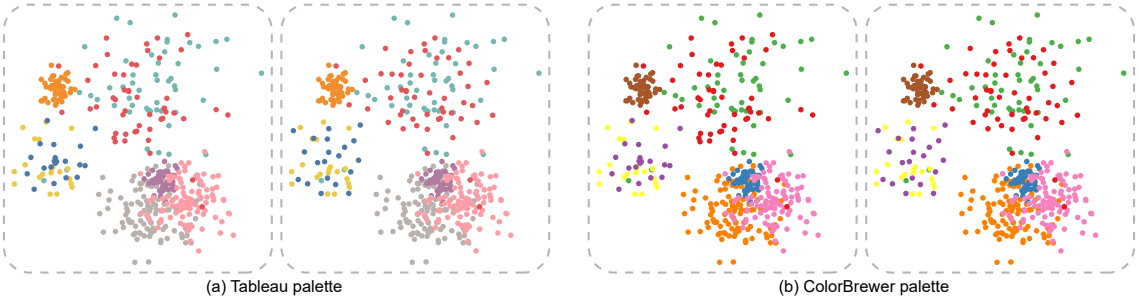

Fig. 3. Results generated by a co-saliency based color assignment with the existing Tableau-10 palette (a) and the ColorBrewer palette (b). Many existing palettes consist of bright colors only, where classes with smaller changes cannot be de-emphasized appropriately.

consisting of a co-saliency term E_{CoS} (see Eq. 1), a name difference term E_{ND} and a color discrimination term E_{CD}, balanced by ω_0, ω_1 and ω_2. For more details about E_{ND} and E_{CD}, we refer readers to [32]. By using their optimization method, we are able to generate desired color palettes up to 40 colors in real time. For example, Fig. 4 (b) shows an example which uses the same dataset as in Fig. 3, but improves the distinctness of the two changed classes while maintaining class separability.

3.3 Parameter Effect

Besides using different weights for the terms in palette generation [32], our co-saliency model involves three parameters: the weight λ between the two contrasts, the threshold for the class importance κ, and ν, which is related to the definition of the class change degree that is used as our default class importance. Since ν is fixed in our experiments and the class importance can be specified by the user, we mainly discuss here the effects of λ and κ.

Balancing Weight λ. Although this parameter modulates the influence between class contrast with its neighbors and background, it offers a compromise between DR1 and DR2. As shown in Fig. 4 (a), considering only the contrast to the background would result in a good "pop-out" effect, but other classes might be hard to discriminate. While considering only the contrast with nearest neighbors, such as done in Fig. 4 (d), all the classes are easy to distinguish but the changed classes are hard to find out. This is reasonable, because pre-attentive vision lets a bright and saturated color region within regions of de-saturated colors "pop out" to the viewer [19]. In our experiments, we found that setting $\lambda = 0.4$ as a default value allows to simultaneously emphasize changes and preserve the discriminability between classes, see the example in Fig. 4 (b).

Importance Threshold κ. The importance threshold κ selects classes with large importance to be highlighted. With a default value of zero, all classes with an importance value larger than zero are ensured to be highlighted. Likewise, a large κ will de-emphasize classes with small importance. We further allow users to specify κ by interaction through a widget in our interactive application.

Note that our optimization inherently works for more than two categorical visualizations and the data with many classes. Since our contrast measures are built on the data-space nearest neighbour graphs, its produced palettes also work well for scatterplots with significant overlap between classes (see the supplementary material).

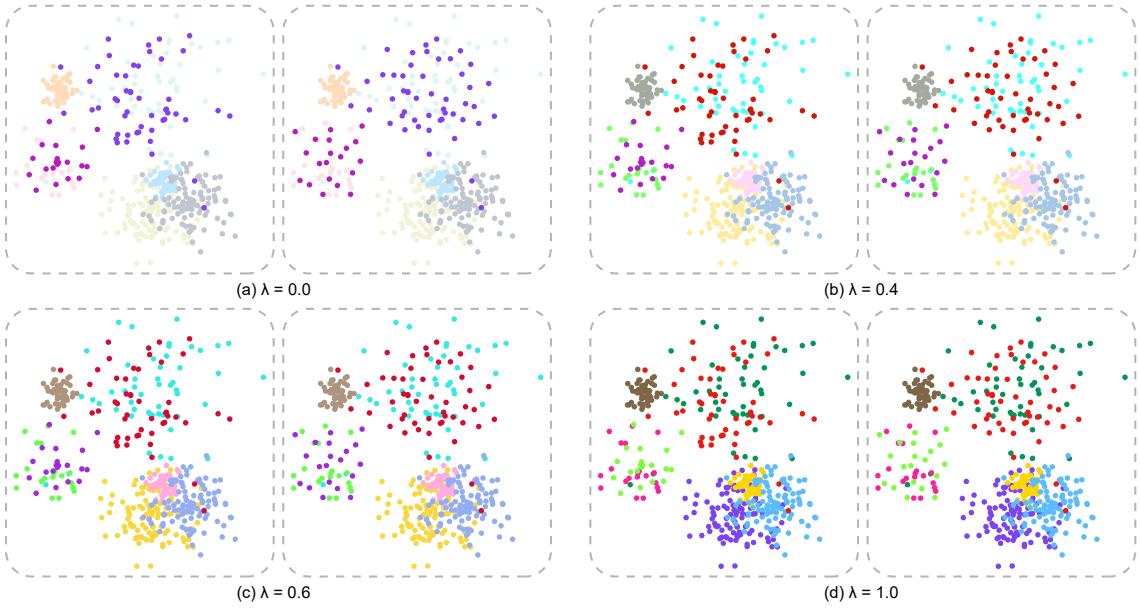

Fig. 4. Effect of contrast weight λ: (a) result considering only contrast to the background; (b) result with $\lambda = 0.4$; (c) result with $\lambda = 0.6$; (d) result generated by only considering contrast with nearest classes.

3.4 Bar and Line Charts

Like Palettailor [32], our color mapping generation method works also for other categorical visualization types such as bar or line charts. This is achieved by treating each bar or line segment in both views as a point and then using the same method to compute their class contrast. Taking line charts as an example, we order the line segments along the time axis and build a one-to-one mapping for line segments to compute θ_i. Doing so, lines with large changes will be highlighted while maintaining the discriminability between multiple lines in each chart. The same is done for bar charts, see Fig. 1. More results can be found in our supplementary material.

4 EVALUATION

We evaluated the effectiveness of our method on supporting juxtaposed visual comparisons by comparing it against existing methods across different visualization types. Considering that oftentimes there are multiple tasks involved in juxtaposed visual comparisons, we evaluated our method from two different perspectives, i.e., examining how well it can support people (1) to *observe the changes* for juxtaposed categorical visualizations (i.e., identifying delta), and (2) to *visually distinguish different classes* in each individual visualization in the pair (i.e., identifying the number of classes being compared), which is considered fundamental to juxtaposed comparison [15]. Besides, we also examined whether the effect of our method would be consistent across different visualization types by applying it to two charts: bar charts and scatterplots, which are typical and widely-used examples of statistical graphics.

To achieve these evaluation goals, we conducted three online controlled experiments through Amazon Mechanical Turk (AMT) involving 30 participants for one bar chart task (replicating Ondov et al.'s [37] study) and 160 participants for two scatterplot tasks respectively, and in each we compared C^3-*palette* with existing benchmark methods.

Colorization Methods (Conditions). In each of our experiments, we compared six different colorization methods, specifically including four benchmark methods (*Random Assignment*, *Optimized Assignment* [46], *Alpha Blending* and *Palettailor* [32]) that colorize based on one of the two input datasets, and our approach (C^3*-palette Assignment*, C^3*-palette Generation*) that colorize based on both input datasets. The first three methods are based on Tableau 20 [43], which is a designer-crafted palette containing colors with a large range of brightness and saturation, and the other three methods are generating and assigning colors to classes automatically. These methods are ordered by the level of optimization applied.

(1) *Random Assignment:* This is the common default result in data visualization by randomly selecting and assigning colors from a given palette to one of the two datasets.
(2) *Optimized Assignment:* Using the optimal discrimination assignment approach [46] to mimic the best discriminable result from user manual selection with a proper palette. This is achieved by selecting and assigning colors from the given palette to one of the two datasets. Since the original method was designed for scatterplots, we extend it for bar charts by treating each bar as a point in scatterplots as Lu et al. [32].
(3) *Alpha Blending:* This is the straightforward way to highlight classes: setting the unchanged classes to be semi-transparent (i.e., $alpha = 0.5$) while keeping the changed classes to be opaque. We choose the 0.5 threshold through empirical tests to balance between saliency and discriminability among classes. We applied this method to *Optimized Assignment* result to improve the discriminability of different classes.
(4) *Palettailor:* Since this is the state-of-the-art palette generation method for single view colorization, we used *Palettailor* [32] to generate and assign colors based on one of the two datasets with the default settings.
(5) C^3*-palette Assignment:* Selecting and assigning colors from the Tableau-20 palette based on *both* datasets as input, using the color assignment optimization solution (Eq. 1) proposed in this paper. This condition aims to reflect the effectiveness of our assignment approach given a palette input with relatively diverse options, such as Tableau-20.
(6) C^3*-palette Generation:* Generating colors using the color generation method (Eq. 7) with the default settings ($\omega_0 = 1.0$, $\omega_1 = 1.0$, $\omega_2 = 1.0$ and $\kappa = 0$). This condition aims to reflect the effectiveness of the fully automated option of our method.

4.1 Experiment 1: Bar Chart Experiment

We evaluated the effectiveness of our method on supporting juxtaposed visual comparisons using bar charts by following Ondov et al.'s [37] comparison evaluation, in which the participants performed the *max delta task*. The *max delta task* can reflect how well our method can support people to *observe the changes* for juxtaposed comparisons. We did not evaluate the other perspective: discriminating classes in individual charts, because the task is considered trivial in the context of bar chart, and so that it could lead to ceiling effect.

For the *max delta task*, we hypothesized that our method would generally be more effective than the benchmark methods on the task performance, and specifically we had the following hypothesis:

H1. Our palette generation method (C^3*-palette Generation*) outperforms the benchmark conditions (*Random Assignment*, *Optimized Assignment*, *Alpha Blending* and *Palettailor*) on the task performance.

H2. Our color assignment method (C^3*-palette Assignment*) using a color palette with a large range of brightness and saturation (*Tableau-20*) outperforms the benchmark conditions (*Random Assignment*, *Optimized Assignment*, *Alpha Blending* and *Palettailor*) on the task performance.

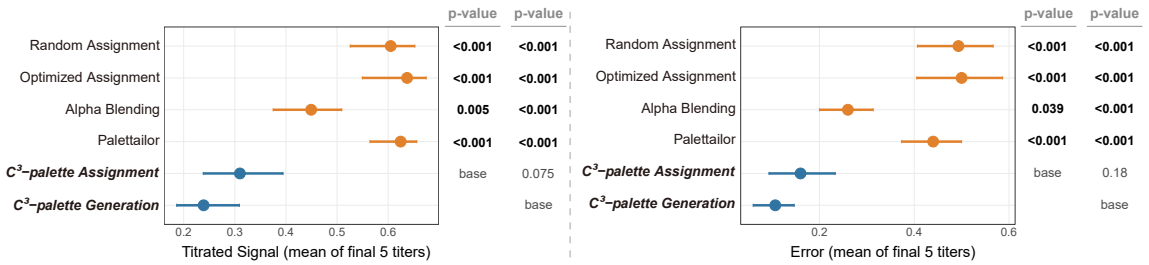

Fig. 5. Confidence interval plots and p-value from the Mann-Whitney test for the *max delta task* of the bar chart experiment. Error bars represent 95% confidence intervals. Each p-value shows the statistical test result of one base with the other conditions. Smaller value means a better performance.

Task & Measures. Following the methodology by Ondov et al. [37], we asked the participants perform the *max delta task* . Specifically, the participants were asked to find the bar that had the largest difference in the two bar charts. For each trial, we measured the *titer value* (ranging from 0 to 1) and *error* (using 0 and 1) to evaluate the performance of different conditions.

Dataset generation. All the stimulus datasets used in this experiment were generated by the titer staircase method [37] in real time, where each bar chart in the pair consisted of 7 data points (bars), and each pair of bar charts were generated differently depending on participant performance in the previous trial. This was done in order to minimize the learning effect. Specifically, we used *titer value* [37], the largest difference between the two bar charts, to quantify and control the difficulty of a trial. A larger titer value indicates an easier trial. For each condition method, the first trial contains the bar chart pair with a titer of 0.5. An erroneous response made the titer of the next trial increase by 0.3 (easier) while a correct answer led to a decrease by 0.1 (harder). We set 0.75 as the maximum titer value to prevent participants from going through too many easy trials.

Procedure. Prior to the main task, each participant viewed the instructions and went through four training trials. The first training trial does not have a time limit and the participant has to answer correctly in order to pass, while the other three trials are identical to the real test (having a time limit of 1.5 seconds for impression) with the largest titer value (0.75), i.e., easiest trials. Then each participant had 120 trials in the main study, i.e., twenty trials for each condition methods. The order of the conditions was randomized. All the bar charts are generated dynamically and the method conditions are run in real-time.

Participants. We recruited 32 participants through the Amazon Mechanical Turk, and each participant went through all the six condition methods with random order. According to the completion time in the pilot study, we paid each participant $1.5 for the task based on the US minimum hourly wage. No participant claimed color vision deficiency on their informed consent.

Analysis. We ran an accuracy-based outlier exclusion to filter participants whose overall proportion of correct trials was lower than two standard deviations from the mean of that from the other participants. This procedure resulted in 2 participants being excluded from the analysis. Following Ondov et al. [37], we performed within-subjects comparisons of the means of the *titer values* of the final 5 trials per condition. Aside from this primary indicator of the task performance, we also compared the *error* measure in a similar manner as a secondary indicator. Specifically, for each measure, we calculated the 95% confidence intervals using the bootstrap method. In addition, we used the more conservative, non-parametric Wilcoxon signed-rank test without normality assumption to compare different condition groups.

Results. As shown in Fig. 5, the condition method C^3*-palette Generation* led to a significantly better task performance (indicated by lower *Titrated Signal* and *Error*) compared to the benchmark conditions ($p < 0.001$). (**H1** confirmed) Similarly, the condition method C^3*-palette Assignment* also led to a significantly better task performance (indicated by lower *Titrated Signal*) compared to the benchmark conditions: *Random Assignment* ($p < 0.001$), *Optimized Assignment* ($p < 0.001$), *Alpha Blending* ($p = 0.005$), *Palettailor* ($p < 0.001$). (**H2** confirmed) Please see the supplementary materials for the detailed statistics.

4.2 Experiment 2: Scatterplot Experiment

In addition to bar charts, we also evaluated our approach in a more complex visualization scenario – scatterplots, where different visual tasks are involved that need to be supported. Specifically, we evaluated whether our method can support people to *observe the changes* for juxtaposed categorical scatterplots, as well as to *visually distinguish different classes* in each individual scatterplot. Therefore we adopted two tasks from literature: the *identifying delta task* [14] and *counting class task* [32] to reflect the performance of the two perspectives. In addition, in this experiment we also examined whether the effect of our method would vary based on *change magnitude*, the magnitude of the change between two scatterplots.

For the two tasks, we applied the similar experiment design and used the same set of pre-defined datasets, while recruited different groups of participants. Thus we describe the dataset generation and experiment organization altogether and report the results separately.

Dataset Generation. The paired scatterplot datasets used in our studies were generated as follows. First, we generated a set of multi-class scatterplots, each containing 8 classes. Each class was generated using Gaussian random sampling and placed randomly in a 600×600 area. Similar to [32], these classes belong to one of the four settings of varying size and density: small & dense ($n = 50, \sigma = 20$), small & sparse ($n = 20, \sigma = 50$), large & dense ($n = 100, \sigma = 50$), and large & sparse ($n = 50, \sigma = 100$).

For each scatterplot generated above, we produced its paired scatterplot by randomly changing one or more of its classes. To systematically changing the classes, we defined *change magnitude* which quantifies the magnitude of change between two scatterplots using Eq. 6. We generated scatterplot pairs with three levels of change magnitude: small, medium and large, by varying three underlying factors: *change type*, *change ratio* and *number of changed classes*. *Change type* defines *how* the points in a class change. There are two types of changes: *point number change*, where the number of points in a class changes, and *point position change*, where the positions of the points in a class change. *Change ratio* defines *how large* the change of a type is, ranging from 0 to 1. Number of changed classes defines *how many* classes are changed, ranging from 1 to 3.

For each change type, we report the process in which the changes were generated. First, for the *point number* change type, we calculated the new point number by adding or subtracting points from the original class by (1± *change ratio*), where the additional points were generated with the same distribution as that for the original class, and the subtraction of points was achieved by randomly deleting data points from the original class. Second, for the *point position* change type, we randomly chose from two kinds of positional changes: moving the *center* of the entire class, and moving the individual point positions, which results in the change of the *shape* of the class. For the *center* change, we moved the center towards a random direction by a distance, which equals to a maximal distance (400 by default) weighted by *change ratio*. For the *shape* change, we moved the density parameter of its Gaussian distribution into the opposite direction of the given value. For example, a small & dense class ($n = 50, \sigma = 20$) would be changed into a small & sparse

($n = 50, \sigma = 50$) class. After calculating the one-to-one mapping between the new and original class using [27], we linearly interpolated the position for each point weighted by *change ratio*.

From the above process, we produced 300 candidate scatterplot pairs for each change type, and then calculated the *change magnitude* for each pair using Eq. 6, and split all pairs into three levels: *small*, *medium*, and *large*. Finally, without loss of fairness, we randomly selected 2 pairs from each change magnitude level for each change type and each number of changed classes. Thus in total we had 36 scatterplot pairs. The detailed dataset is showed in Fig. 6 (a).

Experiment Organization. Instead of using a *within-subject* design that each participant need to do all the $36 * 6 = 216$ condition-dataset combinations, we tested the effects of the 6 method conditions across 36 paired multi-class scatterplot datasets using a *between-subject* experiment design. To avoid ordering effects, where the participant would get familiar with a dataset after seeing it several times (six times in our case), each participant was assigned to a group and saw a specific subset of datasets under different conditions. We used a Latin Square grouping (see Fig. 6 (a)) to organize the trials for each participant.

For attention check, we added four additional validation stimuli for both tasks. Each stimulus has a pair of scatterplots with 6 well-separated classes, including only one changed class with a large change magnitude, and we assigned a de-saturated color to the changed class that made it less salient. Accordingly, there are 36 + 4 = 40 trials in total for each participant group. To alleviate learning effects, we randomly shuffled the display orders of all scatterplot pairs, and randomly placed the two scatterplots in each pair on the left or right side.

	C1	C2	C3	C4	C5	C6
Dataset 1: Small (Position, 1)	**G1**	G2	G3	G4	G5	G6
Dataset 2: Small (Position, 1)	G6	**G1**	G2	G3	G4	G5
Dataset 3: Small (Position, 2)	G5	G6	**G1**	G2	G3	G4
Dataset 4: Small (Position, 2)	G4	G5	G6	**G1**	G2	G3
Dataset 5: Small (Position, 3)	G3	G4	G5	G6	**G1**	G2
Dataset 6: Small (Position, 3)	G2	G3	G4	G5	G6	**G1**
Dataset 7: Medium (Position, 1)	**G1**	G2	G3	G4	G5	G6
Dataset 8: Medium (Position, 1)	G6	**G1**	G2	G3	G4	G5
...						
Dataset 35: Large (Number, 3)	G3	G4	G5	G6	**G1**	G2
Dataset 36: Large (Number, 3)	G2	G3	G4	G5	G6	**G1**

(a)

Task / **Group**	identifying delta task (108)	counting class task (52)
G1	18	9
G2	17	8
G3	19	8
G4	17	9
G5	19	9
G6	18	9

(b)

Fig. 6. Experiment organization for the scatterplot experiment. (a) Grouping of Datasets: 36 datasets × 6 conditions. C: condition; G: participant group; Small (Position, 1): point position change with small change magnitude for 1 changed class; (b) Participants details for each task of the scatterplot experiment.

4.2.1 *Identifying delta task.*

To evaluate how well our approach enables viewers observing changes between juxtaposed categorical scatterplots, we conduct an online *identifying delta* experiment through Amazon Mechanical Turk (AMT) with 108 participants. In this task, we asked participants to identify all the classes that have been changed between two scatterplots.

Hypotheses. We hypothesized that our approach would generally be more effective than the benchmark methods on the juxtaposed comparison tasks, and that this effect would vary based on *change magnitude*.

H1. Our palette generation method (C^3*-palette Generation*) outperforms the benchmark conditions (*Random Assignment*, *Optimized Assignment*, *Alpha Blending* and *Palettailor*) on the task performance.

H2. Our color assignment method (*C^3-palette Assignment*) using a color palette with a large range of brightness and saturation (*Tableau-20*) outperforms the benchmark conditions (*Random Assignment*, *Optimized Assignment*, *Alpha Blending* and *Palettailor*) on the task performance.

H3. There is a significant interaction effect between colorization methods and *change magnitude*, resulting that the performance of our methods (*C^3-palette Generation* and *C^3-palette Assignment*) and that of the benchmark methods (*Random Assignment*, *Optimized Assignment*, *Alpha Blending* and *Palettailor*) vary based on *change magnitude*.

Measures. For each participant, we measured the *time* taken for each trial, and counted the errors (0/1) indicating whether the actual changed classes are aligned with the participant's response. Note that if any of the changed classes was mistakenly identified, the trial would be considered as "wrong" (1).

Pilot Study. We conducted a pilot study involving 28 participants to check the experimental setup and determine the parameters, such as the time limit for a trial. Harnessing by the pilot study, we also obtained our expected effect sizes, which were in further fed into a power analysis. With an effect size Cohen's d of 0.4, alpha level of 0.05 and beta level of 0.8, the power analysis suggested a minimum number of 100 participants for the spot-the-difference task. See the supplementary material for more details.

Participants. We recruited 108 participants (see Fig. 6 (b)) for the experiment on Amazon Mechanical Turk. According to the completion time in the pilot study, we paid each participant $1.5 for the task based on the US minimum hourly wage. No participant claimed color vision deficiency on their informed consent.

Procedure. Each participant went through the following steps in our experiment: (i) viewing a user guide of the task and completing three training trials; (ii) completing each trial as accurately as possible; (iii) providing demographic information. At the beginning of each trial, the number of changed classes was provided. Each participant was asked to select all the changed classes by clicking one of the points belonging to these classes in either of the scatterplots. While the participant was instructed to do the task "*as accurately as possible*", we set a 60-second time limit for each trial for fear that user might spend too much time on the trial. If the participant could not find all the changed classes during the time limit, they were directed to the next trial. This was done since we observed from the pilot study that when participants spent too much time on a single trial, they may decide to quit by selecting a class randomly (which would lead to an incorrect answer) or to spend more time till they get the correct answer (which would lead to an increasing time spent on the trials). Such subject decisions would add noise to our measurements. Thus we added a 60-second time limit, which was indicated by our pilot study: over 92% of the trials were completed within that time.

Analysis. Following previous studies, we analyzed the results using 95% confidence intervals, and also conducted Mann-Whitney tests to compare the differences between conditions. The non-parametric test was used due to observations of non-normally distributed data from our pilot study. In addition, we computed the effect size using *Cohen's d*, i.e., the difference in means of the conditions divided by the pooled standard deviation. We calculated ANOVA-type statistic (ATS) without normality assumption (using R Package GFD [12] to examine the interaction effect between variables.

Results. Fig.7 shows the results of the online experiment. First, we found that our approach *C^3-palette Assignment* leads to a significantly lower error rate than all benchmark conditions: *Random Assignment* ($p < 0.001$), *Optimized Assignment* ($p < 0.001$), *Alpha Blending* ($p = 0.005$), *Palettailor* ($p = 0.002$). *C^3-palette Generation* also has a significantly lower error rate than all benchmark conditions: *Random Assignment* ($p < 0.001$), *Optimized Assignment* ($p < 0.001$), *Alpha Blending* ($p = 0.007$), *Palettailor* ($p = 0.003$). The results for the completion time also shows a similar trend of

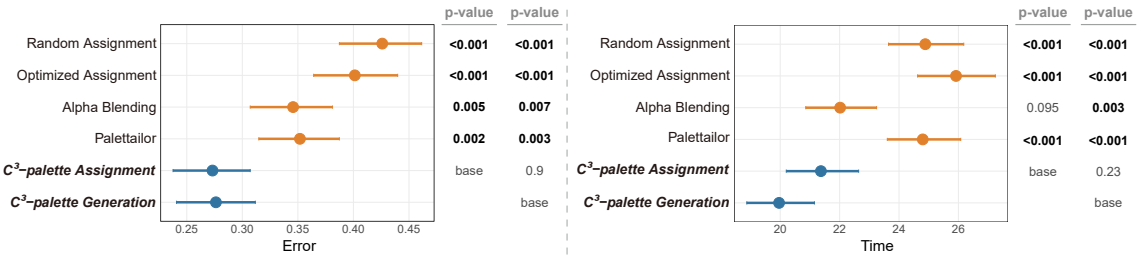

Fig. 7. Confidence interval plots and p-value from the Mann-Whitney test for the *identifying delta task* of the scatterplot experiment. Error bars represent 95% confidence intervals. Each p-value shows the statistical test result of one base with the other conditions. Smaller value means a better performance.

our methods taking significantly less time than the benchmarks, with an exception that C^3*-palette Assignment* has no significant difference to *Alpha Blending* condition. The result indicates that our palette generation method (C^3*-palette Generation*) has a better performance than benchmark conditions in the *identifying delta task* (**H1** confirmed). As for color palettes with a larger range of brightness and saturation, our approach (C^3*-palette Assignment*) is better than the benchmark conditions in terms of error rate and is better than most benchmark conditions on completion time (**H2** partially confirmed).

However, we did not find significant interaction effect between *colorization methods* and *change magnitude* ($F(5, 3876) = 1.036; p > 0.05$), meaning that the effect of our method is not necessarily influenced by the magnitude of change between the two scatterplots (**H3** not confirmed).

4.2.2 *Counting class task.*

To evaluate whether our approach can fundamentally support the visual separability of the classes in each scatterplot, we conducted an online *counting class* experiment through AMT with 52 participants. Following previous methodologies [32, 46], we asked participants to identify how many classes (colors) are there in the given two scatterplots and then choose an answer among several options below the two scatterplots. The experimental design was similar to the first study. We expected to see different patterns of the discriminability across different conditions. Specifically, our methods would lead to a shorter error and time than *Random Assignment* and *Alpha Blending* conditions.

Hypotheses. We hypothesized that our approach would generally be more effective than the benchmark methods on the discrimination tasks, and that this effect would vary based on *change magnitude*. We proposed different hypotheses with the previous task.

- **H1.** Our palette generation method (C^3*-palette Generation*) outperforms the benchmark conditions (*Random Assignment*, *Optimized Assignment*, *Alpha Blending*) and our assignment method (C^3*-palette Assignment*), while is not worse than the state-of-the-art single view discriminable colorization method (*Palettailor*) on the task performance.
- **H2.** Our color assignment method (C^3*-palette Assignment*) based on *Tableau-20* outperforms the conditions (*Random Assignment*, *Alpha Blending*), and is not worse than the state-of-the-art single view discriminable assignment method (*Optimized Assignment*) on the task performance, since all of these methods take into account the class separability.

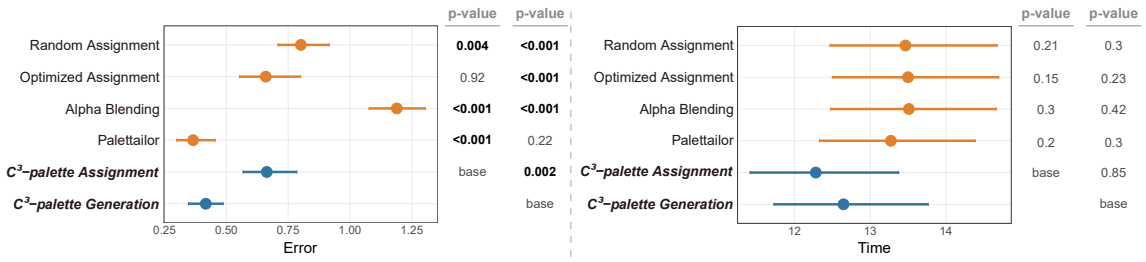

Fig. 8. Confidence interval plots and p-value from the Mann-Whitney test for the *counting class task* of the scatterplot experiment. Error bars represent 95% confidence intervals. Each p-value shows the statistical test result of one base with the other conditions. Smaller value means a better performance.

H3. There is a significant interaction effect between colorization methods and *change magnitude*, resulting that the performance of our methods (C^3*-palette Generation* and C^3*-palette Assignment*) and that of the benchmark methods (*Random Assignment*, *Optimized Assignment*, *Alpha Blending* and *Palettailor*) vary based on *change magnitude*.

Measures. We recorded the participant's answer and response time for each trial, and counted the *error* by calculating the differences between the participant's answer and the actual number of classes.

Pilot Study. This setting is similar to the previous task. We invited 29 participants to do the pilot study and the results were in further fed into a power analysis. With an effect size Cohen's d of 0.6, the power analysis suggested a minimum number of 50 participants for the discriminability task. See the supplementary material for more details.

Participants. We finally recruited 52 participants (see Fig. 6 (b)) for the experiment on Amazon Mechanical Turk. According to the completion time in the pilot study, we paid each participant $1.5 for the task based on the US minimum hourly wage. No participant claimed color vision deficiency on their informed consent.

Results. The procedure and analysis method are similar with the previous task. Fig.8 shows the results of this visual separability experiment. Through this study we first found that C^3*-palette Generation* is comparable to *Palettailor* while it leads to a significantly lower error rate($p < 0.001$) than all other benchmark conditions. Specifically, C^3*-palette Generation* has a significantly lower error rate($p = 0.002$) than C^3*-palette Assignment*. However, there's no significant difference on completion time (**H1** partially confirmed). Second, C^3*-palette Assignment* has a lower error rate than the benchmark conditions (*Random Assignment*, *Alpha Blending*) and is comparable to *Optimized Assignment*, but there's no significant difference on completion time (**H2** partially confirmed). Finally, we did not find a significant interaction effect between *colorization methods* and *change magnitude* ($F(5, 1859) = 1.175, p > 0.05$), meaning that the effect of different methods for visual discriminability seems not necessarily be influenced by the magnitude of change between the two scatterplots (**H3** not confirmed).

4.3 Discussion

In summary, we evaluated the effectiveness of our approach against the benchmark conditions through three online studies, including one bar chart experiment and two scatterplot experiments. We found that first, our method C^3*-palette Generation* outperforms the benchmark methods on juxtaposed comparison tasks across two visualizations: bar chart and scatterplot, and specifically for the scatterplot visualization, the effects of our methods are not necessarily influenced by the change magnitude of the two scatterplots. In addition, we observed that our C^3*-palette Assignment* method, when

coupled with a palette containing colors with a wide range of brightness and saturation, outperforms the benchmark methods. This indicates that our assignment method could perform well if given an ideal palette.

Second, through the scatterplot experiments, we found that our experimental methods (C^3*-palette Generation* and C^3*-palette Assignment*) generally support the fundamental visual separability of the classes. It is worth noting that C^3*-palette Generation* is not worse than *Palettailor* , while the C^3*-palette Assignment* is not worse than the *Optimized Assignment* . This indicates that our approach maintains the class discriminability of the scatterplot while enhances the class saliency to help user observe changes between different scatterplots.

Third, for both visualizations, one of the benchmark conditions *Alpha Blending* yielded a better performance in the *identifying delta* tasks compared with the other benchmark methods (yet still worse than our methods), which indicates that manipulating the opacity may help people identify changes to some extent. However, when it came to the separability task, *Alpha Blending* led to the worst performance of all the conditions, which indicates a tradeoff of class discriminability.

There are some **limitations** within our evaluation. First, we tested the assignment of our approach using a carefully-designed and diverse palette as input (i.e., the C^3*-palette Assignment* condition used Tableau-20), while it remains to be unclear whether other palettes (e.g., Tableau-10 or ColorBrewer) would lead to similar or different results. Second, our experiment focused on identifying the differences between two visualizations, which is a simplified situation, since in real-world cases often more than two visualizations are compared. Third, our evaluation focused on two visualization types: bar chart and scatterplot, while the effectiveness of our method on the juxtaposed comparison tasks of other visualizations (e.g., line chart, node-link tree) remains to be explored. Finally, we cannot further analyze the effect of the *change type*, given the current study design, though we did observe that our methods are more effective for certain types of change. That brings us to a series of more fundamental questions: how can we properly define the types of changes? What is the just noticeable change magnitude for each change type? Further research is needed to answer these questions so that our approach can be thoroughly evaluated.

5 INTERACTIVE SYSTEM

To help users interactively design colors for comparing multi-class scatterplots, we developed a web-based multi-view visualization tool[2] (see the screenshot in Fig. 9 (a)). It consists of four coordinated views: (i) a control panel, (ii) an importance adjustment panel for selecting κ and the importance of each class, (iii) the juxtaposed visualizations, and (iv) a history view.

After uploading multiple labeled datasets, the system automatically finds an optimal color mapping scheme to colorize the input data, while each class is encoded as a dot on the x-axis of the importance adjustment view indicating the change degree. If the user likes the color mapping scheme, s/he can save it to the history view. By default, our system finds a color mapping scheme that highlights classes with large changes and renders them in ascending order of the corresponding change degrees. To facilitate a coherent exploration, we provide a color name constraint for palette generation, so that the consistency of color names will be preserved in the produced palettes.

Color Name Constraints. Adjusting class importance and κ allows to highlight classes of interest with newly generated color palettes. However, this might not be intuitive for users, since the colors might be completely changed in the new palette. To address this issue, one straightforward way is to assign large opacities to classes of interest and small values

[2] https://c3-palette.github.io/

Fig. 9. Our interactive colorization system and a case study. (a) Screenshot of the system consisting of four panels: (i) control panel; (ii) importance adjustment panel; (iii) visualization panel; and (iv) history panel. (b) Using the system to explore the changes of gases in an air quality data set [10]: (top) An automatically generated palette creates salient colors for lines; (middle, bottom) the palettes highlighting two lines with small changes generated by our methods without and with colour name constraint.

to de-emphasized classes. However, this method might not be able to let such classes pop out, since their assigned colors often have a low contrast to the background (e.g., the yellow class in the top of Fig. 9 (b)).

To maintain consistent color schemes and highlight classes of interest, we introduce a color name constraint [20] for palette generation. Specifically, the name difference between the new color and the one in the previous palette should be smaller than a threshold during the search for new palettes. Note that using the hue preserving constraint [8] during the palette generation cannot produce consistent color schemes, see the results in the supplemental material.

5.1 Case Study

To shed further light onto the ecological validity of our approach, we conducted a case study on a real-world categorical dataset visualized with three line charts. Here, we analyze an air quality data set [10] that contains hourly responses of a gas multi-sensor device deployed in an Italian city from March 12 to March 18, 2014. The top in Fig. 9 (b) shows the juxtaposed line graphs encoded by our generated color palette, where each gas type is represented by a line with a unique color. We can see that all gases are encoded with highly salient colors, making it hard to explore changes of specific gases. This is reasonable because the default κ is zero, but all gases have large changes. Thanks to our interaction mechanism, users can directly select classes of interest to be highlighted by assigning them a large importance, while the θ values of the other classes are set to a smaller value than κ. Using the color name constrained palette generation method, the produced palette lets the selected lines pop out from the others (see the bottom in Fig. 9 (b)). Hence, users can easily explore the changes of the selected two gases (CO and NMHC) in the three juxtaposed views.

6 CONCLUSION AND FUTURE WORK

We presented C^3-palette, a data-aware approach for producing color palettes for comparing horizontally juxtaposed categorical visualizations that allows a better identification of the biggest changes between two data series, while maintaining the visual discrimination of classes. This goal is achieved by a novel co-saliency model, which characterizes the most co-salient features between juxtaposed labeled data visualizations while maintaining class discrimination in

the individual visualizations. We evaluated C^3-palette through a crowd-sourcing study, which empirically demonstrates that our produced palettes allow for an efficient visual comparison and good class discrimination.

Our work concentrated on juxtaposed comparisons to detect changes between multiple datasets, whereas its optimal color palette might not be appropriate for understanding other analytical comparison tasks (e.g., correlation tasks, rf. [37]). Future work needs to investigate the effectiveness and extensions of our approach for such comparison tasks. Furthermore, mark shape [30] and mark size [42] might have an effect on the perceptual precision of visual comparisons and we will explore the possibility to model the influence of these factors.

Second, our approach produces colors with salient hues to highlight classes with large changes, but those colors do not visually indicate the ranking of class changes. It would be helpful to associate a color ordering constraint [5] with the degree of changes, so that the ranking of class changes can be shown clearly. On the other hand, our method can be extended to generate palettes for people with color vision deficiency by incorporating a physiologically-based model [34] into our optimization framework.

Third, while our second user study only examined the interaction effect between change magnitude and different colorization methods, we plan to investigate how this effect is influenced by different types of changes in scatterplots, such as point number, center position and shape. The order of rendering is critical for the comparison task and in this paper we treat it simply by rendering less important classes first. But when there are multiple important large classes at the same positions, less important classes might be overlapped and hard to distinguish. Thus a professional render order algorithm would be necessary for multi-class scatterplot rendering.

Last, our study only evaluated the effectiveness of our palettes with horizontally juxtaposed visualizations, while there are different layout methods such as vertical arrangement, mirrored arrangement, overlaid, and animation. Previous studies [37] show that animation performs well in identifying the largest difference and we will conduct studies to learn how well our palette works in this setting. On the other hand, there are a few different visual comparison methods [16] such as plotting differences and faceting groups [48]. It would be helpful to fully investigate the strengths and limitations of each of these methods for visual comparisons.

REFERENCES

[1] D. Albers, C. Dewey, and M. Gleicher. 2011. Sequence Surveyor: leveraging Overview for Scalable Genomic Alignment Visualization. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2392–2401. https://doi.org/10.1109/TVCG.2011.232

[2] M. Aupetit and M. Sedlmair. 2016. SepMe: 2002 New visual separation measures. In *2016 IEEE Pacific Visualization Symposium*. 1–8. https://doi.org/10.1109/PACIFICVIS.2016.7465244

[3] Patrick Baudisch and Carl Gutwin. 2004. Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 367–374.

[4] Ali Borji, Ming-Ming Cheng, Qibin Hou, Huaizu Jiang, and Jia Li. 2019. Salient object detection: a survey. *Computational Visual Media* 5, 2 (2019), 117–150. https://doi.org/10.1007/s41095-019-0149-9

[5] R. Bujack, T. L. Turton, F. Samsel, C. Ware, D. H. Rogers, and J. Ahrens. 2018. 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 (2018), 923–933. https://doi.org/10.1109/TVCG.2017.2743978

[6] H. Chen, W. Chen, H. Mei, Z. Liu, K. Zhou, W. Chen, W. Gu, and K. Ma. 2014. Visual Abstraction and Exploration of Multi-class Scatterplots. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 1683–1692. https://doi.org/10.1109/TVCG.2014.2346594

[7] J. Chuang, D. Weiskopf, and Torsten Möller. 2009. Energy Aware Color Sets. *Computer Graphics Forum* 28 (2009). https://doi.org/10.1111/j.1467-8659.2009.01359.x

[8] Johnson Chuang, Daniel Weiskopf, and Torsten Moller. 2009. Hue-preserving color blending. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1275–1282.

[9] Charles E. Connor, Howard E. Egeth, and Steven Yantis. 2004. Visual Attention: bottom-Up Versus Top-Down. *Current Biology* 14, 19 (2004), R850–R852. https://doi.org/10.1016/j.cub.2004.09.041

[10] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia. 2008. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical* 129, 2 (2008), 750–757. https://doi.org/10.1016/j.snb.2007.09.060

[11] James T Enns. 1990. Three-dimensional features that pop out in visual search. (1990).

[12] Sarah Friedrich, Frank Konietschke, and Markus Pauly. 2017. GFD: An R Package for the Analysis of General Factorial Designs. *Journal of Statistical Software, Code Snippets* 79, 1 (2017), 1–18. https://doi.org/10.18637/jss.v079.c01

[13] H. Fu, X. Cao, and Z. Tu. 2013. Cluster-Based Co-Saliency Detection. *IEEE Transactions on Image Processing* 22, 10 (2013), 3766–3778. https://doi.org/10.1109/TIP.2013.2260166

[14] Eiji Fukuba, Hajime Kitagaki, Akihiko Wada, Kouji Uchida, Shinji Hara, Takafumi Hayashi, Kazushige Oda, and Nobue Uchida. 2009. Brain Activation during the Spot the Differences Game. *Magnetic Resonance in Medical Sciences* 8, 1 (2009), 23–32. https://doi.org/10.2463/mrms.8.23

[15] M. Gleicher. 2018. Considerations for Visualizing Comparison. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 413–423. https://doi.org/10.1109/TVCG.2017.2744199

[16] Michael Gleicher, Danielle Albers, Rick Walker, Ilir Jusufi, Charles D. Hansen, and Jonathan C. Roberts. 2011. Visual Comparison for Information Visualization. *Information Visualization* 10, 4 (2011), 289–309. https://doi.org/10.1177/1473871611416549

[17] C. C. Gramazio, D. H. Laidlaw, and K. 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. https://doi.org/10.1109/TVCG.2016.2598918

[18] 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. https://doi.org/10.1179/000870403235002042

[19] Christopher G Healey, Kellogg S Booth, and James T Enns. 1995. Visualizing real-time multivariate data using preattentive processing. *ACM Transactions on Modeling and Computer Simulation* 5, 3 (1995), 190–221. https://doi.org/10.1145/217853.217855

[20] Jeffrey Heer and Maureen Stone. 2012. Color naming models for color selection, image editing and palette design. 1007–1016. https://doi.org/10.1145/2207676.2208547

[21] L. Itti, C. Koch, and E. Niebur. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 11 (1998), 1254–1259. https://doi.org/10.1109/34.730558

[22] David E. Jacobs, Dan B. Goldman, and Eli Shechtman. 2010. Cosaliency: where People Look When Comparing Images. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*. 219–228. https://doi.org/10.1145/1866029.1866066

[23] H. Jänicke and M. Chen. 2010. A Salience-based Quality Metric for Visualization. *Computer Graphics Forum* 29, 3 (2010), 1183–1192. https://doi.org/10.1111/j.1467-8659.2009.01667.x

[24] N. Jardine, B. D. Ondov, N. Elmqvist, and S. Franconeri. 2020. The Perceptual Proxies of Visual Comparison. *IEEE Transactions on Visualization and Computer Graphics* 26, 1 (2020), 1012–1021. https://doi.org/10.1109/TVCG.2019.2934786

[25] Hye-Rin Kim, Min-Joon Yoo, Henry Kang, and In-Kwon Lee. 2014. Perceptually-Based Color Assignment. *Computer Graphics Forum* 33, 7 (2014), 309–318. https://doi.org/10.1111/cgf.12499

[26] Y. Kim and A. Varshney. 2006. Saliency-guided Enhancement for Volume Visualization. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 925–932. https://doi.org/10.1109/TVCG.2006.174

[27] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly* 2, 1-2 (1955), 83–97. https://doi.org/10.1007/978-3-540-68279-0_2

[28] S. Lee, M. Sips, and H. Seidel. 2013. Perceptually Driven Visibility Optimization for Categorical Data Visualization. *IEEE Transactions on Visualization and Computer Graphics* 19, 10 (2013), 1746–1757. https://doi.org/10.1109/TVCG.2012.315

[29] Sharon Lin, Julie Fortuna, Chinmay Kulkarni, Maureen Stone, and Jeffrey Heer. 2013. Selecting Semantically-Resonant Colors for Data Visualization. *Computer Graphics Forum* 32, 3 (2013), 401–410. https://doi.org/10.1111/cgf.12127

[30] Tingting Liu, Xiaotong Li, Chen Bao, Michael Correll, Changehe Tu, Oliver Deussen, and Yunhai Wang. 2021. Data-Driven Mark Orientation for Trend Estimation in Scatterplots. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–16.

[31] María-Jesús Lobo, Emmanuel Pietriga, and Caroline Appert. 2015. An Evaluation of Interactive Map Comparison Techniques. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3573–3582. https://doi.org/10.1145/2702123.2702130

[32] K. Lu, M. Feng, X. Chen, M. Sedlmair, O. Deussen, D. Lischinski, Z. Cheng, and Y. Wang. 2021. Palettailor: discriminable colorization for categorical data. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 475–484. https://doi.org/10.1109/TVCG.2020.3030406

[33] S. LYi, J. Jo, and J. Seo. 2021. Comparative Layouts Revisited: design Space, Guidelines, and Future Directions. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 1525–1535. https://doi.org/10.1109/TVCG.2020.3030419

[34] G. M. Machado, M. M. Oliveira, and L. A. F. Fernandes. 2009. A Physiologically-based Model for Simulation of Color Vision Deficiency. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1291–1298. https://doi.org/10.1109/TVCG.2009.113

[35] L. E. Matzen, M. J. Haass, K. M. Divis, Z. Wang, and A. T. Wilson. 2018. Data Visualization Saliency Model: a Tool for Evaluating Abstract Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 563–573. https://doi.org/10.1109/TVCG.2017.2743939

[36] Tamara Munzner, François Guimbretière, Serdar Tasiran, Li Zhang, and Yunhong Zhou. 2003. TreeJuxtaposer: scalable Tree Comparison Using Focus+Context with Guaranteed Visibility. *ACM Transactions on Graphics* 22, 3 (2003), 453–462. https://doi.org/10.1145/882262.882291

[37] B. Ondov, N. Jardine, N. Elmqvist, and S. Franconeri. 2019. Face to face: evaluating visual comparison. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 861–871. https://doi.org/10.1109/TVCG.2018.2864884

[38] Zening Qu and Jessica Hullman. 2017. Keeping multiple views consistent: constraints, validations, and exceptions in visualization authoring. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2017), 468–477. https://doi.org/10.1109/TVCG.2017.2744198

[39] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40, 2 (2000), 99–121. https://doi.org/10.1023/A:1026543900054

[40] V. Setlur and M. C. Stone. 2016. A Linguistic Approach to Categorical Color Assignment for Data Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (2016), 698–707. https://doi.org/10.1109/TVCG.2015.2467471

[41] Gaurav Sharma, Wencheng Wu, and Edul N Dalal. 2005. The CIEDE2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations. *Color Research & Application* 30, 1 (2005), 21–30. https://doi.org/10.1002/col.20070

[42] Stephen Smart and Danielle Albers Szafir. 2019. Measuring the separability of shape, size, and color in scatterplots. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–14.

[43] Tableau Software. [n.d.]. The tableau visualization system. http://www.tableausoftware.com/.

[44] C. Tominski, C. Forsell, and J. Johansson. 2012. Interaction Support for Visual Comparison Inspired by Natural Behavior. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2719–2728. https://doi.org/10.1109/TVCG.2012.237

[45] C. Tominski, G. Fuchs, and H. Schumann. 2008. Task-driven color coding. In *Proceedings of 12th International Conference Information Visualisation*. 373–380. https://doi.org/10.1109/IV.2008.24

[46] Yunhai Wang, Xin Chen, Tong Ge, Chen Bao, Michael Sedlmair, Chi-Wing Fu, Oliver Deussen, and Baoquan Chen. 2019. Optimizing color assignment for perception of class separability in multiclass scatterplots. *IEEE Transactions on Visualization and Computer Graphics* 25, 1 (2019), 820–829. https://doi.org/10.1109/TVCG.2018.2864912

[47] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. 2000. Guidelines for Using Multiple Views in Information Visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. 110–119. https://doi.org/10.1145/345513.345271

[48] Hadley Wickham et al. 2009. Elegant graphics for data analysis. *Media* 35, 211 (2009), 10–1007.

[49] Dingwen Zhang, Huazhu Fu, Junwei Han, Ali Borji, and Xuelong Li. 2018. A review of co-saliency detection algorithms: fundamentals, applications, and challenges. *ACM Transactions on Intelligent Systems and Technology* 9, 4 (2018), 1–31. https://doi.org/10.1145/3158674

[50] L. Zhou and C. D. Hansen. 2016. A Survey of Colormaps in Visualization. *IEEE Transactions on Visualization and Computer Graphics* 22, 8 (2016), 2051–2069. https://doi.org/10.1109/TVCG.2015.2489649