Color-Name Aware Optimization to Enhance the Perception of Transparent Overlapped Charts

This **supplementary material** provides additional experimental results for our submitted paper. Experimental data and analysis code can be accessed at https://osf.io/xevk9/?view_only=1b79aeefec774209ad60f1a5b0cceda2.

Navigation:

I. Comparing different similarity measures

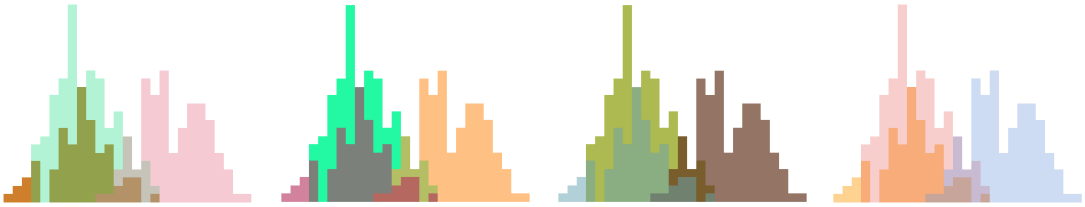
In this paper, we tried multiple measures for quantifying the association between different blended colors of one class, including color similarity, luminance similarity, hue similarity, and name similarity. Color Similarity(CS) is calculated by $1 - CD(c_i, c_j)$, where $CD(c_i, c_j)$ is a normalized CIEDE2000 color difference [8].This metric often resulted in colors with a small color distance but significant differences in their names. For instance, the pink and grey colors in Figure 1-a exhibit high color similarity ($CS($●,●$) = 0.61$) despite their distinct appearances. The Luminance Similarity(LS) is calculated by $1-0.01*LD(c_i, c_j)$, where LD is the absolute difference between two colors' luminance values in CIE Lab space. However, large luminance similarity does not necessarily indicate a strong association between colors, as shown in Figure 1-b, the colors of components exhibit highly similar luminance($LS($●,●$) = 0.70$, $LS($●,●$) = 0.92$, $LS($●,●$) = 0.78$), yet grouping them into a single class poses a challenge. Hue Similarity(HS) is calculated by $1 - HD(c_i, c_j)/180$, where HD is the absolute difference between two colors' hue values in CIE Lab space. However, a high hue similarity does not guarantee similar colors. For example, as shown in Figure 1-c, the green color has a similar hue to the brown color($HS($●,●$) = 0.83$), but people often find it challenging to associate the two colors together.

Based on these attempts, we finally decided to use the name similarity as the final similarity measure. As shown in Figure 1-d, the blue class is more easily distinguishable using name similarity compared to the other metrics, due to the stronger association between the names of the blended colors.

II. Detailed Discussion of the Results

We assessed the effectiveness of our approach against benchmark conditions (traditional alpha blending, local color blending, and hue-preserving blending) using colors from the Tableau-10 scheme or Colorgorical through three crowdsourced experiments. To color-code the stimuli, we randomly selected subsets of two, three, or four colors from the corresponding palette to assign to the classes in the stimuli. This random selection process ensured a diverse range of color combinations, providing a comprehensive evaluation of our approach's performance across different color schemes.

Each participant completed a total of 72 stimuli, consisting of 3 *classes* $\times$ 3 *smoothness levels* $\times$ 2 *benchmarks* $\times$ 2 *repetitions*, with 2 *repetitions* corresponding to Colorgorical and Tableau palettes.



(a) Color Similarity (b) Luminance Similarity (c) Hue Similarity (d) Name Similarity

Fig. 1: The optimization results by using different similarity measures for $S(c_i, c_j)$.

This design ensured that each palette was equally compared with our method, allowing for a balanced assessment of our approach's effectiveness in different color environments.

In this study, we use Tableau to represent designer-crafted palettes and Colorgorical to represent auto-generated palettes. Our comparative analysis demonstrates that the results generated by our algorithm are superior in several aspects to the existing designer-crafted and auto-generated palettes. Specifically, our approach consistently achieved lower error rates and faster response times, highlighting its effectiveness in providing clear and efficient visualizations.

A. Results for Distribution Estimation

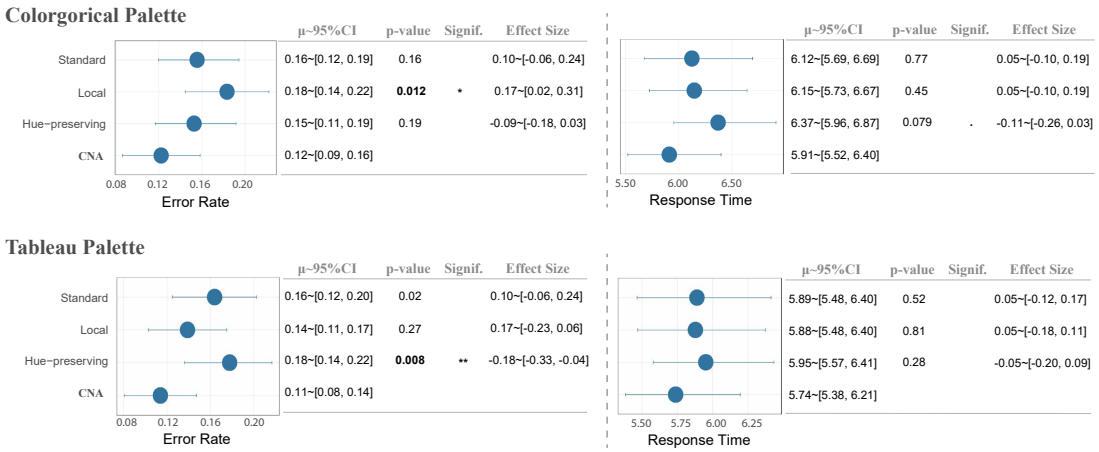


Fig. 2: Results for the *Distribution Estimation Task*, including effect sizes and significance tests for our method against the benchmarks. Error bars ($\mu \sim$ 95%CI), p-value, and significance level are calculated from the Mann-Whitney test. Effect size is calculated using Cohen's d

Figure 2 presents the results for this task in comparison with benchmarks using the Colorgorical palette. Our proposed method surpasses the benchmarks characterized by fixed color and opacity assignment, in both error rate and response time. Notably, our algorithm significantly outperforms certain benchmark conditions(*Local*) in reducing error rate. Although the difference is not pronounced, our algorithm still maintains an advantage in terms of time efficiency compared to other methods.

It also showcases the results for this task compared with benchmarks using the Tableau palette. Specifically, our optimization achieved a significantly reduced error rate compared to certain benchmarks (*Hue-preserving*). While our optimization does not significantly outperform all the other benchmarks in terms of error rate and response time, it still holds a slight advantage in terms of overall results.

B. Results for Class Discrimination

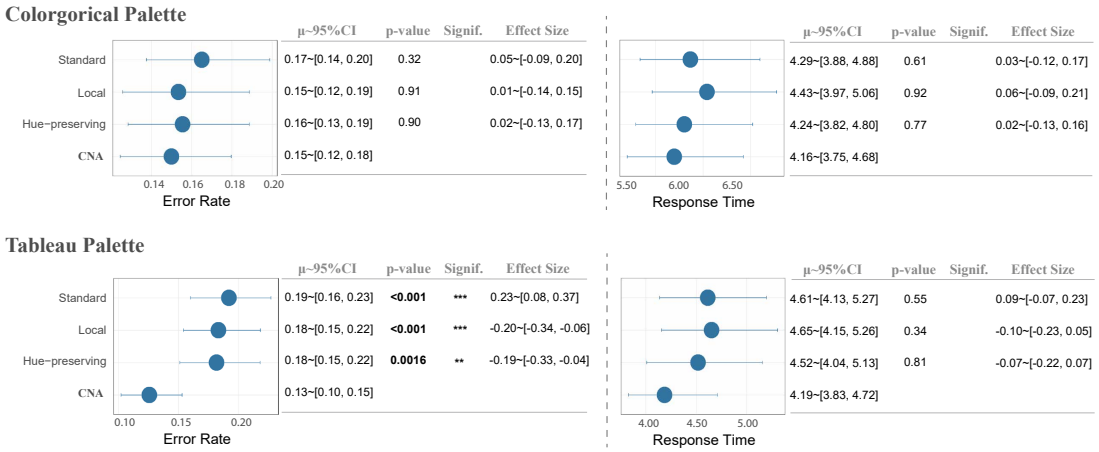


Fig. 3: Results for the *Class Discrimination Task*

The results are depicted in Figure 3. In comparison with colors sampled from the Colorgorical palette, the error rate with our optimization method consistently remained lower than that of the fixed assignment strategy.

It also presents the results for this task. Our proposed method surpasses the benchmarks that utilize colors sampled from the Tableau palette, characterized by fixed color and opacity assignment, in both error rate and response time. Notably, our algorithm significantly outperforms all benchmark conditions (*Standard*, *Local*, *Hue-preserving*). This demonstrates the robustness of our approach in class discrimination tasks, particularly when compared with designer-crafted palettes like Tableau.

C. Results for User Preference

For each comparison, we assigned a score of 1 if the participant preferred our approach and -1 for preferring one of the other benchmarks. Zero was assigned for a neutral choice when the participant indicated no clear preference. The results are depicted in Figure 4, with a positive score indicating a preference for our technique.

Compared with colors sampled from the Colorgorical palette, our method appears preferable to *Hue-preserving*, and comparable to the other two benchmarks (*Standard*, *Local*). As for the Tableau palette, our optimization has a slight advantage over all three benchmarks (*Standard*, *Local*, *Hue-preserving*). However, none of these comparisons reached statistical significance. This indicates that in terms of aesthetics, our approach is likely similar to the benchmarks, suggesting that our method maintains visual appeal while enhancing other aspects of visualization.

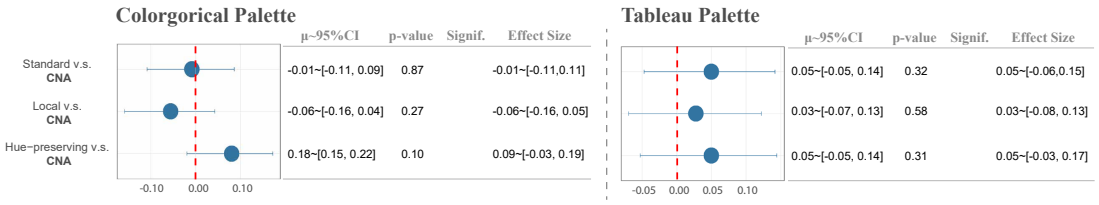
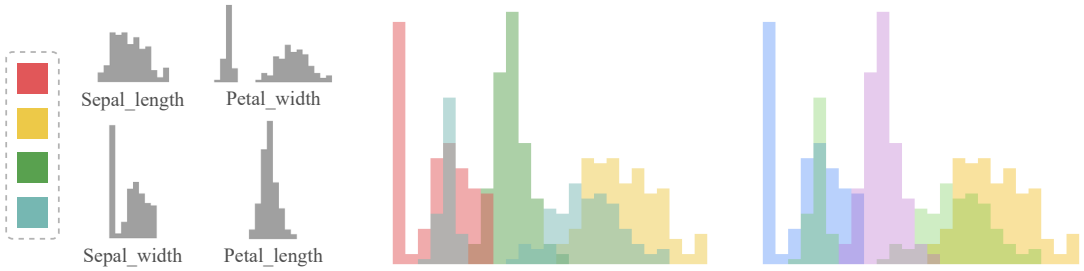


Fig. 4: Results for the *User Preference Task*. A preference score greater than 0 indicates that participants prefer our method over the respective benchmark.

III. Visualizing Iris Dataset with semi-transparent overlapping histograms

To further explore the potential of our proposed method, we conducted this case study on the well-known Iris dataset [2], to compare the distribution difference of the four variables: sepal width, sepal length, petal width, and petal length. The histogram is first calculated by 25 bins (see Figure 5-a) and then colorized with the Tableau 10 palette. Following the online course for overlapping histograms with Matplotlib in Python [6], we set the opacity of each color to 0.5. As shown in Figure 5-b, with the setting of the same opacity and default color assignment, the distribution of sepal width(the green class) is hard to discriminate, due to the similar appearance to the mixed color from petal length(blue class) and sepal length(yellow class). The gray color from petal width(red color) and petal length(blue class) are very distinctive from the surrounding colors.



(a) Input Palette & Data Distributions (b) Random Color Assignment, Rendering Order & Same Opacity (c) **CNA**

Fig. 5: Visualizing the Iris dataset [2] with the overlapping histogram. With the input palette and data distributions (a), apply the default color assignment and the same opacity to the histogram (b); using our method, we automatically generate optimal color, opacity, and rendering order (c). Our optimization result achieves better readability of the overlapping histograms.

Our optimization result avoids these problems, as shown in Figure 5-c, we can see that our optimization removes the false colors by optimizing the color, hence the purple class, i.e., the petal length class, can be easily distinguished. Our algorithm also has a clear distribution for each class.

IV. Visualizing CIFAR100 Dataset with Cluster-encapsulating Hulls

Cluster-encapsulating hulls are often used as an alternative for scatterplot visualizations [1], [7], as they allow a better perception of the distribution of whole clusters. However, due to the potential overlapping of complex shapes, this kind of visualization may lead to ambiguous colors [5]. To illustrate the applicability of our method in this domain, we generated cluster-encapsulating hulls from four CIFAR100 object classes [4]. As shown in Figure 6-a, we first colorized the hulls using the Tableau-10 palette with random color assignment. Using a fixed opacity of $\alpha = 0.5$ (see Figure 6-b, the distribution of the green class becomes ambiguous, especially for parts overlapping with the red class. Applying a local color blending model (Figure 6-c) improves the depth order perception, but still does not reduce false colors. Using our optimization, the optimal color and opacity assignments reduce false colors. For example, the different parts of the orange cluster are easier to discriminate, as compared with a randomly assigned green color in Figure 6-b & c. The extent of clusters is now more clear with our method (Figure 6-d), and the visualization overall can be read with minimal ambiguity.

V. Palettes and histograms used in the evaluation

We developed a collection of 18 multi-class overlapping histograms, each consisting of two, three, or four classes. As shown in Figure 7, these histograms exhibited a diverse array of characteristics,

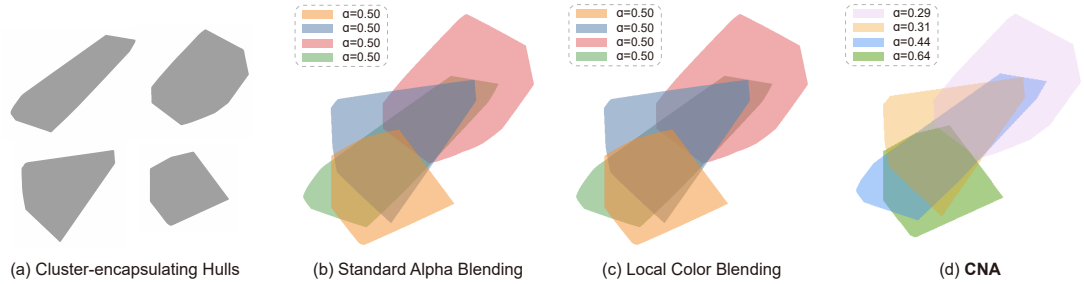


Fig. 6: Visualizing the CIFAR100 dataset with cluster-encapsulating hulls. (a) Input cluster-encapsulating hulls for each class; (b) applying random color assignment and uniform opacity(0.5) with standard alpha blending model to the hulls, leading to many false colors; (c) using local color blending model does not reduce false colors; (d) our optimal colors, opacity settings, and rendering order, make the shape distribution of each class easy to figure out, while also ensuring good color separability for all segments.

encompassing varying degrees of distribution smoothness and overlap. To color-code these histograms, we employ color palettes obtained equally from Tableau[9] and Colorgorical[3], ensuring a balanced application between the two sources. We maintain a uniform opacity of 0.5 and a consistent rendering order to simulate typical usage scenarios. Depending on the number of classes in the histogram, we randomly select sub-palettes comprising two, three, or four colors. The stimuli corresponding to the four comparison methods—standard alpha blending, local color blending, hue-preserving color blending, and our color-name aware optimization (CNA)—are presented in Figure 7, Figure 8, Figure 9, and Figure 10, respectively. The first three benchmark methods use the same palette, uniform opacity of 0.5 and consistent rendering order, compared with our optimized results (CNA), allowing a direct comparison of how each blended method handles the color coding of overlapping histograms.

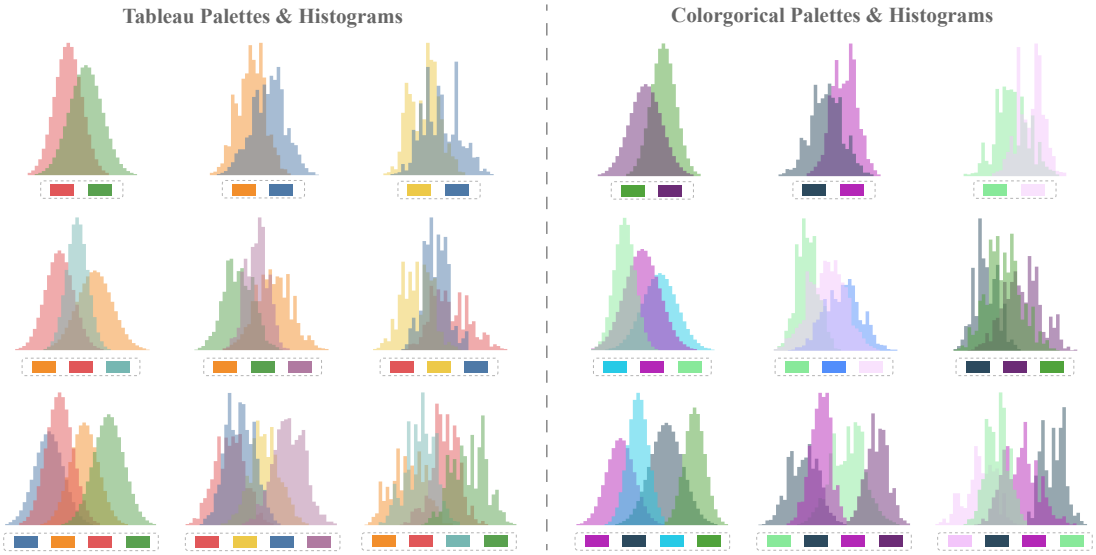


Fig. 7: Stimuli used in the evaluation section, generated with both Tableau and Colorgorical palettes, applying a uniform opacity of 0.5 and consistent rendering order, based on the *standard alpha blending model*.

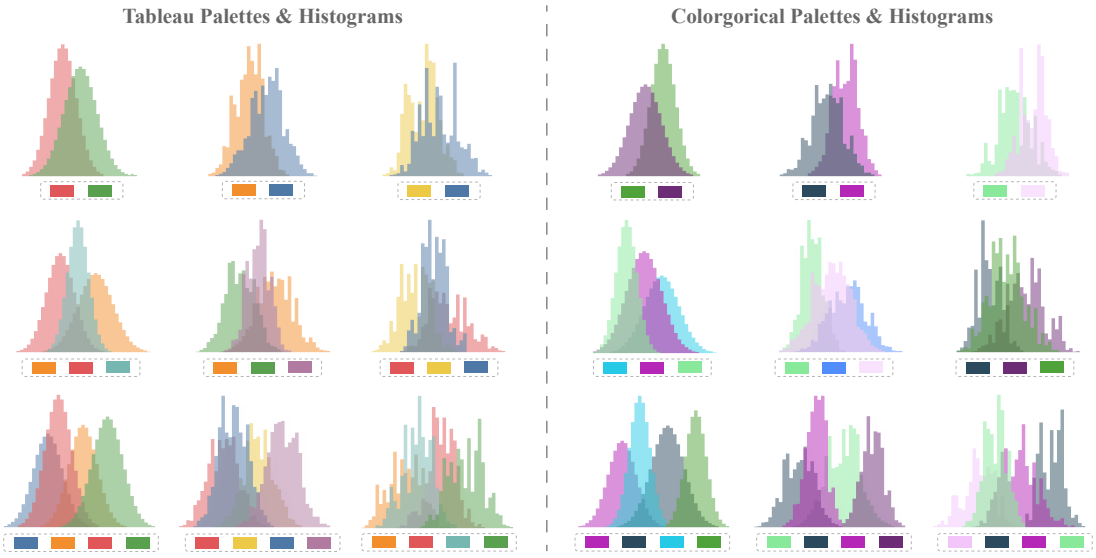


Fig. 8: Stimuli used in the evaluation section, generated with both Tableau and Colorgorical palettes, applying a uniform opacity of 0.5 and consistent rendering order, based on the *local color blending model*.

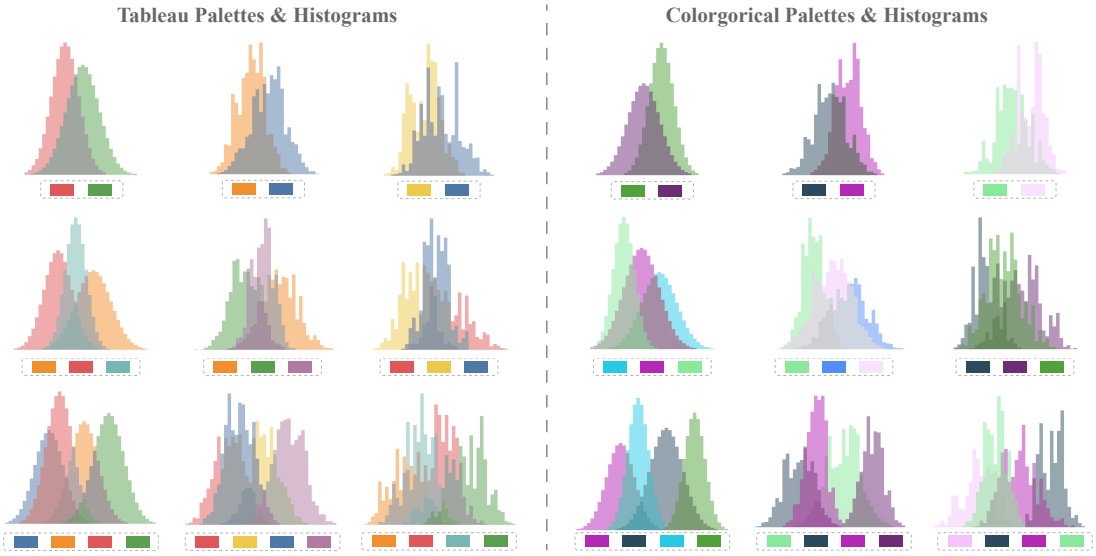


Fig. 9: Stimuli used in the evaluation section, generated with both Tableau and Colorgorical palettes, applying a uniform opacity of 0.5 and consistent rendering order, based on the *hue-preserving color blending model*.

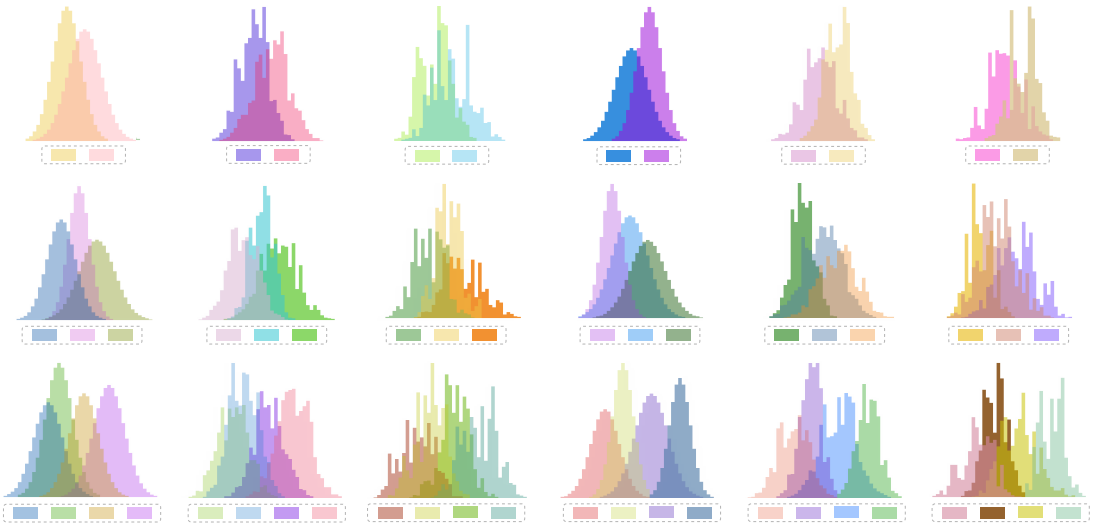


Fig. 10: The stimuli used in the evaluation section, generated by our color-name aware optimization (*CNA*). This approach automatically determines optimal settings for color, transparency, and rendering order to produce the final visualizations, ensuring clear discriminability of all segments while enhancing the perception of the whole from its constituent parts. The palettes displayed beneath each histogram are rendered using the auto-generated colors along with their corresponding opacity settings.

REFERENCES

[1] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1539–1148, 2008. doi: 10.1109/TVCG.2008.153

[2] R. A. Fisher. Iris. UCI Machine Learning Repository, 1988. DOI: https://doi.org/10.24432/C56C76.

[3] C. C. Gramazio, D. H. Laidlaw, and K. B. Schloss. Colorgorical: creating discriminable and preferable color palettes for information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):521–530, 2017. doi: 10.1109/TVCG.2016.2598918

[4] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[5] M. Luboschik, A. Radloff, and H. Schumann. A new weaving technique for handling overlapping regions. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pp. 25–32, 2010. doi: 10.1145/1842993.1842999

[6] riyaaggarwal. Overlapping histograms with matplotlib in python. https://www.geeksforgeeks.org/overlapping-histograms-with-matplotlib-in-python/, 2020. Accessed: 2020-11-26.

[7] T. Schreck, M. Schüßler, F. Zeilfelder, and K. Worm. Butterfly plots for visual analysis of large point cloud data. 2008.

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

[9] T. Software. The tableau visualization system. http://www.tableausoftware.com/.