應用色票與色彩和諧引導之向量圖標著色

林妙
國立臺灣大學

沈奕超
東京大學

秦孝媛
國立臺灣大學

陳若曦
國立臺灣大學

陳炳宇
國立臺灣大學

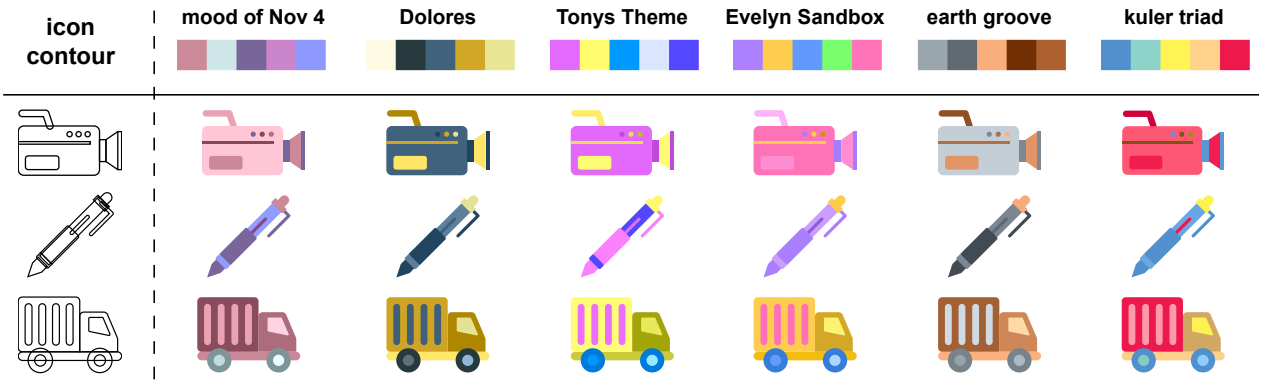

Figure 1: Given an input icon contour in vector format, our method can generate different colorized icons using different color palettes. Designers can further customize the colorization results directly in vector format.

ABSTRACT

我們提出了一種無需光柵化即可為向量圖標著色的方法。給定向量圖標的輪廓以及包含五種顏色的色票，我們的方法會根據色票生成向量圖標的多種著色結果。考慮到圖標是由各種幾何元素所構成的一種抽象表現，因此我們除了關注每條曲線的局部特徵，也將曲線之間的連接關係作為全局特徵納入考量。此外，我們結合兩個控制項來調節顏色之間的多樣性與和諧性，使圖標的顏色組合能夠更符合人類偏好與感知。與過去不同的是，我們的方法完全是在向量格式的條件下進行，不須經過任何光柵圖與向量圖之間的轉換。我們和過去基於像素之光柵圖標的上色方法進行比較，以證明我們提出的上色工具與著色結果是足夠有效的。

KEYWORDS

icon colorization, vector graphics, palette, color harmony

1 INTRODUCTION

Nowadays, icons are widely employed in posters, websites, banners, and other design interfaces [50]. These icons can be used to decorate pages or clearly explain the semantics by placing them beside the text, facilitating visual comprehension [12, 19, 20, 40]. However, proficient design skills are necessary for creating icons with compelling visual expression. An essential step in icon design is the process of coloring. During this stage, designers may need to consider various color references, such as text descriptions, color palettes, or other reference patterns. Additionally, they must take into account constraints imposed by color conditions, overall icon style, and aesthetic considerations for color combinations. Furthermore, due to the varying demands for icon styles in different contexts, designers must approach each design case-by-case. This process can be time-consuming and labor-intensive.

In studies on line art colorization, neural-based methods such as convolutional neural networks (CNNs) [30] and generative adversarial networks (GANs) [14] have been widely utilized for coloring icons, sketches, and animated characters. For instance, Sun *et al.* [50] proposed a dual conditional GAN that simultaneously regulates the icon outlines from the content input and the color combinations from the color reference icons. Li *et al.* [34] employed an encoder-decoder network for icon coloring and implemented conditional normalizing flow to enable the network to generate different results based on user-specified styles. While these works have achieved impressive results in icon coloring, they are limited to pixel-based objects, also known as raster images. Raster images have a fixed pixel size, which can result in image aliasing after zooming. However, for graphics and industrial design, including icon design, a more precise format is needed to describe visual content. In this case, vector graphics are used, which consist of parametric lines and curves instead of pixel arrays, allowing for infinite scaling without distortion [11, 66].

We aim to implement vector icon coloring. We utilize professionally designed icons in scalable vector graphics (SVG) format to learn the features and essential rules of icon coloring. Then, we map them to appropriate color combinations based on the user's

CGW ' 23, July 11–12, 2023, Taichung City, Taiwan

https://doi.org/XXXXXXX.XXXXXXX

selected color palette. Inspired by the work of Jiang *et al.* [26], who used graph neural network (GNN) for object detection and classification in vector graphics such as floor plans and diagrams, we build upon their dual-stream GNN architecture to extract features from each curve in the icon. Considering that icons are abstract representations composed of multiple geometric components, we use GNN to identify feature relationships among the curves of icons. Furthermore, in graphic design, harmony is a critical concept that directly affects the visual experience for users [19]. An important aspect is efficiently choosing the right color combination for objects or interfaces. Therefore, we incorporate two control mechanisms to regulate the diversity and coherence of color combinations. Subsequently, we perform color transformation between the possible color combinations of the icon and the selected color palette through optimization. This approach allows for color expansion in terms of hue and lightness, enabling icons to exist beyond the colors provided by the color palette.

We compared our method with the state-of-the-art pixel-based approaches. Our pipeline allows vector-form icons as direct input without converting between raster and vector space. This method enables designers to use our output directly as a basis for subsequent coloring tasks, making vector icon coloring more effortless and efficient.

2 RELATED WORK

2.1 Color Harmony Evaluation

The evaluation of color aesthetics is an essential subject with a long history. Jacobson and Ostwald [25] considered that harmony equals order. Moon and Spencer [42] built a harmony classification. Itten [24] drew up basic color theory and developed applications of the color wheel. One well-known theory is the harmonic template on hue provided by Matsuda [41] and Tokumaru *et al.* [52] (Figure 2). They devised eight types of harmonic templates based on the color wheel. Each type except type N is defined as one or two contiguous gray blocks. When color combinations fall entirely into the gray area, they are defined as harmonious color combinations. Based on these templates, Cohen-Or *et al.* [8] proposed a color harmonization method that quantized image colors into histograms and increased image coordination. Zaeimi and Ghoddosian [59] combined these templates and Munsell color system [7] to advance an optimized color harmony algorithm.

In addition to the traditional color theorem, many studies analyze the harmony of colors through model training. Ou and Luo [45] investigated the harmony of two colors. They proposed essential factors that affect color harmony, such as any two colors equal in hue and chroma that differ only in lightness are harmonious. Ou *et al.* [44] predicted the overall harmony of the three-color combination. O'Donovan *et al.* [43] utilized an extensive online dataset to study color compatibility. Kita *et al.* [29] inherited this compatibility research and further studied the palette expansion while maintaining color harmony.

Our study uses online icons as target data. Through the designer's works, we aim to extract color-matching principles more suitable for icon coloring. At the same time, we adopt the widely used harmonic templates [41, 52] as the basis for judging color blending, encouraging the model to generate harmonious color combinations.

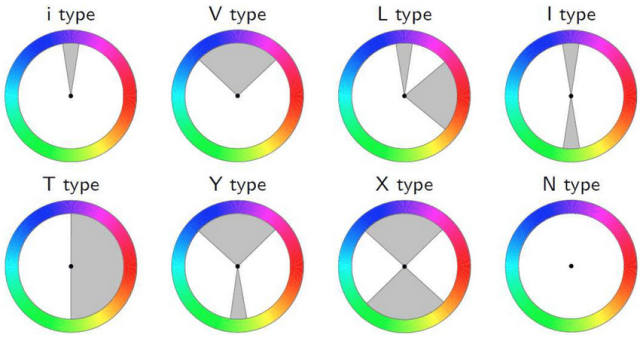

Figure 2: Harmonic templates on the color wheel. A combination of colors is considered harmonious, except N type, when all colors fall into the gray area. These gray regions can be rotated by any angle while maintaining relative areas.

2.2 Reference-Based Colorization

The task of colorizing nature images usually starts with a grayscale image. Lin *et al.* [36] used factor graphs for pattern coloring. Kim *et al.* [28] implemented a rule-driven approach to colorize segmented images. Zhang *et al.* [64] developed a user-guided method to lead image colorization. Another common practice is to use CNNs to complete coloring [22, 63]. Isola *et al.* [23] further implemented image-to-image translation tasks in various fields.

Unlike images, the initial stages of line art creation are usually not grayscale images. Stuff like comics, cartoons, and sketches usually only have black lines as input. GAN-based methods are commonly used to generate colors for line characters [18, 38]. Some works also use another picture or sketch as a reference [31, 35, 60]. These reference images are usually taken from the same category as the input sketch. It is expected that the same parts to get similar color transformations, such as hair color for hair and skin color for skin. Zhang *et al.* [61] proposed a semi-automatic method where users only need to give color hints instead of complex images. Ci *et al.* [6] allowed users to color the entire picture by drawing simple color lines. Kim *et al.* [27] did not require the user to choose a color but only to provide simple text prompts.

Icons are also a type of line art. But compared to sketches, they are more minimalist, more abstract, and even smaller. Sun *et al.* [50] first proposed the work of icon coloring. They used two discriminators to compute the structural and stylistic similarity of content and color reference icons. Han *et al.* [17] created a GAN network based on three conditions, adding a mask to limit the scope of the icon content to avoid mixing with the background. Li *et al.* [34] adopted an encoder-decoder architecture and applied normalizing flow so that the same style can generate various color combinations. Although these works ' results are appealing, since they are based on pixel icons, these methods are challenging to apply directly to vector graphics. Our goal is to extend icon coloring work to vector formats.

2.3 Image Recoloring

We define image recoloring as recoloring from color images, not grayscale images or sketches. A common practice in natural image recoloring work is to add color features from one image to another [16, 47, 51, 54]. Some works focused on luminance adjustment [1, 2, 37], some were based on the principle of harmony, ignoring lightness [8, 33].

In the absence of image references, some works are devoted to extracting the primary colors from the original image and presenting the picture in the form of a palette. Csurka *et al.* [10] used semantics and annotations for palette classification and color transfer. Yoo *et al.* [58] used the primary color of the image as the basis for image region segmentation. Wang *et al.* [55] formulated color adjustment as an interpolation problem and adjusted image color by user input sentiment words. Chang *et al.* [4] computed luminance and a*b* channels separately and corrected for colors outside the visual range. However, our color references are chosen by the user and do not refer to the color style of the original image.

Our requirements are closer to palette-guided recoloring work. Wang *et al.* [53] performed soft segmentation on the image, and users can apply local refinement according to the block of interest. Instead of using a color transfer function, Cho *et al.* [5] obtained realistic recolored images through image feature learning. Zhang *et al.* [62] regarded each color as a linear combination of some primary colors. Unlike real pictures, the area that icons can effectively segment is relatively small, and the segmented content does not necessarily contain substantive meaning. We refer to the traditional color transformation method proposed by Chang *et al.* [4]. Our task is palette-driven. We calculate lightness separately from chroma to transform color within icons ' simple lines.

2.4 Applications of Vector Graphics

Vector graphics is a sequence of parameters. The advantage is that this format can be scaled infinitely, ensuring the content is not distorted and is more precise than raster images. Zhang *et al.* [65] and Ha and Eck [15] first tried to simulate the generation of sketch strokes by sequence modeling. With the introduction of attention mechanism, Carlier *et al.* [3] and Ribeiro *et al.* [48] used a transformer-based architecture to learn feature representation and interpolation for vector sketches. Zou *et al.* [66] treated strokes as a vector format and expanded from sketches to generate realistic drawings in vector format. In addition to simulating stroke generation, there is more and more research on how to convert raster graphics directly into vector graphics or the converting between raster and vector space [39, 46, 49, 56].

Apart from the synthesis of vector graphics, how to analyze the characteristics of vector graphics is also an essential topic. In recognition of vector graphics, Li *et al.* [32] identified the category of the target object by extracting the keystroke features of the input vector sketch. Xu *et al.* [57] introduced a novel hashing loss for sketch retrieval. Collomosse *et al.* [9] learned a search embedding that unifies vector and raster representations. Jiang *et al.* [26] developed a dual-stream GNN architecture that does not require rasterization and realized object detection and classification of vector objects.

This paper refers to the method proposed by Jiang *et al.* [26] for analyzing vector context to recognize the composition of vector icons. Different from previous studies, the task of pattern recognition is often performed on objects with distinct features. For example, Jiang *et al.* [26] implemented object detection on floor plans and diagrams with standard format. However, icons are abstract expressions, which means that the elements of an icon do not necessarily have substance. Therefore, in addition to analyzing the independent features of each curve, we add group features to increase the recognition ability of the model for icon representations.

3 METHOD

Figure 3 shows our complete workflow. The input of our colorization pipeline is an icon contour and a five-color palette. We develop an algorithm consisting of two steps: generating tinting hints and performing palette-based color transfers.

3.1 Feature Extraction

Curves. Given an SVG icon contour, we use the Python API for Blender (bpy) to parse the SVG file. Convert the SVG icon to mesh to get diffuse colors, then flip the mesh back to the curve, and icon curves can be considered cubic Bézier. We normalize the icon size to 512*512 and get the coordinates of the start, end, and two control points. We also record the index of each curve so that we can apply the color back to the icon later. Since we first use the YOLaT model proposed by Jiang *et al.* [26] to extract curve features, according to their method, each point's coordinates, color, and width are used as the basic information of each node in the graph. We set the stroke color to black and the width to 6, simulating user input with only black lines. We also set the edge attributes of stroke-wise edges and position-wise edges [26], and then use the YOLaT dual-stream GNN [26] to obtain curve features. For detailed model architecture, please refer to the original paper.

Different from the original methods, they slice multiple proposals for each object and predict the class of each of them. In our case, the object is equivalent to an icon, and we expect one of the curves to be a color. For this, we do not slice the curves by different proposals, so one curve is considered as one proposal and the prediction target is the color of each curve. Since YOLaT [26] only deals with black and white vector graphic documents, which usually have a clear visual style for each category, such as Floorplans and Diagrams from the SESYD database[1]. Compared to our case, icons are colorful. Moreover, icons are abstract patterns; many curves may not necessarily have substantial meaning, such as cutting objects to represent shadows. Sometimes, after decomposing each icon curve, it is often a simple geometric shape, such as a circle or a rectangle. To sum up, it is difficult to distinguish the difference between curves using only curve features. So we add group connection to define the relationship between the curves further.

Groups. We use the curve features obtained by YOLaT [26] as the base feature for each curve. Joins two curves to form a group edge. Then define the coordinates of the bottom-left and the upper-right point of the i^{th} curve ' s bounding box as $(x_i,\ y_i)$ and $(X_i,\ Y_i)$, respectively. For each group, that is, each pair of curves, we define six group edge attributes as follows:

(1) $x_2 - x_1$

[1] http://mathieu.delalandre.free.fr/projects/sesyd/

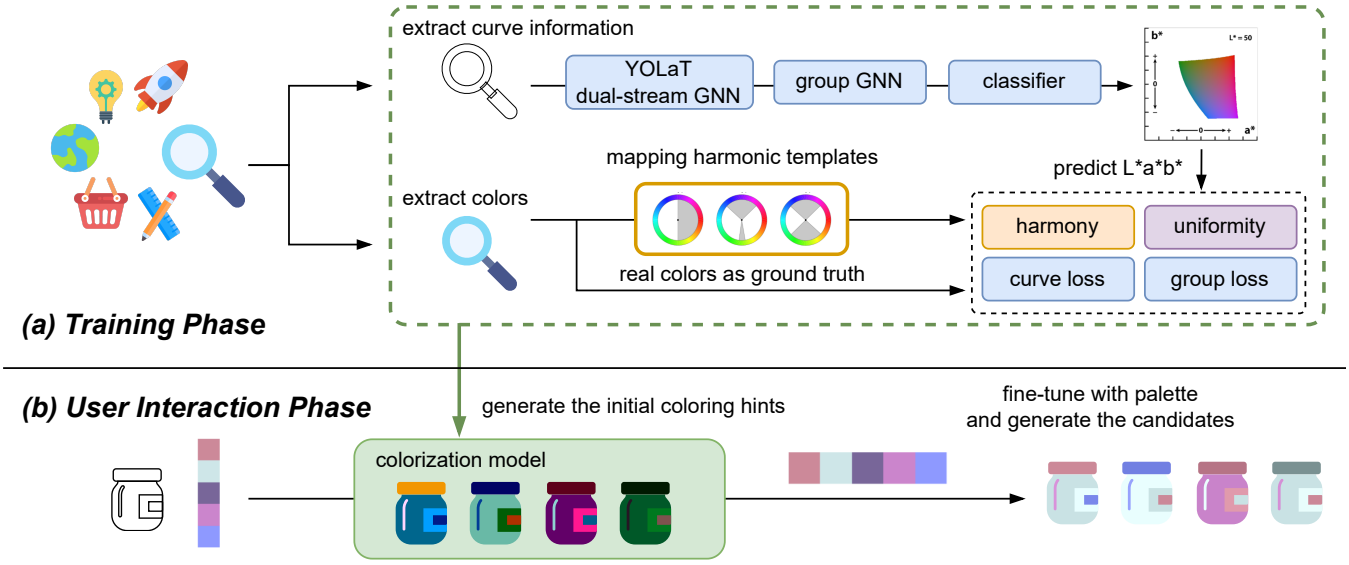

Figure 3: Overview. (a) In training phase, we use SVG icons as training input, extracting the curves' local and group features and using the GNN model to predict color hints of each curve (blue blocks). We use two adjustment terms to control the variety (purple block) and harmony (orange block) of predicted colors. (b) When the user enters an SVG icon outline and a color card, the colorization model (green blocks) will output the initial color hints, then fine-tune the colors according to the palette, providing a variety of color combinations.

(2) $y_2 - y_1$
(3) $X_2 - X_1$
(4) $Y_2 - Y_1$
(5) angle between $(x_1,\ y_1)$ and $(x_2,\ y_2)$
(6) angle between $(X_1,\ Y_1)$ and $(X_2,\ Y_2)$

Numbers one to four represents the distance between the bounding box coordinates of the two curves. Note that this distance includes directionality, which may be a negative number. Numbers five and six are the angles between the line segments connecting two points and the positive direction of the x-axis.

3.2 Color Prediction

Our model takes curve features and group attributes as input, predicting the color of each curve through two GNN layers. We predict colors in the CIELAB color space because it is the closest expression of color to human visual perception. In CIELAB color space, L* represents luminance, which ranges from 0 to 100; a* and b* represent green to red and blue to yellow, respectively, ranging from -128 to 127. We are not directly predicting the exact values of L*, a*, and b* because the mean-square error loss will tend to optimize the solution to the mean values [63], resulting in all the output curves being the same color. Therefore, we refer to the method Zhang *et al.* [63] used and transform regression into a classification problem.

We split L*, a*, and b* every ten bins so that L* has ten classes and a* and b* have 26. Since different chroma ranges that can be displayed by different luminance, to ensure that the predicted color is within a reasonable range, we regard a* and b* as a plane with 26×26 classes. The loss of the model is divided into two parts, for the curve and the group, respectively. We use the cross-entropy loss to calculate the ***curve loss***, defined as

$$CE_y = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{M} y_{i,c}\log(f(\hat{y}_{i,c})),\ y = L^*\ and\ a^*b^*\,, \tag{1}$$

$$CE = CE_{L^*} + CE_{a^*b^*}\,, \tag{2}$$

where N is the number of samples, and M is the number of classes. $f(\cdot)$ is an activation function; here we use softmax. y_i represents the real class, while $\hat{y}_i$ represents the probability distribution of the predicted class. Finally, we add CE_{L^*} and $CE_{a^*b^*}$ to get curve loss.

Some groups are randomly selected in each training iteration to calculate the ***group loss***. We use each curve ' s output vector of L* and a*b* to calculate the cosine similarity between curve pairs. And the similarity of ground truth colors is defined as

$$s_{i,j} = \exp(-\alpha \times D_{i,j}),\ \text{for } i, j = 1, \ldots, M\,. \tag{3}$$

D_{ij} is the Euclidean distance between the luminance and chroma classes of curve i and curve j. α is a hyperparameter set to 0.1 here. Since we divide the two continuous spaces into discrete categories, the closer the classes, the higher the similarity. Therefore, we use an exponential function to convert the class distance into similarity. We use MSE loss to measure the distance between the predicted and actual similarities between the two curves. Then add MSE loss for L* and a*b* to get group loss. Group loss aims to strengthen the connection between the training curves, expecting the model to better distinguish the difference between curves. But if the icon has only one curve, it would not have group loss and only use curve loss.

3.3 Uniformity Regulation

To avoid the predicted color difference being too small because the curves of the icons are all basic geometric shapes, we add ***uniformity regulation*** to maintain the diversity of output colors. We use the negative mean of the predicted a* and b* standard deviation, a common statistical method for testing uniformity.

$$\sigma_a = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{a}_i - \bar{a})^2}\,, \quad \sigma_b = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{b}_i - \bar{b})^2}\,, \tag{4}$$

$$Uniformity\ Regulation = -\frac{1}{2}(\sigma_a + \sigma_b)\,. \tag{5}$$

We expect model output to be more diverse instead of only recognizing fixed colors. Note that if the weight of the uniformity regulation is too large, the model will tend to output the colors of the extreme values, which means the icon will be full of green, red, blue, and yellow because this color combination maximizes the standard deviation. We finally set the uniformity weight to 0.01.

3.4 Harmony Control

We add the second adjustment, ***harmony control***, to maintain the visual harmony of the output color combination. We use harmonic templates on the hue wheel defined by Matsuda [41] and Tokumaru *et al.* [52] as a standard. Among the eight templates, type N is excluded because we restrict icons to have colors.

$$\begin{gathered} F(X, T(\alpha)) = \sum_{x \in X} (\min_{e \in E} \|H(x) - e\|) \cdot S(x)\,, \\ \alpha \in \begin{cases} [0, \pi), & \text{if the template is symmetric} \\ [0, 2\pi), & \text{otherwise} \end{cases}\,, \end{gathered} \tag{6}$$

$$A(X) = F(X, T_m(\alpha_0))\,. \tag{7}$$

In Equation 6, we first use the original colors (x) of the input icon (X) to fit the template (T) that best matches. Following the method of Cohen-Or *et al.* [8], we also take saturation (S) into account. Then, we use the angular difference as a measure of harmony. When the color falls entirely within the gray area, the distance is 0; otherwise, the hue distance will be calculated as the angle between the color and the nearest border of gray regions (E). After finding a matching template, we record its type ($T_m(\alpha_0)$) and effective region ($A(X)$) as the ground truth of color harmony. The predicted color combination is calculated to the hue distance of the ground truth template as harmony control. This distance will be significant at the beginning of training because the model has not learned enough features to predict good color combinations. So we take natural log to balance its control strength and set the weight to 0.01.

3.5 Recoloring with Palettes

After predicting the color combination of the input icon, which we called the initial hint, we compute the hue distance to match each template color to the closest palette color. We view the palette fitting as an optimization process as

$$f(H_{L^*a^*b^*}, P_{L^*a^*b^*}) = \sum_{i=1}^{n} \Delta E^*_{abi} + \lambda\,, \tag{8}$$

$$\Delta E^*_{ab} = \sqrt{(P_L{}^* - H_L{}^*)^2 + (P_a{}^* - H_a{}^*)^2 + (P_b{}^* - H_b{}^*)^2}\,, \tag{9}$$

$$\lambda = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\beta}{(\sqrt{(H_{iL^*} - H_{jL^*})^2} + 1) \times (\Delta E^*_{ab}(H_i, H_j) + 1)}\,. \tag{10}$$

The goal is to minimize the distance between each predicted color and the corresponding palette color. We label the hint and palette colors as H and P, respectively, and n is the number of curves. Since multiple predicted colors may correspond to the same palette color, their lightness should be different to maintain harmony and recognition [45]. So we add a adjustment term (Equation 10) to prevent the curve colors from being too similar. β is a constant; we set it to 1000. Additionally, the luminance of each curve will maintain its relative relationship during optimization. For example, if curve i is predicted to be lighter than curve j, then curve i will still be lighter than curve j after recoloring [4, 47, 51]. Note that in CIELAB color space, corresponding to different lightness, the adjustable range of a* and b* are different. It is necessary to keep the color within a reasonable range throughout the optimization process.

4 EXPERIMENTS

4.1 Datasets

Vector icons. We collect 13132 icons from Flaticon [2], a public website of icon creations by designers. All icons are in SVG format with flat colors and no borders. The icon outline is extracted from the path parameters mentioned in the feature extraction part. And we set the stroke color to black and the width to six to simulate user input. The ratio of the training, validation, and testing data is 0.8:0.1:0.1.

Palettes. We use the Kuler dataset provided by O'Donovan *et al.* [43]. Each palette contains a theme name and five different RGB colors. We randomly select some of these chips to present the results of palette coloring.

4.2 Implementation Details

We train the colorization network using curve and group loss with two regulation terms for fifty epochs. Cause we observe that the coloring results predict intense colors for training more iterations. That is, the predicted color combination will consist mainly of red, green, blue, and yellow. Referring to YOLaT [26], we use Adam optimizer and set the learning rate as 2.5×10^{-4}. We set the batch size to 8, and the weights of curve loss, group loss, uniformity regulation, and harmony control are set to 0.2, 0.8, 0.01, and 0.01, respectively.

4.3 Results and Comparison

Figure 4 shows the colorization results of our model. Given a five-color palette and a SVG icon outline, our model can provide diverse coloring icons. We compare our coloring results to several iconic line art coloring works, Comic [13] applied to manga coloring, MUNIT [21] used to image-to-image translation, SCFT [31] and SGA [35] applied to sketches, AdvIcon [50] and Style-Structure [34] are specific to icon coloring. Since previous studies are not necessarily suitable for icons, we fine-tune each work using the pixel icon dataset provided by Sun *et al.* [50]. Implementations of these

[2] https://www.flaticon.com/

Figure 4: Coloring icons. Our colorization algorithm can provide designers with various color combinations as candidates.

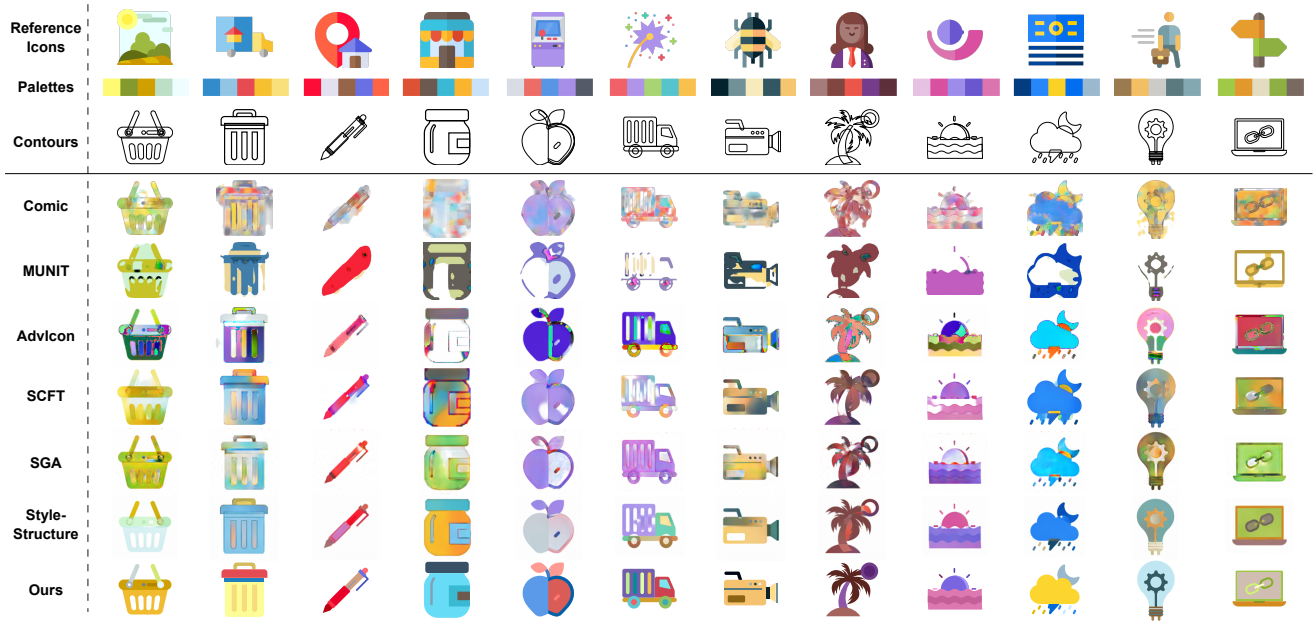

Figure 5: We compare the colorization results with the baselines. Our coloring is completed in vector format, while the baselines work in pixel images.

works were obtained from the authors of the papers. In addition, previous works require another image as a color reference, while we use the color palette directly. We select some icons as color references for comparison baselines to present the coloring results. Then, we use k-means to take five representative colors from these reference icons as our palette input.

Figure 5 shows all coloring results. Unlike our entire pipeline being in a vector environment, the works of baselines require rasterizing the vector icons before coloring. Moreover, their coloring results are also in pixel format, and designers need to perform an extra step to convert these pixel images into vector icons to use them. There is a concept of layers in vector graphics, which means that even if the designer has overlapping lines when designing the image contour, these lines can still be covered by the order of different layers after coloring. If icons with overlapping curves are directly rasterized, they will have more redundant cutouts, making pixel-based network coloring more difficult. Another advantage of coloring in vector format is that the image has apparent boundaries and reduces mis-coloring between the foreground and background.

4.4 Ablation Studies

We conduct ablation studies demonstrating that group loss and two adjustment terms contribute to icon colorization. When only curve loss is used (Figure 6 (a)), from the results of predicting luminance and coloring with the palette, many curves with similar shapes

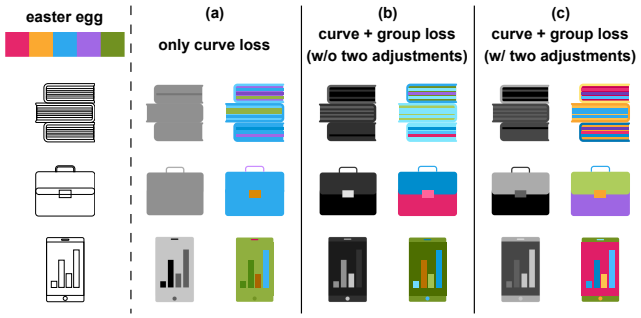

Figure 6: Ablation studies. The grayscale images are the luminance predicted by our model, and the color icons are the coloring results after fitting the palette. (a) Only use curve loss. (b) Use curve and group loss. (c) Our coloring results.

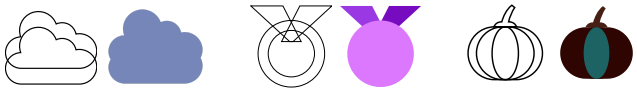

Figure 7: When similar shapes overlap, the model tends to assign them the same color, making it difficult to distinguish the figures, such as two overlapping clouds (left), two circles on the medal (middle), and similar shapes on the pumpkin (right).

are indistinguishable, such as rectangles. When adding group loss (Figure 6 (b)), similar curves can be identified by considering the relative relationship between the curves. However, the palette has a variety of colors. While the overall prediction of the book is primarily blue, and the mobile phone is dominated by green. After adding two adjustment terms, that is, our coloring results (Figure 6 (c)), add variety and harmony to the icon colors. Let the color combination of the icon better match the reference palette.

5 LIMITATIONS

Although our model produces stunning colorization results, many aspects remain for improvement. In icon design, a skill often used is to exploit a figure's symmetry or reuse the exact figure repeatedly. However, if an icon contains many curves of the same shape, the curve loss in our model tends to make them have the same color, as does the group loss. As a result, if similar shapes overlap, they may be tinted the same color, making the curves indistinguishable.

6 CONCLUSION

We present a palette-based colorization pipeline for vector icons without rasterization. We simultaneously combine vector icons' local and group features to strengthen the feature descriptions. Uniformity regulation and harmony control help our model to maintain variety and coordination in color combinations. In addition, we propose color extensions in terms of hue and value, enabling icons to exist beyond the colors provided by the palette without converting to other icons or images for color reference. Furthermore, our work is performed entirely in a vector environment without converting between raster and vector space. This approach enables designers to use our output directly as a basis for subsequent coloring tasks, making vector icon coloring more effortless and efficient. In the future, we aim to expand our pipeline to support gradient coloring and perform style transfer on vector icons.

REFERENCES

[1] Soonmin Bae, Sylvain Paris, and Frédo Durand. 2006. Two-scale tone management for photographic look. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 637–645.

[2] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. 2011. Learning photographic global tonal adjustment with a database of input/output image pairs. In *CVPR 2011*. IEEE, 97–104.

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

[4] Huiwen Chang, Ohad Fried, Yiming Liu, Stephen DiVerdi, and Adam Finkelstein. 2015. Palette-based photo recoloring. *ACM Trans. Graph.* 34, 4 (2015), 139–1.

[5] Junho Cho, Sangdoo Yun, Kyoung Mu Lee, and Jin Young Choi. 2017. Palettenet: Image recolorization with given color palette. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops*. 62–70.

[6] Yuanzheng Ci, Xinzhu Ma, Zhihui Wang, Haojie Li, and Zhongxuan Luo. 2018. User-guided deep anime line art colorization with conditional adversarial networks. In *Proceedings of the 26th ACM international conference on Multimedia*. 1536–1544.

[7] Sally Cochrane. 2014. The Munsell Color System: A scientific compromise from the world of art. *Studies in History and Philosophy of Science Part A* 47 (2014), 26–41.

[8] Daniel Cohen-Or, Olga Sorkine, Ran Gal, Tommer Leyvand, and Ying-Qing Xu. 2006. Color harmonization. In *ACM SIGGRAPH 2006 Papers*. 624–630.

[9] John Collomosse, Tu Bui, and Hailin Jin. 2019. Livesketch: Query perturbations for guided sketch-based visual search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2879–2887.

[10] Gabriela Csurka, Sandra Skaff, Luca Marchesotti, and Craig Saunders. 2010. Learning moods and emotions from color combinations. In *Proceedings of the seventh Indian conference on computer vision, graphics and image processing*. 298–305.

[11] Praveen Kumar Dhanuka, Nirmal Kumawat, and Nipun Jindal. 2019. Vector based glyph style transfer. In *ACM SIGGRAPH 2019 Posters*. 1–2.

[12] Sidong Feng, Suyu Ma, Jinzhong Yu, Chunyang Chen, Tingting Zhou, and Yankun Zhen. 2021. Auto-icon: An automated code generation tool for icon designs assisting in ui development. In *26th International Conference on Intelligent User Interfaces*. 59–69.

[13] Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. 2017. Comicolorization: semi-automatic manga colorization. In *SIGGRAPH Asia 2017 Technical Briefs*. 1–4.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020), 139–144.

[15] David Ha and Douglas Eck. 2017. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477* (2017).

[16] Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. 2011. Non-rigid dense correspondence with applications for image enhancement. *ACM transactions on graphics (TOG)* 30, 4 (2011), 1–10.

[17] Qin-Ru Han, Wen-Zhe Zhu, and Qing Zhu. 2020. Icon colorization based on triple conditional generative adversarial networks. In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*. IEEE, 391–394.

[18] Paulina Hensman and Kiyoharu Aizawa. 2017. cGAN-based manga colorization using a single training image. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 3. IEEE, 72–77.

[19] Tsuei-Ju Hsieh. 2017. Multiple roles of color information in the perception of icon-type images. *Color Research & Application* 42, 6 (2017), 740–752.

[20] Kuo-Chen Huang. 2008. Effects of computer icons and figure/background area ratios and color combinations on visual search performance on an LCD monitor. *Displays* 29, 3 (2008), 237–242.

[21] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal Unsupervised Image-to-image Translation. In *ECCV*.

[22] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. 2016. Let there be color! Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (ToG)* 35, 4 (2016), 1–11.

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

[24] Johannes Itten. 1970. *The elements of color*. Vol. 4. John Wiley & Sons.

[25] Egbert Jacobson and Wilhelm Ostwald. 1948. *Color harmony manual*. Container Corporation of America Chicago.

[26] Xinyang Jiang, Lu Liu, Caihua Shan, Yifei Shen, Xuanyi Dong, and Dongsheng Li. 2021. Recognizing vector graphics without rasterization. *NeurIPS* 34 (2021), 24569–24580.

[27] Hyunsu Kim, Ho Young Jhoo, Eunhyeok Park, and Sungjoo Yoo. 2019. Tag2pix: Line art colorization using text tag with secat and changing loss. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9056–9065.

[28] Hye-Rin Kim, Min-Joon Yoo, Henry Kang, and In-Kwon Lee. 2014. Perceptually-based Color Assignment. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 309–318.

[29] Naoki Kita and Kazunori Miyata. 2016. Aesthetic rating and color suggestion for color palettes. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 127–136.

[30] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[31] Junsoo Lee, Eungyeup Kim, Yunsung Lee, Dongjun Kim, Jaehyuk Chang, and Jaegul Choo. 2020. Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5801–5810.

[32] Lei Li, Changqing Zou, Youyi Zheng, Qingkun Su, Hongbo Fu, and Chiew-Lan Tai. 2018. Sketch-R2CNN: An attentive network for vector sketch recognition. *arXiv preprint arXiv:1811.08170* (2018).

[33] Xujie Li, Hanli Zhao, Guizhi Nie, and Hui Huang. 2015. Image recoloring using geodesic distance based color harmonization. *Computational Visual Media* 1 (2015), 143–155.

[34] Yuan-kui Li, Yun-Hsuan Lien, and Yu-Shuen Wang. 2022. Style-Structure Disentangled Features and Normalizing Flows for Diverse Icon Colorization. In *CVPR 2022*.

[35] Zekun Li, Zhengyang Geng, Zhao Kang, Wenyu Chen, and Yibo Yang. 2022. Eliminating Gradient Conflict in Reference-based Line-Art Colorization. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XVII*. Springer, 579–596.

[36] Sharon Lin, Daniel Ritchie, Matthew Fisher, and Pat Hanrahan. 2013. Probabilistic color-by-numbers: Suggesting pattern colorizations using factor graphs. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–12.

[37] Dani Lischinski, Zeev Farbman, Matt Uyttendaele, and Richard Szeliski. 2006. Interactive local adjustment of tonal values. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 646–653.

[38] Yifan Liu, Zengchang Qin, Zhenbo Luo, and Hua Wang. 2017. Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks. *arXiv preprint arXiv:1705.01908* (2017).

[39] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. 2019. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7930–7939.

[40] Pinjie Lv, Xinyue Wang, and Chengqi Xue. 2021. Research on Automatic Recognition Method of Icon Style. In *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, Vol. 1. IEEE, 931–935.

[41] Yutaka Matsuda. 1995. Color design. *Asakura Shoten* 2, 4 (1995), 10.

[42] Parry Moon and Domina Eberle Spencer. 1944. Geometric formulation of classical color harmony. *JOSA* 34, 1 (1944), 46–59.

[43] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2011. Color compatibility from large datasets. In *ACM SIGGRAPH 2011 papers*. 1–12.

[44] Li-Chen Ou, Patrick Chong, M Ronnier Luo, and Carl Minchew. 2011. Additivity of colour harmony. *Color Research & Application* 36, 5 (2011), 355–372.

[45] Li-Chen Ou and M Ronnier Luo. 2006. A colour harmony model for two-colour combinations. *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur* 31, 3 (2006), 191–204.

[46] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. 2021. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7342–7351.

[47] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. 2001. Color transfer between images. *IEEE Computer graphics and applications* 21, 5 (2001), 34–41.

[48] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. 2020. Sketchformer: Transformer-based representation for sketched structure. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14153–14162.

[49] I-Chao Shen and Bing-Yu Chen. 2021. Clipgen: A deep generative model for clipart vectorization and synthesis. *IEEE Transactions on Visualization and Computer Graphics* 28, 12 (2021), 4211–4224.

[50] Tsai-Ho Sun, Chien-Hsun Lai, Sai-Keung Wong, and Yu-Shuen Wang. 2019. Adversarial colorization of icons based on contour and color conditions. In *Proceedings of the 27th ACM International Conference on Multimedia*. 683–691.

[51] Yu-Wing Tai, Jiaya Jia, and Chi-Keung Tang. 2005. Local color transfer via probabilistic segmentation by expectation-maximization. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, Vol. 1. IEEE, 747–754.

[52] Masataka Tokumaru, Noriaki Muranaka, and Shigeru Imanishi. 2002. Color design support system considering color harmony. In *FUZZ-IEEE'02*, Vol. 1. IEEE, 378–383.

[53] Baoyuan Wang, Yizhou Yu, Tien-Tsin Wong, Chun Chen, and Ying-Qing Xu. 2010. Data-driven image color theme enhancement. *ACM Transactions on Graphics (TOG)* 29, 6 (2010), 1–10.

[54] Baoyuan Wang, Yizhou Yu, and Ying-Qing Xu. 2011. Example-based image color and tone style enhancement. *ACM Transactions on Graphics (TOG)* 30, 4 (2011), 1–12.

[55] Xiaohui Wang, Jia Jia, and Lianhong Cai. 2013. Affective image adjustment with a single word. *The Visual Computer* 29 (2013), 1121–1133.

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

[57] Peng Xu, Yongye Huang, Tongtong Yuan, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, Zhanyu Ma, and Jun Guo. 2018. Sketchmate: Deep hashing for million-scale human sketch retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8090–8098.

[58] Jae-Doug Yoo, Min-Ki Park, Ji-Ho Cho, and Kwan H Lee. 2013. Local color transfer between images using dominant colors. *Journal of Electronic Imaging* 22, 3 (2013), 033003–033003.

[59] Mohammad Zaeimi and Ali Ghoddosian. 2020. Color harmony algorithm: an art-inspired metaheuristic for mathematical function optimization. *Soft Computing* 24 (2020), 12027–12066.

[60] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. 2017. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan. In *2017 4th IAPR Asian conference on pattern recognition (ACPR)*. IEEE, 506–511.

[61] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. 2018. Two-stage sketch colorization. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.

[62] Qing Zhang, Chunxia Xiao, Hanqiu Sun, and Feng Tang. 2017. Palette-based image recoloring using color decomposition optimization. *IEEE Transactions on Image Processing* 26, 4 (2017), 1952–1964.

[63] Richard Zhang, Phillip Isola, and Alexei A Efros. 2016. Colorful image colorization. In *ECCV 2016*. Springer, 649–666.

[64] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. 2017. Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999* (2017).

[65] Xu-Yao Zhang, Fei Yin, Yan-Ming Zhang, Cheng-Lin Liu, and Yoshua Bengio. 2017. Drawing and recognizing chinese characters with recurrent neural network. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 849–862.

[66] Zhengxia Zou, Tianyang Shi, Shuang Qiu, Yi Yuan, and Zhenwei Shi. 2021. Stylized neural painting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15689–15698.