## ABSTRACT

We propose a method for colorizing vector icons without rasterization. Given a vector icon contour and a color palette containing five colors, our method produces multiple colorized icons. Considering icons as abstract compositions of various geometric elements, we incorporate both local curve features and the connectivity between curves into our model. Additionally, we introduce two controls to adjust the diversity and harmony of colors, ensuring that the color combinations better align with human preferences and perception. Unlike previous methods, our process is entirely vector-based, eliminating the need for any raster-to-vector conversion. We compare our method with pixel-based colorization techniques to demonstrate the effectiveness of our colorization tools and results.

### KEYWORDS

icon colorization, vector graphics, palette, color harmony

## 1 INTRODUCTION

In current design practices, icons are widely utilized in posters, websites, banners, and various graphic interfaces [50]. These icons can contribute to page decoration or clearly explain semantics by accompanying text, thereby facilitating visual comprehension [12, 19, 20, 40]. Nevertheless, proficient design skills are necessary to create icons with compelling visual expression. The process of coloring, during which designers need to consider various color references, such as text descriptions, color palettes, or other color references, is 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 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 introduced conditional normalizing flow to produce various outputs 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
works, we aim to extract color-matching principles more suitable without converting between raster and vector space. This method (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.

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.

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.

2.2 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. Čsurka 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” channel 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 vector space 

retrieval. Collomosse [15] 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. Colomosse 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 YOLOv3 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 YOLOv3 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 YOLOv3 [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 database1. 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 YOLOv3 [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 ith curve’s bounding box as (xi, yi) and (Xi, Yi), respectively. For each group, that is, each pair of curves, we define six group edge attributes as follows:

\[ (1) \quad x_2 = x_1 \]

1http://mathieu.delalandre.free.fr/projects/seyd/
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 colors 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 \( 26 \times 26 \) 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

\[
CE_y = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{M} y_{i,c} \log(f(y_{i,c})), \ y = L^* \text{ and } a^*b^*,
\]

(1)

\[
CE = CE_{L^*} + CE_{a^*b^*},
\]

(2)

where \( N \) is the number of samples, and \( M \) is the number of classes. \( f() \) is an activation function; here we use softmax. \( y \) represents the real class, while \( \hat{y} \) 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

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

(3)

\( D_{i,j} \) is the Euclidean distance between the luminance and chroma classes of curve \( i \) and curve \( j \). \( \alpha \) 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.

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

\[
\text{Uniformity Regulation} = -\frac{1}{2} (\sigma_a + \sigma_b). \tag{5}
\]

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.

\[
F(X, T(\alpha)) = \sum_{x \in X} \left( \min_{c \in E} ||H(x) - c|| \right) \cdot S(x),
\]

\[
\alpha \in \begin{cases} [0, \pi), & \text{if the template is symmetric} \\ [0, 2\pi), & \text{otherwise} \end{cases},
\]

\[
A(X) = F(X, T_m(a_0)) \tag{6}. \]

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 (\(T_m(a_0)\)) 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

\[
f(H_L', a', b', P_L, a^*, b^*) = \sum_{i=1}^{n} \Delta E_{ab}^* + \lambda, \tag{8}
\]

\[
\lambda = \sum_{i=1}^{n} \sum_{j=1}^{n} \beta \cdot \left( \frac{1}{(\Delta H_{L}^2 + \Delta L_{L}^2 + 1)^{1/2}} \cdot (\Delta E_{ab}^* (H_i, H_j) + 1) \right). \tag{10}
\]

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. \(\beta\) 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

https://www.flaticon.com/
Blues
Ketan’s Other Theme
He Edited
from the earth
Kind of Blue
Sanborn again
David Loops Them
Nicole

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 the 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...
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


Corporation of America Chicago.

Im2vec: Synthesizing vector graphics without vector supervision. In Incompatibility from large datasets. In Color harmony manual.


