Vectorized Colorization of Icon Line Art Based on Closed Contour Extraction

Ning Wang[1], Sen Ning[1], Yifei She[1], Bin Liu[2], Haojie Li[2], and Zhihui Wang[2(✉)]

[1] Software of Dalian Engineering, University of Technology, Dalian, China
[2] DUT-RU International School of Information Science and Engineering, Dalian University of Technology, Dalian, China
zhwang@dlut.edu.cn

Abstract. In the field of icon line art colorization, several Generative Adversarial Networks (GANs) based methods have achieved remarkable success. However, these methods often suffer from issues such as noise, color inconsistencies, and distortion when the generated color icons are enlarged. To address these challenges, we propose a novel approach for vectorized colorization of icon line art (LAVC), leveraging the principle of closed contour extraction. Specifically, our contour semantic descriptor (CSD) aims to fill the vector paths of the same descriptors with the same color for color inconsistencies. Our fusion model fuses the SVG line art, contour semantic descriptor, and color raster image generated from line art, to generate high-quality color vector icons without noise and distortion. Furthermore, we collect two datasets, IconLine and ClipLine, which provide high-quality line art and color image pairs for icons. Experimental evaluations conducted on our datasets demonstrate that our method outperforms existing techniques in terms of icon line art colorization, while maintaining distortion-free scalability.

Keywords: Icon Line Art Colorization · Line Art Vectorization · Contour Semantic Descriptor

1 Introduction

Vectorized colorization of icon line art aims to get a colorized and vectorized icon for a given line art. Owing to the non-distortive nature of vectorized colorization icons, it finds extensive applications in banners, billboards, websites, and applications, among others. Existing methods typically involve manual work using image processing software like Adobe Photoshop [2] and Illustrator [1] which is high labor costs for designers. Therefore, we propose an automatic approach to vectorize and colorize icon line art. With the advancements in deep learning, various tasks based on line art have emerged. Among them, the task of line art colorization [3,8,14,15] has gained significant attention, as it involves adding colors to line art and producing colored outputs. Specifically for icon line art,

N. Wang and S. Ning—Authors contributed equally.

V. C. M. Leung et al. (Eds.): Qshine 2023, LNICST 574, pp. 38–53, 2024.
https://doi.org/10.1007/978-3-031-65123-6_4

there is also a colorization method [25] available that focuses on icon line art colorization. However, these methods often generate icons with considerable noise, resulting in inconsistent colors. Moreover, the colored icons produced by these methods are raster images, which can lead to distortion when enlarged.

However, creating or editing vector images typically requires specialized software tools such as Adobe Illustrator [1] and Inkscape [19], making it challenging for amateur users. Recently, some techniques for generating vector images [17,20,23] have been developed. However, certain methods [20,23] only support the vectorization of colored raster images. Additionally, some of these methods [17] generate vector paths that are not closed contours, making it impossible to fill them with color. Therefore, the previously described methods are not well-solved for the simultaneous vectorization and colorization of icon line art.

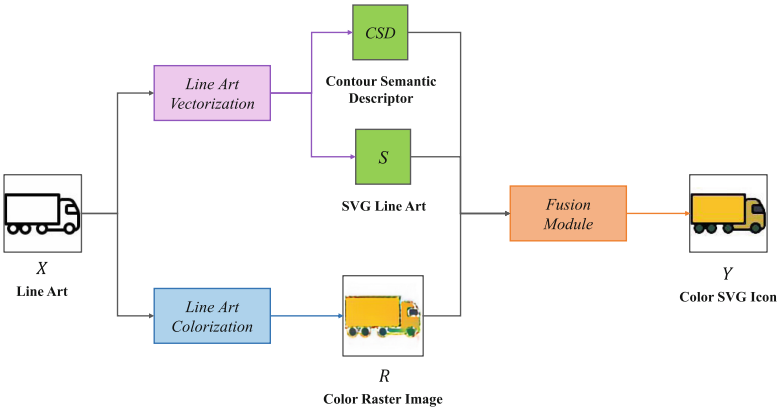

Fig. 1. Our vectorized colorization framework of icon line art (LAVC) based on closed contour extraction. Our approach primarily comprises two branches, line art vectorization and line art colorization.

We introduce a novel method for the vectorized colorization of icon line art (LAVC), leveraging the principle of closed contour extraction, to fill the closed SVG line art with the same color. The line art vectorization branch generates SVG line art, Contour Semantic Descriptor (CSD), while the colorization branch produces corresponding colored raster images independently. These components, including the SVG line art, CSD, and colored raster image, are then combined in our fusion module to produce a colored vectorized icon with less noise, consistent colors, and without distortion as shown in Fig. 1. Notably, our proposed Contour Semantic Descriptor (CSD) is specifically designed to address the challenge of color inconsistency in icons. We define features such as the contour shape, contour area, and contour shape parameters and combine them to get the CSD. We demonstrate that our proposed method generates superior results compared to the existing icon colorization methods. In summary, the key contributions of this paper are summarized as follows:

- We propose a novel method for vectorized colorization of icon line art (LAVC) based on closed contour extraction, which outperforms existing methods for icon line art colorization.
- We introduce a contour semantic descriptor (CSD) effectively addressing the issue of inconsistent icon colors.
- We collect two icon datasets, namely IconLine and ClipLine, which provide high-quality line art and corresponding colored image.

2 Related Work

Generative Adversarial Networks (GANs) have found extensive applications, such as generating realistic images [8], videos [27]. It has been widely employed in conditional image generation tasks, including image restoration [18], style transfer [22], image cartoonization [7], and image colorization [30].

2.1 Line Art Colorization

Line art colorization can generally be divided into two categories: stroke-based colorization and reference-based colorization. Ci et al. [8] proposed a user-guided line art colorization method based on Wasserstein GAN (WGAN) [4]. Dou et al. [11] proposed to generate more attractive images with harmonious color composition and fewer artifacts by exploring the drawing prior in HSV color space. Zhang et al. [29] proposed a split filling mechanism for flat images by explicitly controlling the 'influence areas' of the user color hints. Lee et al. [14] introduced a reference-based line art colorization method that utilizes self-enhanced references and dense semantic consistency. Sun [25] employs two discriminators for icon colorization based on Conditional GAN (CGAN) [16], structure, and color, respectively. Although line art colorization methods have become quite mature, they still struggle to generate icons with uniformly distributed colors and without distortion when enlarging raster images.

2.2 Image Vectorization

Image vectorization methods can be categorized into line art vectorization and color image vectorization. Mo et al. [17] proposed a virtual sketching framework for line art vectorization that utilizes a recurrent neural network (RNN). Su et al. [24] utilized deep reinforcement learning for getting the vectorization of manga strokes. Shen et al. [23] presented a deep-generation model for clip art vectorization that employs a recurrent neural network LSTM to sequentially synthesize vector paths and fill them with colors using a loss function. Dominici et al. [10] present PolyFit, a raster clip-art perception-aligned vectorization method that utilizes intermediate polygonal fitting. Zhu et al. [31] present triangular configuration B-spline (referred to as TCB-spline)-based vector graphics for raster image vectorization. Du et al. [12] proposed an automatic method to convert a raster image into layered regions of linear gradients to be saved in vector

graphics. However, all these methods only support the vectorization of colored raster images or line art vectorization, they cannot simultaneously handle the vectorization and colorization of line art.

3 Proposed Method

3.1 Overall Framework

The overall framework (LAVC) of the proposed algorithm in this paper is illustrated in Fig. 1. The input of our method is a line art (X), and it simultaneously executes the line art vectorization and line art colorization branches. The results obtained from these two branches, namely the SVG line art (S), contour semantic descriptor (CSD), and colored raster image (R), are fed into the fusion module, resulting in the final output, a colored SVG icon (Y).

3.2 Line Art Vectorization

The line art vectorization module takes the input of line art and performs the following steps: closed contour extraction, contour vectorization, and vector path optimization, resulting in the generation of SVG line art. Additionally, after the contour processing step, contour semantic extraction is performed to obtain the contour semantic descriptor.

Closed Contour Extraction. This step contains extraction of all closed contours, handling of duplicate contours, handling of open line segments, and contour sorting.

We use the FindContours algorithm [26] to extract all the closed contours from the image. Due to the nature of contour detection methods, both the outer contours and inner contours are detected simultaneously, which can result in duplicate contours. Hence, we calculate the intersection-over-union (IoU) ratio between each inner contour and its corresponding outer contour. If the IoU ratio exceeds a certain threshold θ_{IoU}, the contour is considered a duplicate. The decision to remove or retain duplicate contours can lead to different outcomes in the line art vectorization process.

Beyond duplicate contours, line art may also encompass standalone lines that are not connected to any other contours. These standalone lines need to be preserved to ensure that the structure of the vectorized and colored icon line art remains unchanged. In this paper, these independent lines are referred to as open line segments. A virtual width algorithm is proposed to determine whether a contour is an open line segment. The virtual width algorithm treats the width of the contour as approximately zero. By using the contour's area and perimeter, the virtual width of the contour can be approximated. If the virtual width of a contour is below a certain threshold θ_v, it is classified as an open line segment. The formula for calculating the virtual width of a contour is as follows:

$$width_v \approx \frac{contArea}{contLen/2} \tag{1}$$

where $width_v$ represents the virtual width of the contour, $contArea$ denotes the contour's area, and $contLen$ denotes the contour's perimeter.

SVG files are rendered following a strict order based on the sequence of elements. Those listed earlier in the file are rendered first, creating a potential for overlapping or obscuring subsequent elements. To mitigate issues of path overlap or occlusion when the SVG file is rendered as a raster image, it is indispensable to arrange the contours appropriately. In this study, we prioritize contours based on their area, sorting in descending order. This systematic arrangement ensures that smaller contours remain visible and are not overlapped. Notably, the Bubble Sort [5] algorithm is utilized to accomplish this task.

Contour Vectorization. In this step, each contour list is converted into a vector path in the SVG file format. For this conversion process, we propose the cnt2svg method contains three steps. Firstly, the width and height of the line art image are obtained, and they are written into the $\langle svg \rangle$ tag of the SVG file according to the specification. Then, the coordinate points in the contour list are written into each path element in the syntax of the SVG file format. Finally, the remaining information is added to complete the SVG file, resulting in the SVG vector file corresponding to the line art.

Vector Path Optimization. The parameters in the path elements generated in our method are presented in the form of coordinate points, which can result in curved edges of the vector paths that are not sufficiently smooth. To address this issue, we proposes path optimization methods specifically designed for circular and polygonal paths.

Circular path optimization involves determining whether each vector path represents a circle. When the circularity of a vector path exceeds a certain threshold θ_c, it can be classified as a circle. The formula for calculating circularity is as follows:

$$C = \frac{contArea}{\pi * max^2} = \frac{4\pi * contArea}{contLen^2} \tag{2}$$

where C represents circularity, max denotes the maximum distance between the contour center and all contour pixels, $contArea$ represents the contour area, and $contLen$ represents the contour perimeter. It is worth mentioning that the circularity of a contour can be calculated using the contour area and perimeter, without the need for the max parameter. When a vector path is identified as a circle, it undergoes optimization. The centroid coordinates of this circular path are obtained using the image moments. The radius of the circle is determined based on the centroid coordinates and points on the contour. Finally, the obtained parameters are converted into the $\langle circle \rangle$ element of the SVG file, resulting in a smooth circular vector path.

When a path is determined not to be a circle, the next step is to determine if it represents a polygon. In this paper, we use Ramer-Douglas-Peucker algorithm [21] for the fitting polygon. If the number of corners obtained from the polygon fitting of a contour falls between 3 and 6, the contour is classified as a

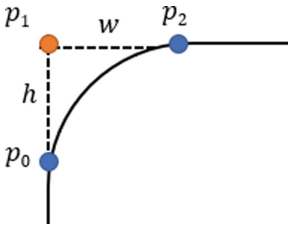

Fig. 2. Our polygon corner optimization method.

polygon. The optimization of the vector path is performed for these cases, and the polygon fitting function provides the number of corners and the coordinates of each corner. The method for optimizing the corners is illustrated in Fig. 2.

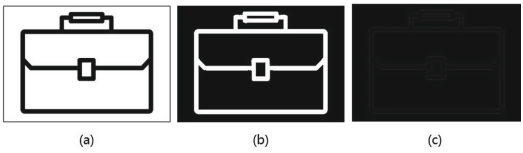

Fig. 3. The process of icon line art vectorized. (a) Icon line art. (b) Binarized icon line art. (c) Visualization of vector paths.

In the figure, p_1 represents a corner, while p_0 and p_2 are the start and end points of the quadratic Bezier curve. The black dashed lines (p_0p_1 and p_1p_2) represent the original edges of the corner, and the black solid line (p_0p_2) represents the optimized edge of the corner. During the optimization process, p_1 is set as the control point of the quadratic Bezier curve. The start and end points of the quadratic Bezier curve (p_0 and p_2) are selected on the two adjacent edges of the corner p_1. The quadratic Bezier curve replaces the corner in the path element of the SVG file, resulting in a smooth polygonal vector path. The process of generating vector path from input line art is shown in Fig. 3.

3.3 Contour Semantic Descriptor

Relying solely on filling closed contours with the same color as a constraint for achieving colored results is insufficient. For instance, in a simple facial expression consisting of 3 circles representing a face and two eyes, it is necessary to fill the two eyes with the same color while the face should have a different color. Therefore, this paper introduces the Contour Semantic Descriptor (CSD), which is extracted after the closed contour extraction step to obtain CSD. The colors of vector paths are subject to averaging and subsequent filling only under circumstances where there is a high degree of similarity between the Color Structure Descriptors (CSDs) of differing vector paths. The data storage format of CSD

is in the form of a list, comprising three elements: contour shape, contour area, and contour shape parameter. Each contour or vector path has its corresponding CSD. The contour shape is identified by a shape ID, obtained through the contour semantic extraction method (CSEM). The contour area is represented by an integer value. The contour shape parameter is detailed in Table 1, where parameters like radius and side length are represented as integers, and angles are represented by their cosine values. In the case of multiple side lengths, they need to be arranged in ascending order.

Table 1. The shape parameter of CSD

Contour Shape	Shape ID	Contour Shape Parameter
Circle	1	Radius
Triangle	2	Side lengths
Rectangle	3	Adjacent edge lengths
Parallelogram	4	Adjacent edge lengths and included angle
Right Trapezoid	5	Upper base, lower base and height
Isosceles Trapezoid	6	Upper base, lower base and height
Regular Pentagon	7	Side length
Regular Hexagon	8	Side length
Other	0	None

The contour semantic extraction method (CSEM) includes recognizing basic shapes, common quadrilaterals, and complex shapes. Basic shapes comprise circles, triangles, and rectangles. Common quadrilaterals include parallelograms, right trapezoids, and isosceles trapezoids. Complex shapes encompass regular pentagons and hexagons. Specifically, the recognition of basic shapes relies on circularity, triangularity, and rectangularity measurements. Common quadrilaterals are identified based on the parallel relationships among the edge vectors obtained through polygon fitting. Complex shapes are determined by the cosine values of the angles formed by the edge vectors.

In particular, we identifies isosceles trapezoids based on their area, where a normalized value closer to 0 indicates a shape closer to an isosceles trapezoid. The formula for calculating the area of an isosceles trapezoid is as follows:

$$area_{isT} = \frac{(up + down) \times len_{mid}}{2} \tag{3}$$

where $area_{isT}$ represents the area of the isosceles trapezoid, up denotes the length of the upper base, $down$ refers to the length of the lower base, and len_{mid} represents the distance between the midpoints of the parallel sides. The normalization formula is as follows:

$$norm_{area} = \frac{contArea - area_{isT}}{contArea} \tag{4}$$

where $norm_{area}$ represents the normalized result, and $contArea$ denotes the area of the contour.

When computing the similarity of semantic descriptors, the calculation follows the order of contour shape, contour area, and contour shape parameters. When determining whether contour shape parameters are equivalent, it is necessary to arrange the side lengths in ascending order. If there are two side lengths, their ratio is calculated. If there are multiple side lengths, they need to be normalized before comparison. Only when the contour shapes are identical, and the differences in contour area and contour shape parameters are not significant, can their semantic descriptors be considered identical. Through the aforementioned steps, contours with identical semantic contour descriptors can be obtained, allowing them to be colored with the same color.

3.4 Line Art Colorization

The line art colorization branch in our framework utilizes the IconGAN method proposed by Tsai-Ho Sun [25]. This method employs a conditional generative adversarial network (GAN) for icon colorization, where the icon structure and a reference icon are used as the structure and color conditions, respectively. These conditions are encoded and concatenated as input to the generator, producing a colored rasterized icon. The generator is guided by the structure discriminator and the color discriminator to generate icons that satisfy both the structural and color conditions. However, the colored icons generated by this method may exhibit noise and color inconsistencies, and the generated results are in a raster format, leading to distortion when scaled up. To address these issues, this method is combined with the line art vectorization method proposed in this paper, resulting in superior-quality colored vector icons.

The loss function of IconGAN consists of two parts: structural loss and color loss. The structural loss is represented as L_s:

$$\begin{aligned} L_s\left(G, D_s\right) = & \mathbb{E}_{y_c \sim P_1, y_s \sim S}\left[\log\left(1 - D_s\left(G\left(y_c, y_s\right), y_s\right)\right)\right] \\ & + \mathbb{E}_{x \sim p}\left[\log D_s(x, S(x))\right], \end{aligned} \tag{5}$$

where G represents the generator, D_s represents the structural discriminator, y_c represents the color reference image, y_s represents the structural sketch, x represents the real color icon, and $S(x)$ represents the structural image generated from the real color icon.

The color loss function is computed as follows:

$$\begin{aligned} L_c\left(G, D_c\right) = & \mathbb{E}_{y_c \sim P, y_s \sim S}\left[\log\left(1 - D_c\left(G\left(y_c, y_s\right), y_c\right)\right)\right] + \\ & \mathbb{E}_{x_1, x_2 \sim P, k(x_1) = k(x_2)}\left[\log D_c\left(x_1, x_2\right)\right], \end{aligned} \tag{6}$$

where L_c represents the color loss, D_c represents the color discriminator, x_1 and x_2 represent any two icons within the same cluster, and $k(x)$ represents the clustering index of x. Finally, the complete loss function is computed as follows:

$$L = \arg\min_{G} \max_{D_s, D_c}\left(L_s\left(G, D_s\right) + L_c\left(G, D_c\right)\right), \tag{7}$$

where L represents the total loss, L_s represents the structural loss, and L_c represents the color loss. This method takes the icon sketch and color reference image as conditions to generate colored icons. However, the quality of the generated colored icons may suffer from poor coloring effects. In a closed contour path, there may be inconsistencies in color. Hence, our framework is combined with our vectorization branch to achieve improved results in generating colored SVG icons.

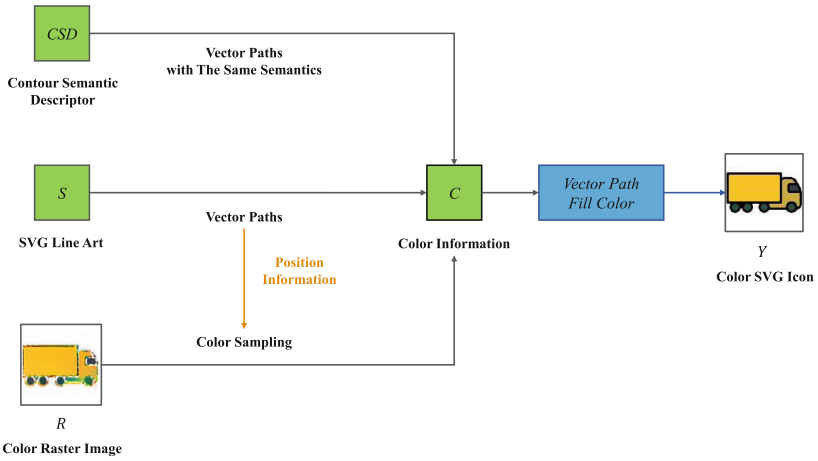

Fig. 4. Our fusion module.

3.5 Fusion Module

The fusion module in this paper, as shown in Fig. 4, combines the SVG line art (S), contour semantic descriptors (CSD), and a color raster image (R). Firstly, we extract all vector paths from S and obtain their positional Information. Leveraging the positional Information, we conduct color sampling in R to acquire color information (C) for these paths. Then, we identify the vector paths with the same semantics in CSD to ensure consistent colorization. Finally, the colors are filled into the corresponding vector paths, resulting in the final colored SVG icon.

When calculating the position information of a vector path, it is necessary to identify the extreme points of the path in the x and y directions. In this paper, these four points are referred to as the path extreme points. Additionally, the centroid position of the vector path is computed using image moments. In summary, the position information of a vector path consists of the coordinates of five points: the centroid, the leftmost point, the rightmost point, the topmost point, and the bottommost point.

When calculating the color information of a vector path, the first step is to determine whether the contour is an outer contour or an inner contour based on the contour's hierarchy. If it is an outer contour, it is filled with black color. If it is an inner contour, color sampling needs to be performed on the corresponding color raster image (R). During color sampling, the color is sampled at the path extreme points p and the centroid q on R. The raster image (R) tends to exhibit noise in the vicinity of its corresponding extremum points. However, there is less noise in the vicinity of the centroid. Hence, the color from the corresponding position in the raster image R is directly attributed to the centroid q. While for extreme points p, we compute the average color within the vicinity of p, with the mean values for the three channels of the RGB color spectrum being calculated independently. There are three conditions to assign a color to SVG vector paths expressed:

$$color = \begin{cases} extreme \quad points \quad p, & case1 \\ centroid \quad q, & case2 \\ \alpha_1 * p + \alpha_2 * q, & otherwise \end{cases}$$

case 1: the color difference between the extreme points and the centroid is greater than a threshold θ_{cd}, and the color calculation for the path is based on the corresponding color of the extreme points. case 2: the centroid coordinates are not within the interior of other higher-level paths, the centroid color sampling is still added. otherwise: the difference is smaller than the threshold, and the color calculation for the path is a weighted average of the colors of the extreme points and the centroid. This approach helps to avoid failures in coloring concentric or nested vector paths.

When filling colors for vector paths, the RGB colors are converted to hexadecimal format and written into the corresponding style element in the SVG file. It is worth noting that if multiple vector paths have the same semantic descriptor, after calculating colors individually for these paths, the average of these colors is taken as the color for those paths. This ensures color consistency among vector paths with the same semantic descriptor.

4 Experiments

4.1 Dataset

The task addressed by our method is vectorized colorization of icon lines, existing color images/line art datasets are not suitable for this task. Therefore, we propose two high-quality paired datasets: IconLine and ClipLine. IconLine and ClipLine contain 1162 color icon/line art image pairs and 1668 color clip/line art image pairs, respectively. All images are resized to 100×100 pixels as shown in Fig. 5.

We attempt two approaches for extracting line art from color images, namely XDoG [28] and Canny [9]. The line art produced by the XDoG algorithm [28] exhibits non-uniform background colors, significant noise, and inconsistent line thickness. Meanwhile, the line art generated by the Canny possesses a uniformly

Fig. 5. Two high-quality paired datasets: IconLine and ClipLine.

white background, no noise, and consistent line thickness. Therefore, we chose Canny for generating the line art. To simulate the real icon line art, we set the parameters of Canny with $\tau_1 = 100$, $\tau_2 = 200$. In the test dataset, we collect 240 high-quality real icon line art.

4.2 Experimental Settings

In our experiments, we perform acceleration using a 48G Nvidia A40 graphics card. The parameters are set as follows: In the closed contour extraction step, we set the IoU ratio threshold θ_{IoU} to 90% and the virtual width threshold θ_v to 10. In the line art colorization branch, we set the image resolution to 128×128. In addition, we combine the IconLine dataset and the ClipLine dataset together as the training dataset to enhance the generalization capability of IconGAN. During testing, we randomly select reference images from the training data, in order to increase the diversity of generated color styles. In the fusion module, we set the extreme points weight $\alpha_1 = 1$ and the centroid weight to $\alpha_2 = 4$. We place a greater emphasis on centroid color sampling because the color of the centroid more accurately represents the predominant color of the path. The color difference threshold θ_{cd} for each RGB channel is set to 64.

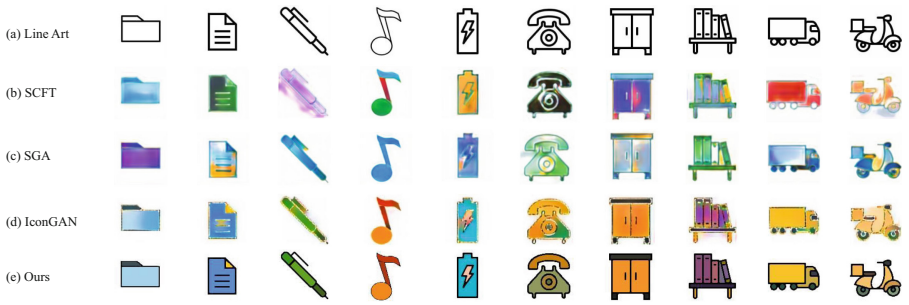

Fig. 6. Compared with the existing icon colorization method SCFT [14], SGA [15], and Icon GAN [25].

4.3 Qualitative Experiment

Figure 6 shows a qualitative comparison of the proposed method. It can be observed that the colorization results of SCFT [14], SGA [15], and IconGAN [25] show a significant amount of messy colors and noise, while our results present clean colors and higher semantic consistency. As IconGAN is purpose-built for icon coloring, it has delivered superior results when compared to SCFT and SGA. Hence, IconGAN [25] has been selected as the baseline model for our study. Additionally, our method successfully addresses the artifacts in colorization results. Only one color is filled in each vector path. Due to our contour semantic extraction method, the colors of the same semantic descriptor are consistent. For example, the wardrobe, books, and truck wheels are the same color. Finally, the color icons generated by those GAN-based methods are in raster format, leading to distortion when enlarged. In contrast, our results are vector icons, allowing for scaling without distortion.

4.4 Quantitative Experiments

Table 2. Evaluations of IS and FID on our method.

Method	IS	FID	MOS
SCFT	4.00	135.69	4.73
SGA	3.70	144.39	4.23
IconGAN(baseline)	4.62	126.85	5.03
LAVC(Ours)	4.39	82.72	6.39

We adopted Fréchet Inception Distance (FID) [13], Inception Score (IS) [6], and MOS as measurements for evaluating the generation performance. A lower FID value implies higher similarity, which typically indicates better visual effects.

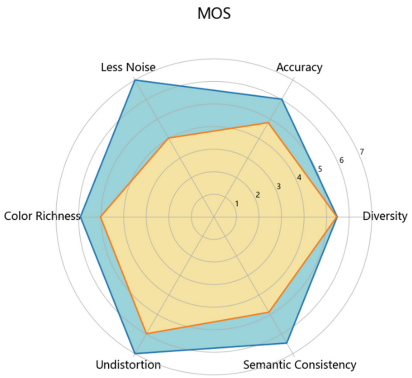

Fig. 7. MOS comparison of our LAVC (blue region) with our baseline Icon GAN(yellow region) [25] (Color figure online).

IS is based on the Inception Net-V3 model trained on the ImageNet dataset. A higher IS score indicates greater similarity to the ImageNet dataset. The obtained IS and FID scores are reported in Table 2. As can be seen, our method achieves a lower FID compared to IconGAN, indicating that our results exhibit a closer resemblance to the training data. Additionally, our method performs a lower IS score compared to IconGAN. This can be attributed to the presence of numerous complete color blocks in our results, but the images in ImageNet dataset often do not have such characteristics. In our study, we also employed the Mean Opinion Score (MOS) user study, which directly reflects human perceptual judgment regarding visual quality, to compare the performance of our method with IconGAN as shown in Fig. 7. We randomly selected 30 sketches from the test dataset. For each sketch, they were individually colorized using these methods. Subsequently, we invited 10 volunteers to assess the colorization results of these methods and assigned integer scores ranging from 1 (poor quality) to 7 (excellent quality) to each sketch. During this process, the volunteers were instructed to consider six criteria: diversity, semantic consistency, noise, distortion level, color richness, and accuracy. Finally, we utilized radar charts to visualize the comparative results of each MOS and calculated the average of the assigned scores as the MOS for each method. Our approach achieved the highest MOS scores across various dimensions.

4.5 Ablation Experiment

The ablation experiments of this study are shown in Fig. 8 and Table 3. In the column (c), the histograms and battery levels are not displayed. This is due to the sequential rendering of SVG files, where smaller vector paths are overlaid by larger vector paths, resulting in incomplete visualization of all vector paths' coloring results. The colors of the images in column (d) are obtained through color sampling of the images in column(b). However, in the last row of column

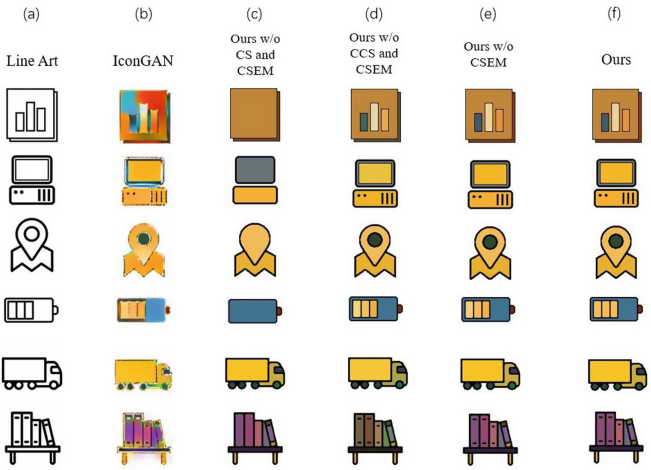

Fig. 8. Ablation experiment. CS represents contour sorting in closed contour extraction of line art vectorization. CSEM represents the contour semantic extraction module. CCS represents centroid color sampling in our fusion model.

(d), the colors of the book are different from those in column (b). This is because color sampling is only performed on the extreme points of the book's vector path, and the colors of the centroid are not sampled. As a result, due to the significant difference between the colors of the extreme points and the centroid in the vector paths of the books in columns (b), the main colors of the books in columns (d) and (b) are inconsistent. In column (e), it can be observed that the battery levels and identically sized books have inconsistent colors. This is because the CSEM is not executed, and thus, vector paths with the same semantic descriptors are not filled with the same color. In column (f), all modules of the proposed method are implemented, addressing the deficiencies and issues observed in the previous experiments. As a result, there is a significant improvement in the quality of generated results. As shown in Table 3, LAVC(Our) achieves the best results in both FID and MOS metrics, while LAVC(w/o CSEM) achieves the best result in IS. Therefore, we can confirm the effectiveness of each module within the proposed framework.

Table 3. Evaluations of IS and FID on our method. Bold indicates the best result.

Method	IS	FID	MOS
IconGAN(baseline)	4.62	126.85	5.03
LAVC(w/o CS and CSEM)	2.31	130.93	2.10
LAVC(w/o CCS and CSEM)	4.47	84.04	4.66
LAVC(w/o CSEM)	**4.66**	84.42	5.75
LAVC(Ours)	4.39	**82.72**	**6.39**

5 Conclusion and Limitation

In conclusion, this paper presents a novel method for vectorized colorization of icon line art (LAVC), effectively addressing issues of noise and distortion when raster images are enlarged. The innovation of our approach lies in the simultaneous execution of vectorization and colorization processes on line art and the introduction of a specific Contour Semantic Descriptor (CSD) for SVG line art, designed to tackle color inconsistencies. After training and testing on our assembled IconLine and ClipLine datasets, our proposed method demonstrated exceptional performance, highlighting its significant potential as a valuable tool for automated icon graphic design.

Limitation: The representation of our Contour Semantic Descriptor (CSD) is limited as it only considers three attributes: contour shape, contour area, and contour shape parameter, without taking into account the intrinsic semantic attributes of the line art itself. For instance, in Fig. 6, it is evident that the colors of the two motorcycle wheels in our results in column 9 are not consistent. This inconsistency arises due to significant differences in the Contour Semantic Descriptor caused by occlusion, resulting in the coloring of two identical wheels being inconsistent.

References

1. Adobe: Adobe illustrator (2023). www.adobe.com/products/illustrator.html
2. Adobe: Adobe photoshop (2023). www.adobe.com/products/photoshop.html
3. Anwar, S., Tahir, M., Li, C., Mian, A., Khan, F.S., Muzaffar, A.W.: Image colorization: a survey and dataset. arXiv preprint arXiv:2008.10774 (2020)
4. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International Conference on Machine Learning, pp. 214–223. PMLR (2017)
5. Astrachan, O.: Bubble sort: an archaeological algorithmic analysis. ACM Sigcse Bull. **35**(1), 1–5 (2003)
6. Barratt, S., Sharma, R.: A note on the inception score. arXiv preprint arXiv:1801.01973 (2018)
7. Chen, Y., Lai, Y.K., Liu, Y.J.: Cartoongan: generative adversarial networks for photo cartoonization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9465–9474 (2018)
8. Ci, Y., Ma, X., Wang, Z., Li, H., Luo, Z.: User-guided deep anime line art colorization with conditional adversarial networks. In: Proceedings of the 26th ACM International Conference on Multimedia, pp. 1536–1544 (2018)
9. Ding, L., Goshtasby, A.: On the canny edge detector. Pattern Recogn. **34**(3), 721–725 (2001)
10. Dominici, E.A., Schertler, N., Griffin, J., Hoshyari, S., Sigal, L., Sheffer, A.: Polyfit: perception-aligned vectorization of raster clip-art via intermediate polygonal fitting. ACM Trans. Graph. (TOG) **39**(4), 77-1 (2020)
11. Dou, Z., Wang, N., Li, B., Wang, Z., Li, H., Liu, B.: Dual color space guided sketch colorization. IEEE Trans. Image Process. **30**, 7292–7304 (2021)
12. Du, Z.J., Kang, L.F., Tan, J., Xu, K.: Image vectorization and editing via linear gradient layer decomposition (2023)

13. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Adv. Neural Inf. Process. Syst. **30** (2017)
14. Lee, J., Kim, E., Lee, Y., Kim, D., Chang, J., Choo, J.: 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, pp. 5801–5810 (2020)
15. Li, Z., Geng, Z., Kang, Z., Chen, W., Yang, Y.: Eliminating gradient conflict in reference-based line-art colorization. In: Avidan, S., Brostow, G., Cisse, M., Farinella, G.M., Hassner, T. (eds.) ECCV 2022. LNCS, vol. 13677, pp. 579–596. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-19790-1_35
16. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
17. Mo, H., Simo-Serra, E., Gao, C., Zou, C., Wang, R.: General virtual sketching framework for vector line art. ACM Trans. Graph. (TOG) **40**(4), 1–14 (2021)
18. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: feature learning by inpainting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2536–2544 (2016)
19. Project, I.: Inkscape (2023). https://inkscape.org/
20. Reddy, P., Gharbi, M., Lukac, M., Mitra, N.J.: Im2vec: synthesizing vector graphics without vector supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7342–7351 (2021)
21. Saalfeld, A.: Topologically consistent line simplification with the Douglas-Peucker algorithm. Cartogr. Geogr. Inf. Sci. **26**(1), 7–18 (1999)
22. Sanakoyeu, A., Kotovenko, D., Lang, S., Ommer, B.: A style-aware content loss for real-time hd style transfer. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 698–714 (2018)
23. Shen, I.C., Chen, B.Y.: Clipgen: a deep generative model for clipart vectorization and synthesis. IEEE Trans. Visual Comput. Graphics **28**(12), 4211–4224 (2021)
24. Su, H., Niu, J., Liu, X., Cui, J., Wan, J.: Vectorization of raster manga by deep reinforcement learning. arXiv preprint arXiv:2110.04830 (2021)
25. Sun, T.H., Lai, C.H., Wong, S.K., Wang, Y.S.: Adversarial colorization of icons based on contour and color conditions. In: Proceedings of the 27th ACM International Conference on Multimedia, pp. 683–691 (2019)
26. Suzuki, S., et al.: Topological structural analysis of digitized binary images by border following. Comput. Vision Graph. Image Process. **30**(1), 32–46 (1985)
27. Wang, N., et al.: Coloring anime line art videos with transformation region enhancement network. Pattern Recogn. **141**, 109562 (2023)
28. Winnemöller, H., Kyprianidis, J.E., Olsen, S.C.: Xdog: an extended difference-of-gaussians compendium including advanced image stylization. Comput. Graph. **36**(6), 740–753 (2012)
29. Zhang, L., Li, C., Simo-Serra, E., Ji, Y., Wong, T.T., Liu, C.: User-guided line art flat filling with split filling mechanism. In: Conference on Computer Vision and Pattern Recognition,(CVPR), pp. 9889–9898 (2021)
30. Zhang, L., Li, C., Wong, T.T., Ji, Y., Liu, C.: Two-stage sketch colorization. ACM Trans. Graph. (TOG) **37**(6), 1–14 (2018)
31. Zhu, H., Cao, J., Xiao, Y., Chen, Z., Zhong, Z., Zhang, Y.J.: TCB-spline-based image vectorization. ACM Trans. Graph. (TOG) **41**(3), 1–17 (2022)