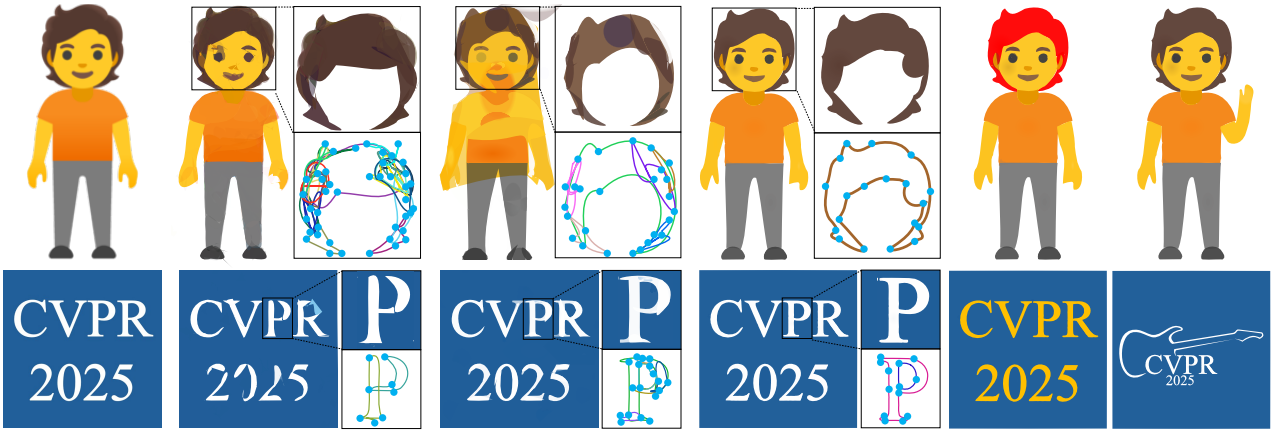
Less is More: Efficient Image Vectorization with Adaptive Parameterization

Kaibo Zhao[1], Liang Bao[1], Yufei Li[1], Xu Su[1], Ke Zhang[1], Xiaotian Qiao[1,2*]

[1]School of Computer Science and Technology, Xidian University, China
[2]Guangzhou Institute of Technology, Xidian University, China



(a) Raster Image (b) O&R [12] (c) SGLIVE [40] (d) Ours (e) Image Editing

Figure 1. Given a raster image as input (a), typical image vectorization methods, i.e., O&R [12] (b) and SGLIVE [40] (c), mainly rely on preset parameters (*i.e.*, a fixed number of paths and control points), and fail to produce pleasant results when the image structure is complex. In contrast, our method (d) can perform well with an adaptive number of paths and control point parameters based on the input image complexity, resulting in fast computation speed, high vectorization accuracy, and flexible editing applications (e), *e.g.*, image color adjustment, object animation, and icon customization.

Abstract

*Image vectorization aims to convert raster images to vector ones, allowing for easy scaling and editing. Existing works mainly rely on preset parameters (*i.e.*, a fixed number of paths and control points), ignoring the complexity of the image and posing significant challenges to practical applications. We demonstrate that such an assumption is often incorrect, as the preset paths or control points may be neither essential nor enough to achieve accurate and editable vectorization results. Based on this key insight, in this paper, we propose AdaVec, an efficient image vectorization method with adaptive parametrization, where the paths and control points can be adjusted dynamically based on the complexity of the input raster image. In particular, we first decompose the input raster image into a set of pure-colored layers that are aligned with human perception. For each layer with varying shape complexity, we propose a novel allocation mechanism to adaptively adjust the control point distribution. We further adopt a differentiable rendering process to compose and optimize the shape and color parameters of each layer iteratively. Extensive experiments demonstrate that AdaVec outperforms the baselines qualitatively and quantitatively, in terms of computational efficiency, vectorization accuracy, and editing flexibility.*

1. Introduction

Vector graphics [25] belong to a typical image format that uses primitive parameters like paths and control points, enabling visual content to be scaled and resized without quality loss. Compared to raster images with structured bitmap format, the efficiency of vector graphics in transmitting and storing information makes them popular for various applications. However, automatic high-quality vector graphics creation is still non-trivial, as it requires professional expertise and could be time-consuming.

Image vectorization, the procedure of converting a raster

Xiaotian Qiao is the corresponding author.

image to a vectorized one, provides an alternative way to utilize the existing large amount of raster images and avoid designing vector graphics from scratch. Early vectorization methods, including mesh-based ones and curve-based ones, mostly trace and approximate image contours using various geometric primitives, and may fail to achieve effective editing capabilities due to the complexity of the generated primitives and the loss of topology. With the rise of deep learning, data-driven methods learn to handle the vectorization of simple graphics or characters using deep neural networks, but often struggle to vectorize complex images. Recently proposed optimization-based methods alleviate such an issue powered by the differentiable rendering ability. However, they still suffer from a substantial amount of optimization time and computational resources, making them impractical in real scenarios.

We are thus interested in a fundamental question of the *impossible trinity*, i.e., whether efficiency, accuracy, and editability of image vectorization could be taken into account simultaneously given the complex raster image. As shown in Fig. 1 (b)(c), existing works mainly rely on a fixed number of preset paths and control points, ignoring the complexity of the image and posing significant challenges to practical applications. We demonstrate that such an assumption is incorrect, as preset paths or control points could be redundant when the input image structure is simple, and not enough when the structure is complex, limiting the editability and interpretability. This suggests that the vectorization parameters, including the paths and control points, should be adjusted adaptively based on the input image complexity.

Motivated by the above observation, in this paper, we propose AdaVec, an efficient image vectorization method with adaptive parameterization, where the paths and control points can be adjusted dynamically based on the complexity of the input image. In particular, given an input raster image, we first propose a multi-layer decomposition module to extract a set of pure-colored layers that are aligned with human perception. Second, we propose a control point simplification module to dynamically adjust the number of control points based on the shape complexity of each extracted layer. Finally, we use a differentiable rendering module to compose and optimize the shape and color parameters of each layer iteratively. As shown in Fig. 1, our method exhibits notable advantages in terms of computational efficiency, vectorization accuracy, and editing flexibility.

To thoroughly evaluate our method, we conduct extensive experiments on three public datasets. Results show that our method can produce more accurate vectorization results with faster computation speed, compared with the baselines. We further show the flexible editing ability of our method with a wide range of applications. Our main contributions are summarized as follows:

- To the best of our knowledge, we are the first to propose an efficient image vectorization approach to address the efficiency, accuracy, and editability issues simultaneously, allowing for fast computation speed, high vectorization accuracy, and flexible editing ability.
- We propose AdaVec, a novel vectorization approach with adaptive parameterization, where paths and control points can be adjusted adaptively based on image complexity.
- Extensive evaluations on public benchmarks demonstrate the state-of-the-art vectorization performance of our method quantitatively and qualitatively, and the benefits on downstream image editing applications.

2. Related Work

Traditional Methods. Early works aim to produce vector images with various geometric parameters, which can be roughly classified into mesh-based methods and curve-based methods. The mesh-based methods [5, 11, 18, 20, 32, 34, 36, 39, 41] divide an image into non-overlapping patches and interpolate colors across them. The curve-based methods [4, 24, 31, 35, 38] use Bézier curves with colors on both sides to create vector images, where a diffusion process is used to blend colors smoothly. While these methods can yield photo-realistic results, they may fail to achieve effective editing capabilities due to the complexity of the generated primitives and the loss of topology. Photo2ClipArt [9] and Du et al. [6] consider image layer decomposition. However, these methods not only rely on the target image but also require pre-provided segmentation maps with multiple assumptions, limiting their potential applicabilities. In contrast, our method decomposes raster images into layers aligned with human perception, enabling flexible image editing without relying on segmentation maps or pre-trained models. By dynamically adjusting the number of paths and control points based on image complexity, our method ensures adaptable and precise editability.

Learning-based Methods. Learning-based vectorization methods mainly focus on converting images into vector graphics on a specific scenario. SVG-VAE [21] models the drawing process of fonts using sequential generative models of vector graphics, providing a scale-invariant latent representation that can be systematically manipulated for style propagation. DeepSVG [2] proposes a hierarchical generative network for generating and interpolating complex SVG icons by disentangling high-level shapes from low-level commands. Im2Vec [28] generates complex vector graphics with varying topologies, requiring only indirect supervision from raster training images without the need for explicit vector graphics supervision. StarVector [29] combines visual representations from a CLIP [26] image encoder with CodeLLMs to generate complex SVGs by align-

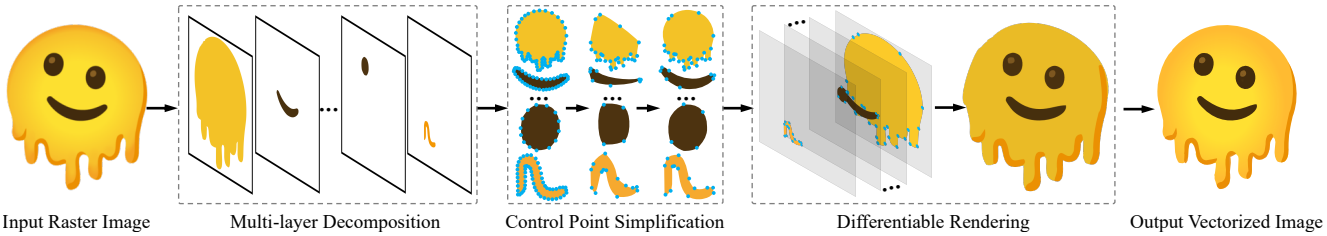


Figure 2. The overall pipeline of our image vectorization method. Given an input raster image, we first extract a set of vector paths that are aligned with human perception in the multi-layer decomposition stage. Then, in the control point simplification stage, we propose an adaptive simplification strategy to reduce redundancy in control points along the vector paths and optimize individual control points to avoid intersections. Finally, in the differentiable rendering stage, we compose the optimized layers and adjust the shape and color parameters iteratively to generate the output vectorized image.

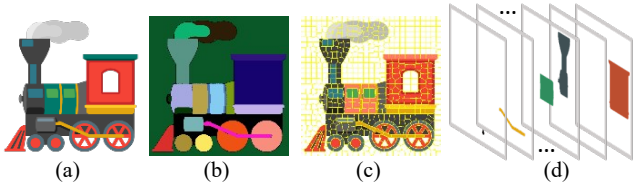


Figure 3. Multi-layer decomposition. Given an input raster image (a), we first generate a segmentation map (b) and a set of superpixels (c). We then conduct the filtering operations to obtain a set of decomposed layers(d) that are aligned with human perception.

ing visual and code tokens through next-token prediction. SuperSVG [13] proposes a two-stage self-training framework that combines coarse structure reconstruction and refined detail enhancement, achieving fast and high-precision image vectorization. Although these methods can handle image vectorization well in a single domain, they may struggle to vectorize complex out-of-distribution images.

More recently, optimization-based vectorization methods alleviate such an issue powered by the differentiable renderer [16, 19, 27], The vector image can be optimized iteratively by gradient descent without training on a specific dataset. LIVE [22] maintains image topology by progressively adding and optimizing Bézier paths with a layer-wise framework. O&R [12] optimizes Bézier curves through differentiable rendering and incorporates shape importance measures to reduce redundant shapes. The segmentation-based methods [40, 42] effectively enhance the quality of vectorization by providing more accurate initialization of paths. However, these methods still suffer from a substantial amount of optimization time and computational resources, making them less useful in practical applications. Different from the above works, we propose an efficient image vectorization method with adaptive parametrization based on the input image complexity, resulting in fast computation speed and high vectorization accuracy.

3. Approach

Our goal is to convert the raster image into a vectorized one with adaptive parameterization. Fig. 2 shows the overall pipeline of our method. Given an input raster image, we first extract a set of vector paths that are aligned with human perception in the multi-layer decomposition stage. Second, in the control point simplification stage, we propose an adaptive simplification strategy to reduce redundancy in control points along the vector paths and optimize individual control points to avoid intersections. Finally, in the differentiable rendering stage, we compose the optimized layers and adjust the shape and color parameters iteratively to generate the output vectorized image.

3.1. Multi-layer Decomposition

The initialization strategy of vector graphics plays a critical role in the subsequent optimization process. To ensure the flexibility of the vectorized results without losing image details, we combine semantic segmentation with superpixel partition. We first use the automatic masks generator from SAM [17] to generate a series of masks $M_S = \left\{m_1^s, \cdots, m_{N_1}^s\right\}, m_i^s \in \{0,1\}^{w \times h}$, where w and h represent the width and height of the image, respectively. However, as shown in Fig. 3(b), some detailed information is missing. We further adopt SLIC [1] to perform an initial superpixel partition of the image, as shown in Fig. 3(c). Due to the redundancy and over-segmentation in the superpixels, we subsequently use DBSCAN [8] to group all superpixels based on the average pixel values of each superpixel, forming multiple clusters $M_P = \left\{m_1^p, \cdots, m_{N_2}^p\right\}, m_i^p \in \{0,1\}^{w \times h}$ that can contain multiple superpixels.

There is a large amount of redundant masks in both M_S and M_P, which need to be filtered to achieve more precise and efficient multi-layer image decoupling. Typically, when the intersection-over-union (IoU) between $m_i^p \in M_P$ and $m_j^s \in M_S$ is high, m_j^s is prioritized due to its smoother edges and richer semantic information. For each m_i^p, The maximum intersection-over-union (MIoU) between m_i^p and

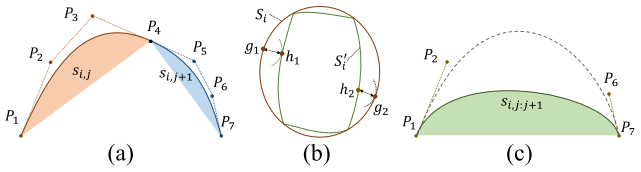


Figure 4. Control point simplification. Given two adjacent Cubic Bézier curves along a path (a), we calculate the chamfer distance between two paths (b). If points P_3, P_4, and P_5 have no critical influence on the path, the control points can be simplified by retaining only points P_1, P_2, P_6, and P_7, resulting in the simplified form shown in (c).

each mask in M_S is defined as:

$$MIoU_{m_i^p} = \underset{m_j^s \in M_S}{max} \frac{MaskArea(m_i^p \odot m_j^s)}{MaskArea(m_i^p \oplus m_j^s)}, \quad (1)$$

where $MaskArea(\cdot)$ is the area of the non-zero region in the mask, $\odot$ and $\oplus$ represent element-wise AND and OR operations, respectively. m_i^p is discarded from M_P when $MIoU_{m_i^p}$ is above a preset threshold. We further sort the mask set $M_S \cup M_P$ in a descending order based on the mask area to obtain a sorted mask set $\{m_1^{sp}, \cdots, m_{N_3}^{sp}\}$. For each mask $m_i^{sp}, 1 \leq i < N_3$, we define the impact factor α_i as:

$$\alpha_i = 1 - \frac{MaskArea(m_i^{sp} \odot m_{i+1:N_3}^{sp})}{MaskArea(m_i^{sp})}, \quad (2)$$
$$m_{i+1:N_3}^{sp} = m_{i+1}^{sp} \oplus m_{i+2}^{sp} \cdots \oplus m_{N_3}^{sp}.$$

m_i^{sp} is discarded when α_i is smaller than a given threshold.

The filtered mask set $M = \{m_1, \cdots, m_N\}$ can effectively implement the human perception hierarchy of the image. As shown in Fig. 3 (d), each mask is assigned an initial color by DiffVG, which is the average RGB value before filling the multi-connected region.

3.2. Control Point Simplification

Traditional image vectorization algorithms are well-suited for efficiently approximating single paths for single connected regions, but often produce redundant control points. To address such an issue, we employ the VTracer [30] to trace and generate initial paths for each mask m_i. As illustrated in Fig. 2, the paths generated for each mask typically contain a substantial amount of redundant control points. To enhance optimization based on the path complexity, we propose an adaptive control point simplification strategy.

Specifically, for each mask m_i corresponding to a path $S_i = \{s_{i,j}\}_{j=1}^{l}$, the path S_i consists of l Bézier curves, where $s_{i,j} = \{p_{i,j,k}\}_{k=1}^{4} = \{(x_{i,j,k}, y_{i,j,k})\}_{k=1}^{4}$ is represented by a Cubic Bézier curve with four control points. Take two adjacent curves $s_{i,j}$and $s_{i,j+1}$ in Fig. 4(a) as an example. To fit a curve between P_1 and P_7 with a Bézier curve as accurately as possible, the following conditions need to be satisfied: (1) There must be no abrupt change between $s_{i,j}$ and $s_{i,j+1}$, i.e., the angle $|180° - \angle P_3P_4P_5| < \delta$. (2) The curve must not self-intersect and must ensure convergence, requiring $\angle P_4P_1P_7 < \angle P_2P_1P_7 < 90°$ and $\angle P_4P_7P_1 < \angle P_6P_7P_1 < 90°$. When both conditions are met, it indicates that P_3, P_4, and P_5 do not have a decisive influence on path S_i. Even if these three points are removed, precise fitting can still be achieved by adjusting the positions of other control points. As shown in Fig. 4(c), only four control points P_1, P_2, P_6, andP_7 are remained. Here, P_1P_2 and P_6P_7 represent the slopes of the simplified curve $s_{i,j:j+1}$ corresponding to $s_{i,j}$ and $s_{i,j+1}$. The simplified path $S_i^{'} = \left\{s_{i,j}^{'}\right\}_{j=1}^{n}$ exhibits considerable divergence from the path S_i, where $s_{i,j}^{'} = \left\{p_{i,j,k}^{'}\right\}_{k=1}^{4} = \left\{(x_{i,j,k}^{'}, y_{i,j,k}^{'})\right\}_{k=1}^{4}$.

Note that directly optimizing parameters based on differentiable rendering may introduce artifacts such as shape distortions [12, 22, 40], especially at intersections of Bézier curves within the same path. To guarantee a more robust optimization process for differentiable rendering, it is important to initially optimize all control points independently, to achieve maximum proximity between the simplified path and the original path while maintaining low computational overhead. Specifically, we perform dense sampling along both S_i and $S_i^{'}$ to obtain point sets $G = \{g_{i,j}(t)\}_{j=1}^{l}$ and $H = \{h_{i,j}(t)\}_{j=1}^{n}$, respectively.

$$g_{i,j}(t) = \mathbf{F}_{p_{i,j,1}p_{i,j,2}p_{i,j,3}p_{i,j,4}}(t), \quad (3)$$

$$h_{i,j}(t) = \mathbf{F}_{p_{i,j,1}^{'}p_{i,j,2}^{'}p_{i,j,3}^{'}p_{i,j,4}^{'}}(t), \quad (4)$$

where $g_{i,j}(t)$ and $h_{i,j}(t)$ are a set of sampled points of Bézier curves with four control points $p_{i,j,1}, p_{i,j,2}, p_{i,j,3}, p_{i,j,4}$ and $p_{i,j,1}^{'}, p_{i,j,2}^{'}, p_{i,j,3}^{'}, p_{i,j,4}^{'}$ conditioned on t. $\mathbf{F}$ is the Cubic Bézier function.

As illustrated in Fig. 4(b), g_1 serves as a sampling point within set G, with its corresponding nearest neighbor in set H designated as h_1. Conversely, h_2, a sampling point in set H, has g_2 as its nearest neighbor in set G. By iteratively minimizing the distance between g_1 and h_1, as well as between g_2 and h_2, also known as the chamfer distance loss [3], the path $S_i^{'}$ can be gradually approximated to the target path S_i:

$$\mathcal{L}_{cd} = \sum_{g \in G} \min_{h \in H} \|g - h\|_2^2 + \sum_{h \in H} \min_{g \in G} \|h - g\|_2^2. \quad (5)$$

3.3. Differentiable Rendering

We utilize a differentiable rendering technique $\mathbf{R}$ to optimize control point parameters S and color parameters C,

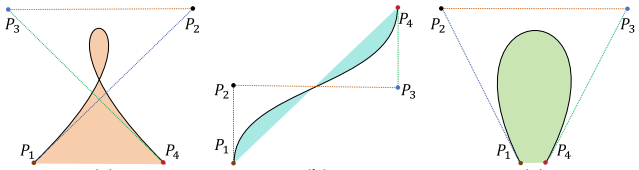


Figure 5. Geometric loss. The four control points P_1, P_2, P_3, and P_4 of the Bézier curve need to avoid the following situations: (a) preventing the line segment P_1P_2 from intersecting with the line segment P_3P_4; (b) ensuring that the directions of $P_1P_2P_3$ and $P_2P_3P_4$ are not opposite; (c) preventing the angles $P_2P_1P_4$ and $P_3P_4P_1$ from forming obtuse angles.

where $\mathbf{R}$ takes S and C as input and outputs the rendered image $\hat{I}$, denoted simply as $\hat{I} = \mathbf{R}(S, C)$. We refine all shape parameters through an iterative process to minimize a predefined loss function that quantifies the discrepancy between the rendered image $\hat{I} \in \mathbb{R}^{w \times h \times 3}$ produced by DiffVG and the input raster image $I \in \mathbb{R}^{w \times h \times 3}$, where w, h, and 3 represent the width, height, and number of channels of the image, respectively.

To ensure visual similarity between the input raster images and generated vector graphics with gradient fills, we adopt MSE $\mathcal{L}_2 = \left\| I - \hat{I} \right\|_2^2$ as the reconstruction loss.

To prevent intersections generated by curved lines, we also adopt the geometric loss from O&R [12], which effectively simplifies shape complexity by imposing a differentiable penalty on the intersection points. As shown in Fig. 5, P_1, P_2, P_3, and P_4 constitute the four key control points of a Cubic Bézier curve. The geometric loss needs to avoid the following three conditions: (a) the line segment P_1P_2 intersects with the line segment P_3P_4; (b) the directions of $P_1P_2P_3$ and $P_2P_3P_4$ are opposite; (c) the angles between $P_2P_1P_4$ and $P_3P_4P_1$ are obtuse. To measure the severity of three distinct scenarios, we consider the orientations $O(P_1, P_2, P_3)$ of points P_1, P_2 and P_3 as follows:

$$O(P_1, P_2, P_3) = S\left((P_{2,y} - P_{1,y}) \cdot (P_{3,x} - P_{2,x}) - (P_{2,x} - P_{1,x}) \cdot (P_{3,y} - P_{2,y})\right), \tag{6}$$

where $S(\cdot)$ represents a sigmoid function. The same orientation for $[P_1, P_2, P_3, P_4]$ can be defined as:

$$f_{\text{orientation}}(P_1, P_2, P_3, P_4) = \text{AND}(O(P_1, P_2, P_3), O(P_2, P_3, P_4)). \tag{7}$$

The intersubsection of P_1P_2 and P_3P_4 occurs when:

$$f_{intersect}(P_1, P_2, P_3, P_4) = AND(XOR(O(P_1, P_2, P_3), O(P_1, P_2, P_4)) XOR(O(P_3, P_4, P_1), O(P_3, P_4, P_2))). \tag{8}$$

We follow the special definitions in O&R [12] to address the non-differentiability of AND and XOR. In addition, the angular loss is defined as:

$$\mathcal{L}_{geom_c} = ReLU(-\frac{P_1P_2 \cdot P_2P_3}{|P_1P_2| \cdot |P_2P_3|}) + ReLU(-\frac{P_2P_3 \cdot P_3P_4}{|P_2P_3| \cdot |P_3P_4|}). \tag{9}$$

The combined geometric loss is:

$$\mathcal{L}_{geometric} = f_{intersect} + f_{orientation} + \mathcal{L}_{geom_c}. \tag{10}$$

In summary, the training loss function $\mathcal{L}$ is defined as:

$$\mathcal{L} = \mathcal{L}_2 + \lambda_{geometric}\mathcal{L}_{geometric}, \tag{11}$$

where $\lambda_{geometric}$ is the geometric loss weight.

4. Experiments

4.1. Experimental Setup

Implementation Details. We use the Adam optimizer with an initial learning rate of 1 for control point optimization, and adjust the learning rates to 0.1 and 0.01 for point parameters and color parameters respectively during the differentiable rendering optimization phase. A total of 100 iterations are used for the training of all parameters. $MIoU$, α_i, δ, $\lambda_{geometric}$ is set to 0.85, 0.1, 8°, and 0.1 respectively.

Dataset. To thoroughly evaluate the effectiveness of our method, we conduct experiments on three public datasets, *i.e.*, Noto Emoji [10], Fluent Emoji [7], and Iconfont [15]. To verify ability of AdaVec to efficiently reconstruct bitmap images with the minimum number of paths, we randomly select 200 images from Noto Emoji and Fluent Emoji. Additionally, to verify that AdaVec can adaptively allocate the number of paths according to image complexity while preserving the details of the output vector images, we also select 100 images with no gradients but complex structures from Iconfont.

Compared Methods. Three existing methods are used for comparison, including O&R [12], SGLIVE [40], and LIVE [22]. LIVE is the first model-free vectorization method based on optimization. O&R and SGLIVE represent the SOTA methods for image vectorization. O&R accelerates the image vectorization process by introducing a vectorization graph initialization algorithm, while SGLIVE enhances the image layering effect through improved image segmentation techniques. By comparing with these methods, we demonstrate that the proposed AdaVec can not only significantly improve the speed of image reconstruction but also achieve multi-layer decomposition of images more effectively.

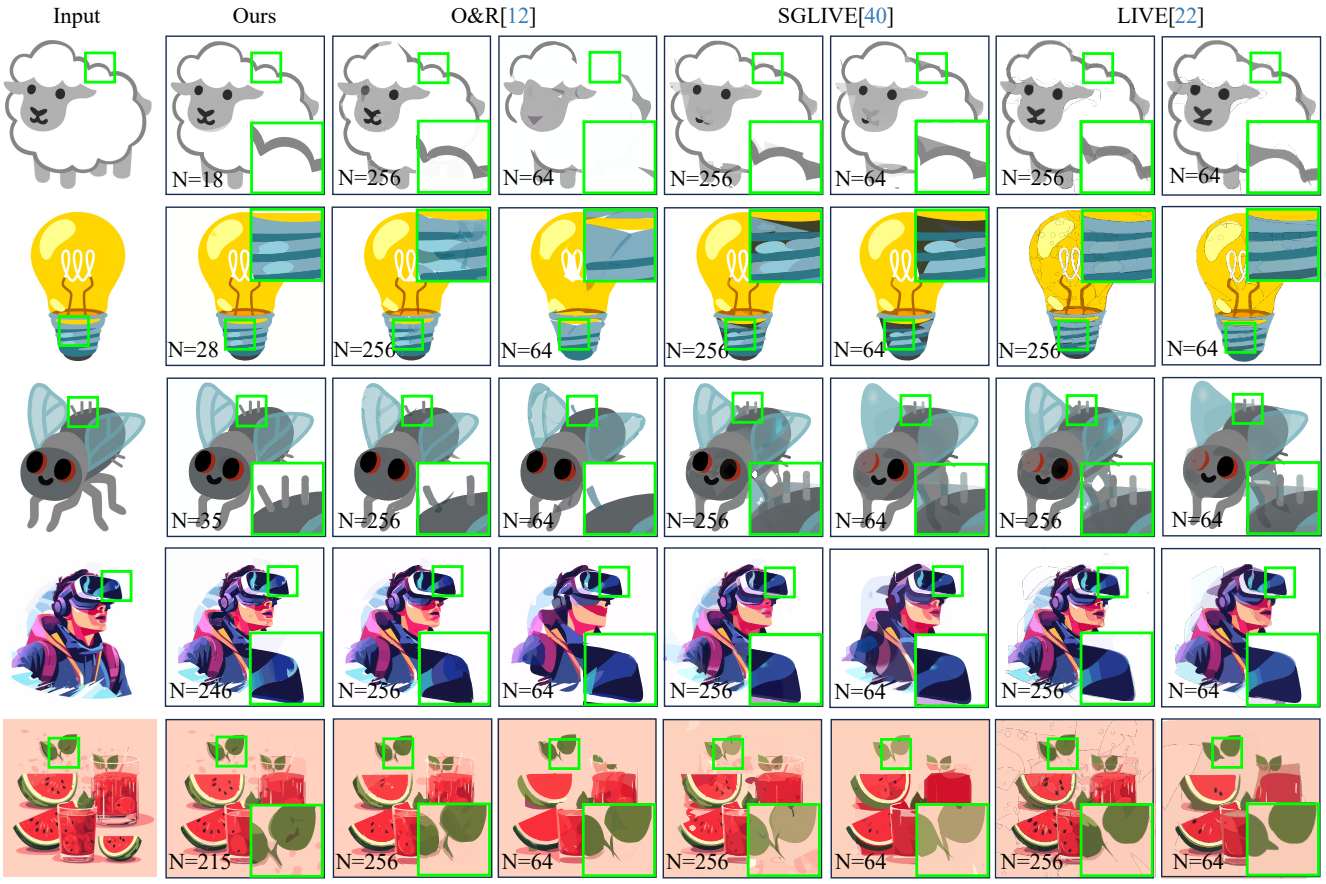


Figure 6. Qualitative reconstruction comparison. We compare the effectiveness of AdaVec against O&R [12], SGLIVE [40], and LIVE [22] in processing images of varying complexity, evaluating by adjusting the number of paths (N) to 256 and 64. The experiment covers simple images (first three rows) and complex images (last two rows). The key regions are enlarged for better visualization.

Evaluation Metrics. We choose seven core evaluation metrics for quantitative comparisons, *i.e.*, (1) Number of Path, which demonstrates the capability of our method to adaptively adopt the minimum number of paths; (2) Number of parameters for control points and color, aiming to illustrate that by simplifying control points, our method can effectively reduce the parameter size of vector graphics; (3) Generation time; (4) Mean Squared Error (MSE); (5) Learned Perceptual Image Patch Similarity (LPIPS)) [37]; (6) Peak Signal-to-Noise Ratio (PSNR) in pixel distance; and (7) Structural Similarity Index Measure (SSIM) [33]. These metrics are all averaged across various datasets.

4.2. Results

Qualitative Evaluation. We conducted a qualitative comparison of the proposed method with compared methods in terms of vector graphic reconstruction quality. As shown in Fig. 6, the path number is fixed at 256 and 64 for the compared methods. However, the strategy of using a fixed number of paths has certain limitations. For simple graphics, excessive paths can lead to redundancy in the first three rows of Fig. 6, while for complex graphics, insufficient paths result in the loss of details in the last two rows of Fig. 6. In contrast, our method can adjust the number of paths dynamically based on the complexity of the image, ensuring that image details are preserved. Consequently, it reconstructs various images with high quality using as few paths as possible. Please refer to supplementary material for additional results.

Quantitative Comparison. As shown in Tab. 1, our method is able to achieve high-quality image reconstruction results with fewer paths and parameters on Noto Emoji and Fluent Emoji, outperforming the benchmark methods even when they use more paths and parameters. For Iconfont with complex scenarios, our method can adaptively select more paths and parameters, efficiently improving image quality while avoiding redundancy. Additionally, in terms of generation time, our method is significantly faster than the benchmark methods. Please refer to the supplementary

Datasets	Methods	Params↓	Time(s)↓	MSE↓	LPIPS↓	PSNR↑	SSIM↑
Noto Emoji [10]	O&R [12] (N=64)	1792	176.3	0.00407	0.0997	23.59	0.928
	O&R [12] (N=256)	7168	269.7	0.00086	0.0278	29.74	0.966
	SGLIVE [40] (N=64)	2432	5051.4	0.00264	0.0632	24.23	0.921
	SGLIVE [40] (N=256)	9728	8721.4	0.00192	0.0488	26.41	0.950
	LIVE [22] (N=64)	1792	4606.9	0.00094	0.0202	29.42	0.923
	LIVE [22] (N=256)	7168	9342.2	0.00036	0.0071	34.53	0.978
	Ours(N=39)	**1220**	**44.9**	**0.00032**	**0.0038**	**35.89**	**0.982**
Fluent Emoji [7]	O&R [12] (N=64)	1792	232.6	0.00070	0.0903	32.06	0.943
	O&R [12] (N=256)	7168	253.5	0.00052	0.0607	33.10	0.960
	SGLIVE [40] (N=64)	2432	1819.8	0.00158	0.1641	28.72	0.936
	SGLIVE [40] (N=256)	9728	7836.9	0.00106	0.1363	29.34	0.954
	LIVE [22] (N=64)	1792	2143.2	0.00107	0.0642	30.81	0.921
	LIVE [22] (N=256)	7168	6783.0	0.00065	**0.0428**	**34.07**	0.965
	Ours(N=26)	**1012**	**50.5**	**0.00047**	0.0449	33.28	**0.972**
Iconfont [15]	O&R [12] (N=64)	1792	253.6	0.00598	0.2044	20.38	0.812
	O&R [12] (N=256)	7168	256.7	0.00174	0.0850	25.13	0.882
	SGLIVE [40] (N=64)	2432	6452.4	0.00357	0.2884	13.54	0.757
	SGLIVE [40] (N=256)	9728	16919.1	0.00146	0.1735	18.48	0.840
	LIVE [22] (N=64)	1792	7239.5	0.00318	0.1465	23.28	0.825
	LIVE [22] (N=256)	7168	19731.9	0.00127	0.0678	25.91	**0.933**
	Ours (N=215)	4598	**164.2**	**0.00102**	**0.0504**	**28.03**	0.925

Table 1. Quantitative comparison of the proposed method with the baselines (*i.e.*, O&R [12], SGLIVE [40], and LIVE [22]). We first evaluate the computational costs by model parameters and running time analysis. We then evaluate their performances using MSE, LPIPS, PSNR, and SSIM scores. The best results are highlighted in bold, and the second one is marked with an underline.

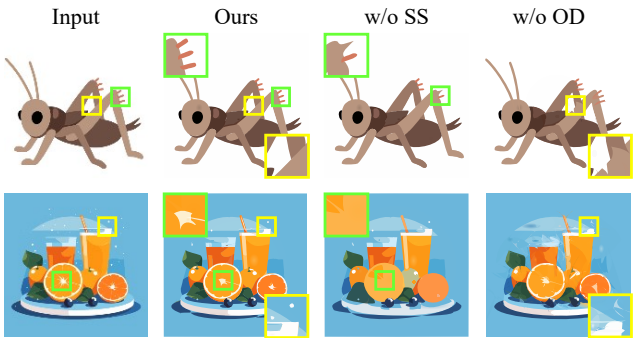


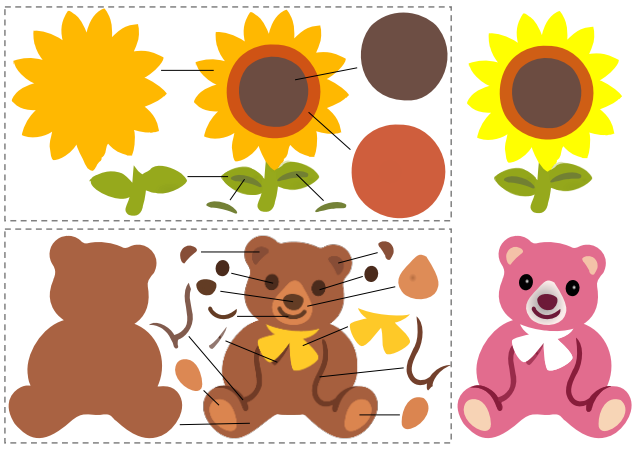
Figure 7. The generation effects of ablation study on Superpixel Segmentation and Optimized Decoupling. The w/o SS refers to applying semantic decomposition without superpixel decomposition. The w/o OD represents no optimized decoupling.

material for additional comparisons.

Ablation Study. To assess the contribution of different components in our proposed method, we perform an ablation study on different stages of our method. Please refer to the supplementary material for additional results and discussions of the ablation study.

SS	OD	MSE↓	LPIPS↓	PSNR↑	SSIM↑
Noto Emoji [10]					
	✓	0.00085	0.0214	29.21	0.885
✓		0.00114	0.0172	28.45	0.892
✓	✓	**0.00032**	**0.0038**	**35.89**	**0.982**
Fluent Emoji [7]					
	✓	0.00152	0.1216	27.58	0.910
✓		0.00068	0.0684	25.23	0.853
✓	✓	**0.00047**	**0.0449**	**33.28**	**0.972**
Iconfont [15]					
	✓	0.00356	0.2102	22.43	0.801
✓		0.00258	0.1403	24.45	0.851
✓	✓	**0.00102**	**0.0504**	**28.03**	**0.925**

Table 2. The results of ablation study on superpixel segmentation and optimized decoupling. The best results are highlighted in bold. The SS stands for superpixel decomposition, and OD represents optimized decoupling.



(a) Image Decomposition (b) Image Editing

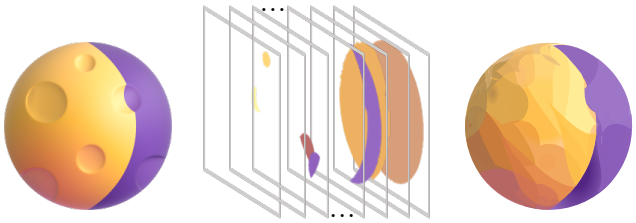
Figure 8. Image editing. Our method can decompose an input raster image into multiple layers (a), and edit one or several of these layers flexibly to obtain various edited images.

We first conduct an ablation study without superpixel decomposition or semantic decomposition in the multi-layer decomposition stage. The w/o SS refers to applying semantic decomposition without superpixel decomposition. As illustrated in the green-boxed area of Fig. 7 and Tab. 2, the detailed elements of the image are effectively preserved by applying superpixel decomposition. The w/o SAM refers to applying superpixel decomposition without semantic decomposition. We can see that w/o SAM leads to higher reconstruction errors and generates more redundant paths and control points, reducing the compactness and representational efficiency of the vectorization results.

To achieve a balance between vector graphic quality and parameter quantity in the control point simplification stage, we then conduct an ablation study on parameter δ, ultimately determining that $\delta = 8°$ is the most suitable value.

In addition, since optimizing all parameters directly using differentiable rendering can lead to artifacts, we first optimize the control point parameters in the control point simplification stage, and optimize both the control point and color parameters in the differentiable rendering stage simultaneously. As illustrated by the yellow-boxed area in Fig. 7 and Tab. 2, such a strategy can reduce the artifacts and improve the generated results effectively.

Image Editing. The image editability highly depends on the layer decomposition result obtained via the vector reconstruction process. We apply our method to the image editing application, and present the visual results in Fig. 8. The results demonstrate that our method can easily achieve efficient editing, including delicate gradient color processing of images, even when faced with extremely complex contours. It proves the flexibility of our method in handling various image editing requirements.



(a) Input (b) Layer Decomposition (c) Output

Figure 9. Failure cases. Our method may effectively perform multi-layer decomposition on gradient-rich images, but the resulting vector graphics are less satisfactory.

To further compare the editability of different vectorization methods, we conduct evaluations on SVGEdit-Bench [23] with 4 editing tasks (Colorize, Move, Resize, and Delete). We recruit 15 participants and record the corresponding editing time of different methods. We further utilize GPT-4o [14] to perform the same task, and let participants give a score. The results demonstrate that our method supports more accurate and faster editability. Please refer to the supplementary material for detailed results.

5. Conclusions

In this paper, we present AdaVec, an efficient image vectorization method that can adjust the number of paths and control points dynamically based on the complexity of the input image. Unlike existing methods that rely on fixed parameters, our method leverages image multi-level decomposition to enhance flexibility and efficiency. By decoupling the optimization process into control point and color parameter optimization, we improve the convergence stability and vectorization quality significantly. Extensive experiments demonstrate that our method excels in preserving image details, ensuring topological integrity, and enabling efficient and flexible editing abilities.

Although our method has achieved certain progress, it still has limitations. Precise reconstruction of extremely complex gradient processes, as shown in Fig. 9, is not available yet. Therefore, one of the potential future works is to explore the dynamic selection process of radial gradients and stop-color quantity.

Acknowledgments: This work was partially supported by Guangdong Basic and Applied Basic Research Foundation (No. 2022A1515110740), National Natural Science Foundation of China (No. 62302356, No. 62172316), Key Research and Development Program of Hebei Province, China (No. 23310302D), and Key Research and Development Program of Shaanxi Province, China (No. 2024GX-ZDCYL-01-11).

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE TPAMI*, 2012. 3

[2] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. Deepsvg: A hierarchical generative network for vector graphics animation. In *NeurIPS*, 2020. 2

[3] Ye Chen, Bingbing Ni, Jinfan Liu, Xiaoyang Huang, and Xuanhong Chen. Towards high-fidelity artistic image vectorization via texture-encapsulated shape parameterization. In *CVPR*, 2024. 4

[4] Wen Dai, Tao Luo, and Jianbing Shen. Automatic image vectorization using superpixels and random walkers. In *CISP*, 2013. 2

[5] James Richard Diebel. *Bayesian Image Vectorization: the probabilistic inversion of vector image rasterization*. Stanford University, 2008. 2

[6] Zheng-Jun Du, Liang-Fu Kang, Jianchao Tan, Yotam Gingold, and Kun Xu. Image vectorization and editing via linear gradient layer decomposition. *ACM TOG*, 2023. 2

[7] Fluent emoji. Fluent emoji, 2023. 5, 7

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996. 3

[9] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM TOG*, 2017. 2

[10] Google. Noto emoji, 2023. 5, 7

[11] Gerben Jan Hettinga, Jose Echevarria, and Jiri Kosinka. Efficient image vectorisation using mesh colours. *Eurographics Association*, 2021. 2

[12] Or Hirschorn, Amir Jevnisek, and Shai Avidan. Optimize & reduce: A top-down approach for image vectorization. In *AAAI*, 2024. 1, 3, 4, 5, 6, 7

[13] Teng Hu, Ran Yi, Baihong Qian, Jiangning Zhang, Paul L Rosin, and Yu-Kun Lai. Supersvg: Superpixel-based scalable vector graphics synthesis. In *CVPR*, 2024. 3

[14] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. 8

[15] Iconfont. Iconfont, 2023. 5, 7

[16] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020. 3

[17] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 3

[18] Yu-Kun Lai, Shi-Min Hu, and Ralph R Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM TOG*, 2009. 2

[19] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM TOG*, 2020. 3

[20] Zicheng Liao, Hugues Hoppe, David Forsyth, and Yizhou Yu. A subdivision-based representation for vector image editing. *TVCG*, 2012. 2

[21] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *ICCV*, 2019. 2

[22] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *CVPR*, 2022. 3, 4, 5, 6, 7

[23] Kunato Nishina and Yusuke Matsui. Svgeditbench: A benchmark dataset for quantitative assessment of llm's svg editing capabilities. *arXiv preprint arXiv:2404.13710*, 2024. 8

[24] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: a vector representation for smooth-shaded images. *ACM TOG*, 2008. 2

[25] Antoine Quint. Scalable vector graphics. *IEEE TMM*, 2003. 1

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 2

[27] Pradyumna Reddy, Paul Guerrero, Matt Fisher, Wilmot Li, and Niloy J Mitra. Discovering pattern structure using differentiable compositing. *ACM TOG*, 2020. 3

[28] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *CVPR*, 2021. 2

[29] Juan A Rodriguez, Shubham Agarwal, Issam H Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. Starvector: Generating scalable vector graphics code from images. *arXiv preprint arXiv:2312.11556*, 2023. 2

[30] Chris Tsang Sanford Pun. vtrace: Raster to vector graphics converter built on top of visioncortex, 2020. 4

[31] Peter Selinger. Potrace: a polygon-based tracing algorithm, 2003. 2

[32] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. *ACM TOG*, 2007. 2

[33] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE TIP*, 2004. 6

[34] Tian Xia, Binbin Liao, and Yizhou Yu. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM TOG*, 2009. 2

[35] Guofu Xie, Xin Sun, Xin Tong, and Derek Nowrouzezahrai. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM TOG*, 2014. 2

[36] Ming Yang, Hongyang Chao, Chi Zhang, Jun Guo, Lu Yuan, and Jian Sun. Effective clipart image vectorization through direct optimization of bezigons. *TVCG*, 2015. 2

[37] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

[38] Shuang Zhao, Frédo Durand, and Changxi Zheng. Inverse diffusion curves using shape optimization. *TVCG*, 2017. 2

[39] Hailing Zhou, Jianmin Zheng, and Lei Wei. Representing images using curvilinear feature driven subdivision surfaces. *IEEE TIP*, 2014. 2

[40] Hengyu Zhou, Hui Zhang, and Bin Wang. Segmentation-guided layer-wise image vectorization with gradient fills. *arXiv preprint arXiv:2408.15741*, 2024. 1, 3, 4, 5, 6, 7

[41] Haikuan Zhu, Juan Cao, Yanyang Xiao, Zhonggui Chen, Zichun Zhong, and Yongjie Jessica Zhang. Tcb-spline-based image vectorization. *ACM TOG*, 2022. 2

[42] Haokun Zhu, Juang Ian Chong, Teng Hu, Ran Yi, Yu-Kun Lai, and Paul L Rosin. Samvg: A multi-stage image vectorization model with the segment-anything model. In *ICASSP*, 2024. 3