SuperSVG: Superpixel-based Scalable Vector Graphics Synthesis

Teng Hu[1], Ran Yi[1]*, Baihong Qian[1], Jiangning Zhang[2], Paul L. Rosin[3], Yu-Kun Lai[3]
[1]Shanghai Jiao Tong University, [2]Youtu Lab, Tencent, [3]Cardiff University
{hu-teng, ranyi, cherry_qbh}@sjtu.edu.cn, vtzhang@tencent.com, {RosinPL, LaiY4}@cardiff.ac.uk

Abstract

SVG (Scalable Vector Graphics) is a widely used graphics format that possesses excellent scalability and editability. Image vectorization, which aims to convert raster images to SVGs, is an important yet challenging problem in computer vision and graphics. Existing image vectorization methods either suffer from low reconstruction accuracy for complex images or require long computation time. To address this issue, we propose SuperSVG, a superpixel-based vectorization model that achieves fast and high-precision image vectorization. Specifically, we decompose the input image into superpixels to help the model focus on areas with similar colors and textures. Then, we propose a two-stage self-training framework, where a coarse-stage model is employed to reconstruct the main structure and a refinement-stage model is used for enriching the details. Moreover, we propose a novel dynamic path warping loss to help the refinement-stage model to inherit knowledge from the coarse-stage model. Extensive qualitative and quantitative experiments demonstrate the superior performance of our method in terms of reconstruction accuracy and inference time compared to state-of-the-art approaches. The code is available in https://github.com/sjtuplayer/SuperSVG.

1. Introduction

Scalable Vector Graphics, commonly known as SVG, is a widely used vector image format that has a wide range of applications and advantages within the domains of web design, graphic design, mobile applications, data visualization, and various other contexts. Compared with raster images that represent content by pixels, SVG describes images by parameterized vectors and benefits from its scalability and editability where it can be resized to any resolution without losing quality and can be easily manipulated by its layer-wise topological information.

Given the superior capabilities of Scalable Vector Graphics (SVG) in image representation and editing, there is much research on the topic of image vectorization, which aims to convert rasterized images into SVG. The existing methods can be categorized into three classes: ***1)*** *Traditional algorithm-based methods* [15, 18, 24, 28, 33, 37], where conventional algorithms are employed to fit images, but they usually suffer from a lower vectorization quality. ***2)*** *Deep-learning-based methods* [9, 10, 12, 19, 21, 22], which parameterize raster images using deep neural networks for reconstruction. They are efficient and can handle the vectorization of simple graphics or characters (*e.g.,* icons and emojis), but struggle to vectorize complex images. ***3)*** *Optimization-based methods* [17, 20, 26, 38], which optimize SVG parameters to fit the target image, yielding relatively superior reconstruction quality. However, these methods entail a substantial amount of time and computational resources, making them impractical for timely processing of large-scale data. In summary, previous image vectorization methods either suffer from low reconstruction quality for complex images, or demand extensive computation time, imposing significant constraints on their practical utility.

To achieve good vectorization quality with high ef-

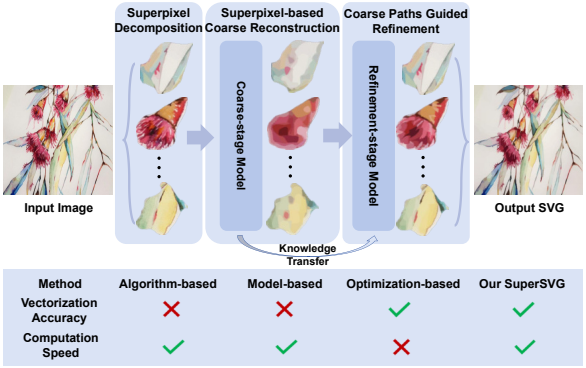

Figure 1. Overview of our SuperSVG: our model first decomposes the image to be vectorized into superpixels, each containing pixels sharing similar colors and contents. The coarse-stage model predicts the path parameters to reconstruct the main structure, and then the coarse paths guided refinement model enriches the details by learning the knowledge from the coarse-stage model. Compared to the previous methods, our SuperSVG achieves both a high vectorization accuracy and fast computation speed.

*Corresponding author.

ficiency, we propose *SuperSVG*, a deep-learning-based method that translates images into scalable vector graphics (SVG) in a coarse-to-fine manner. Since neural networks have difficulties in directly vectorizing complex images, we decompose the input image into different parts in the form of superpixels wherein the pixels share similar colors and textures and then vectorize each part. Then, we propose a *Two-Stage Self-Teaching Training framework* to vectorize the superpixels, where the coarse-stage model is trained to reconstruct the main structure of the image and the refinement-stage model is trained to enrich the image details. We make use of the predicted paths from the coarse-stage model to guide the refinement-stage model in image vectorization. Furthermore, we propose a novel *Dynamic Path Warping loss* which helps the refinement-stage to inherit the knowledge of the coarse-stage model. With the help of the superpixel-based image decomposition and the two-stage self-teaching framework, our SuperSVG can keep the image structure well and reconstruct more details at high speed. Extensive quantitative and qualitative experiments validate the effectiveness of our model.

The main contributions of our work are four-fold:

- We propose SuperSVG, a novel superpixel-based vectorization model that translates the rasterized images into scalable vector graphics (SVG) based on superpixels and vectorizes the superpixels in a coarse-to-fine manner.
- We design a Two-Stage Self-Teaching Training framework, where we employ a coarse-stage model to reconstruct the main structures and a refinement-stage model to enrich the image details based on the coarse-stage output.
- We propose a Coarse Paths Guided Training strategy to guide the refinement-stage model to inherit the knowledge from the coarse-stage model, which greatly improves the performance of the refinement-stage model and avoids converging to suboptimal local minimum.
- We propose Dynamic Path Warping (DPW) loss, which measures the distance between the predicted paths from the refinement-stage model and the pseudo ground truth approximated with coarse paths. By minimizing the DPW loss, the refinement-stage model can distill the knowledge from the coarse-stage model.

2. Related Work

2.1. Image Vectorization

Image vectorization aims to transform a rasterized image into scalable vector graphics (SVG) composed of parameterized vectors. Different from raster images that may become blurry when zooming in, SVG can be rendered at any resolution without losing quality and is convenient to edit, widely used in web design, graphic design, etc. The existing vectorization methods can be classified into 3 categories:

Traditional Algorithm-based Image Vectorization Methods can be classified into mesh-based and curve-based ones. The mesh-based methods [15, 18, 28, 37] segment an input image into non-overlapping patches, and infer the color and the boundary location for each region. The curve-based methods [1, 5, 24, 32, 36] employ Bézier curves with different colors defined on either side to create the vector image. Potrace [24] is a representative method of this type that projects the smooth outlines into Bézier paths, and merge the adjacent paths together. However, the vectorization quality of these methods still needs improvements.

Deep-learning-based Image Vectorization Methods use neural networks to project a raster image into SVG. Im2Vec [21] employs a variational auto-encoder (VAE) [14] to embed the input image and then maps it into path parameters by a Long Short-Term Memory (LSTM) module [23]. Raster2Vec [19] is focused on vectorization of rasterized floor plans using a ResNet [11]. Gao et al. [9] rely on a pre-trained VGG network [25] and a hierarchical Recurrent Neural Network (RNN) to output parametric curves of different sizes. But these methods only focus on simple images and cannot vectorize complex images well. In comparison, our SuperSVG is the first deep-learning-based method that can vectorize images with complex details, thanks to our superpixel decomposition and coarse-path guided refinement that substantially reduce the learning difficulties.

Optimization-based Image Vectorization Methods. DiffVG [17] proposes a differentiable renderer that renders the SVG parameters into images. Based on this, DiffVG minimizes the distance between the rasterized and vector images by optimizing the SVG parameters using gradient descent. LIVE [20] and SAMVG [38] further introduce a layer-wise optimization framework, which achieves better vectorization quality over the previous methods. However, due to the low optimization efficiency, they suffer from a long optimization time. In contrast, our SuperSVG achieves both a good vectorization quality and high efficiency.

2.2. Superpixel Decomposition

Superpixel decomposition is usually used for data preprocessing in vision tasks. Existing superpixel decomposition methods can be categorized into methods based on traditional algorithm or deep learning. For the traditional algorithm based methods, diverse strategies have been employed, *e.g.,* energy-driven sampling [30], geometric flows [16] and clustering [3]. Some recent works [13, 29, 34] employ neural networks to enhance the performance in superpixel decomposition, which shows great potential in this task.

3. Method

Image vectorization aims to translate a rasterized image I into a Scalable Vector Graphic (SVG). An SVG is com-

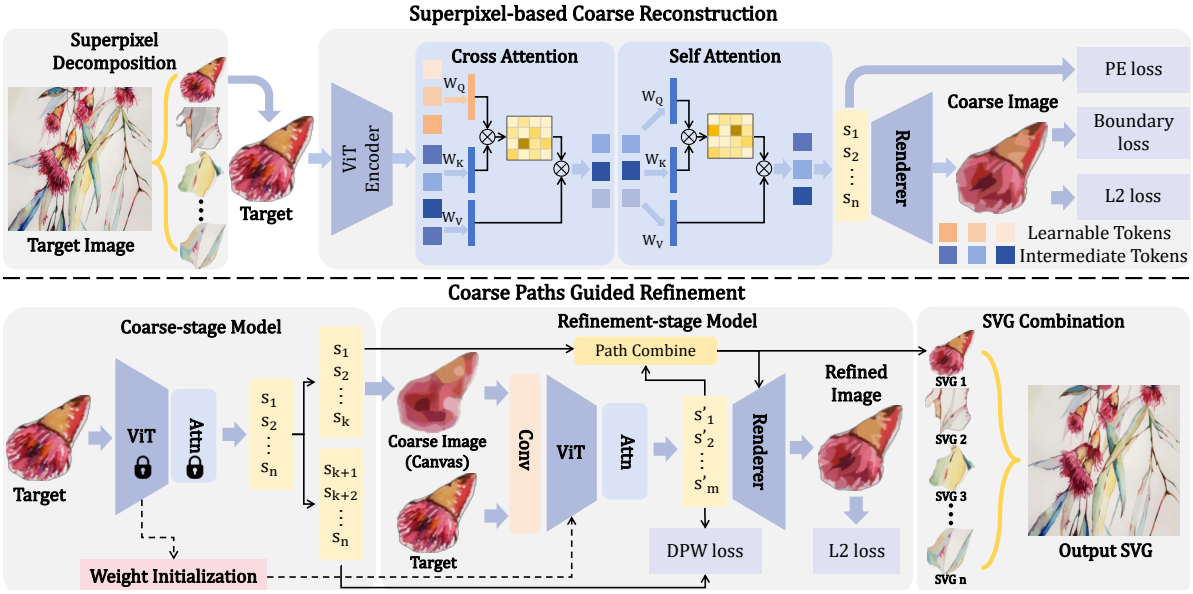

Figure 2. **Main framework of our SuperSVG:** we decompose the target image into superpixels and vectorize each superpixel separately. We employ an attention-based coarse-stage model to predict SVG paths that reconstruct the main structure of the superpixel. Then, a refinement-stage model guided by the coarse paths is designed to predict more SVG paths to refine details based on the coarse image. Finally, by combining all the predicted SVGs for each superpixel, we obtain an output SVG with good structure and fine details.

posed of many vector primitives, which can be SVG paths, ellipses, circles, or rectangles, etc. Following previous works [17, 20, 21], we employ the SVG paths as the shape primitive, where each SVG path defines a region constructed by multiple cubic Bézier curves connected end-to-end with certain color. With the parameters of these SVG paths, the rasterized image can be rendered in any resolution. To obtain the path parameters, some previous methods [17, 20] optimize the path parameters directly to minimize the distance between the input image and the rendered image, which achieves good reconstruction quality but requires long optimization time. To speed up the vectorization process, some deep-learning-based methods [21] employ a deep-learning model to predict the SVG path parameters, but struggle to vectorize complex images.

To achieve good vectorization quality with high efficiency, we propose SuperSVG, a deep-learning-based image vectorization method that translates images into SVG path parameters automatically. To improve the model ability to vectorize complex images, we segment the input image into different parts, within which the pixels share similar colors and textures, and then vectorize each part separately, where superpixels are used for image segmentation as they tend to maintain compactness, uniformity and regularity, particularly suitable for our task. For each superpixel $x \in \mathcal{X}$ where $\mathcal{X}$ is the set of all superpixels, our model converts it into a sequence of path parameters, where each path is composed of several cubic Bézier curves and has a fill color, with a total of N_p parameters. With the predicted path parameters for each superpixel, we employ the differentiable renderer $R(\cdot)$ from DiffVG [17] to get the rendered image $\hat{I}$ in pixel space, which is expected to be close to the input image I.

We propose a two-stage self-teaching framework, composed of a coarse-stage model E_c to reconstruct the main structure and a refinement model E_r to enrich the details, where the predicted paths from coarse-stage model are used to guide the refinement model in vectorization. E_c takes the superpixel x as input and outputs n paths $S = \{s_1, s_2, \cdots s_n\}$ to reconstruct main structure; while E_r takes both the rendered image $R(S)$ and target superpixel x as inputs, and outputs m paths $S' = \{s'_1, s'_2 \cdots s'_m\}$ to refine details. Combining all the predicted S and S' for each superpixel produces the final SVG result.

3.1. Superpixel-based Coarse Reconstruction

Superpixel decomposition. Considering the optimization-based methods suffer from a long optimization time, our SuperSVG builds upon neural networks to efficiently predict SVG paths. However, as neural networks have difficulties in directly vectorizing complex images [8] we therefore simplify the task to vectorizing a certain part of the image containing homogeneous colors and textures. Since superpixel algorithms provide a good tool to decompose images based on local pixel color and also ensure alignment of the regions with the image boundaries, we segment the input image into superpixels, and our model reconstructs each superpixel with scalable vectors. Superpixels also tend to be more regular, making them easier for vectorization. Specifically, we utilize SLIC [3] to decompose the input image

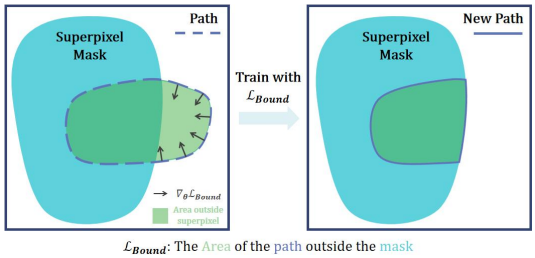

Figure 3. Illustration of our boundary loss $\mathcal{L}_{Bound}$, which computes the area of the SVG paths that are outside the superpixel mask, and guides the paths to be inside the superpixel.

into superpixels. We set the compactness parameter as 30 for SLIC to make superpixels more regular.

Coarse-stage model. For a superpixel x with mask $mask$ (indicating those pixels within x with 1 and 0 otherwise), we first design the coarse-stage model E_c to vectorize the main structure by predicting the SVG path sequence $E_c(x) = S = \{s_1, s_2, \cdots s_n\}$. Inspired by AttnPainter [27], E_c is composed of a Vision Transformer (ViT) encoder [7] and a cross-attention module followed by a self-attention module, which is shown in Fig. 2.

Specifically, the ViT encoder first encodes the input superpixel x into image feature T_f. To control the number of output paths (n) and the parameter number in each path (N_p), we employ a cross-attention module to calculate the correlation between the image feature T_f and n N_p-dimensional learnable path queries T_l, and output an intermediate feature T'_f with the shape of path parameters ($n \times N_p$), which is formulated as:

$$T'_f = Softmax(\frac{(W_Q T_l)(W_K T_f)^T}{\sqrt{d}})(W_V T_f), \quad (1)$$

where W_Q, W_K and W_V are learnable query, key and value matrices.

Then, a self-attention module is employed to process the image feature T'_f and project it into the parameter space with n paths, where each path contains N_p parameters.

Training objectives. *1) Normalized Reconstruction Loss:* We employ the differentiable renderer $R(\cdot)$ in DiffVG [17] to render a raster image $\hat{x} = R(S)$ from the predicted path parameters S. Then, we train the coarse-stage model E_c by minimizing the normalized $\mathcal{L}_2$ distance between the rendering $\hat{x} = R(S)$ and the target image:

$$\mathcal{L}_2 = \|\hat{x} - x\|^2 \cdot \frac{\sum_p mask(p)}{wh}, \quad (2)$$

where w and h are the width and height of the superpixel image x and the superpixel mask $mask$, and $mask(p)$ indicates the mask value (1 or 0) for pixel p.

2) Boundary Loss: To avoid the SVG path from crossing the superpixel boundary, we propose boundary loss to guide the paths to be inside the superpixel. We set the color of all predicted SVG paths to 1 (white) to get a new path sequence S_{binary}. Then, we compute the boundary loss by:

$$\mathcal{L}_{Bound} = \mathop{\mathbb{E}}_{p \sim mask}(R(S_{binary}) \cdot (1 - mask)), \quad (3)$$

Since $(1 - mask)$ is 1 outside the superpixel mask and 0 inside the mask, the loss term calculates the area of the paths that are outside the superpixel. When some SVG paths cross the superpixel boundary, the path area outside is penalized; while when all SVG paths are inside the superpixel, $\mathcal{L}_{Bound}$ reaches 0 (Fig. 3).

3) Path Efficiency Loss: To enable our model to reconstruct the maximum amount of information with the fewest paths, we propose the path efficiency loss $\mathcal{L}_{PE}$. Specifically, for each path i, an additional opacity parameter β_i is predicted. We treat the path as visible if $\beta_i \geq 0.5$, and the loss L_{PE} penalizes the case with more visible paths, *i.e.,* to encourage reconstructing the image with as few paths as possible, calculated as:

$$\mathcal{L}_{PE} = \sum_{i=1}^{n} Sign(\beta_i - 0.5), \quad (4)$$

where $Sign(\cdot)$ is the sign function. Since $Sign(\beta_i)$ is not differentiable, we approximate $\frac{\partial Sign(\beta_i)}{\partial \beta_i}$ by $Sig(\beta_i)(1 - Sig(\beta_i))$, where $Sig(\cdot)$ is the sigmoid function.

The final training objective is formulated as:

$$E_c^* = \arg\min_{E_c} \mathcal{L}_2 + \lambda_{Bound}\mathcal{L}_{Bound} + \lambda_{PE}\mathcal{L}_{PE}. \quad (5)$$

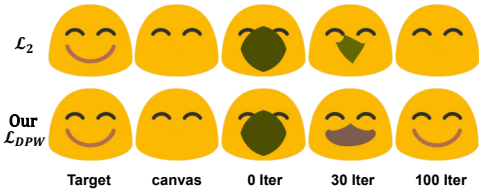

Figure 4. Problem of training the refinement model with $\mathcal{L}_2$ loss alone: optimizing a newly-added path on the canvas by $\mathcal{L}_2$ gradually pulls it to disappear (as a suboptimal local minimum). With our proposed coarse paths guided training and DPW loss, the added path is successfully optimized to resemble the target.

3.2. Coarse Paths Guided Refinement Stage

In the coarse reconstruction stage, our coarse-stage model E_c can output an SVG that captures and reconstructs the main structure of the input superpixel x. The rendered image from the SVG (denoted as c_1) resembles x in general, but lacks some image details, especially when the superpixel is complex. To enrich image details, we employ a refinement model E_r to predict more SVG paths to add more details based on the current canvas c_1.

Model framework. Different from the coarse-stage model E_c that only takes the target superpixel x as input, the refinement model E_r takes both the current canvas c_1 (rendered from the coarse stage output) and the target superpixel

x as inputs, and predict new paths $S' = \{s'_1, s'_2 \cdots s'_m\}$ to be overlaid onto the canvas to refine details. Specifically, 3 convolution layers followed by ReLU activations are employed to fuse the two input images into a feature map. After getting the fused feature map, E_r shares the same structure as the coarse-stage model E_c, which encodes the fused feature map by a ViT Encoder and maps the encoded features into path parameters by a cross-attention and a self-attention layer. To accelerate the training process, we inherit the weights of the ViT encoder in E_c as an initialization.

Local optimal solution with $\mathcal{L}_2$ loss. The goal of the refinement model is to reconstruct more details of the input superpixel based on the current canvas. A simple $\mathcal{L}_2$ loss defined as follows is used as the reconstruction loss:

$$\mathcal{L}_2 = \|x - R([E_c(x), E_r(x, c_1)])\|^2 \cdot \frac{\sum_p mask(p)}{wh}, \quad (6)$$

where the predicted path sequences by coarse-stage model E_c and refinement model E_r are concatenated together to get the final SVG result, and the rendered image of which is expected to resemble the input superpixel x.

However, the refinement model E_r trained with Eq.(6) alone tends to predict paths that are extremely small in area, or even invisible. In Fig. 4, we use an example to illustrate this phenomenon more clearly: we newly add a path onto the canvas and optimize the path parameters with $\mathcal{L}_2$ loss; it can be seen that the new path gradually shrinks and finally disappears in the canvas. A possible reason is that the coarse stage result c_1 is already close to x, and a local optimal solution for E_r is to overlap nothing onto c_1, which is better than adding a sub-optimal path and can prevent the $\mathcal{L}_2$ distance from increasing.

Coarse paths guided training framework. To avoid the refinement model from falling into poor local optimum, we propose a coarse paths guided framework, which inherits the knowledge from the coarse-stage model to help train the refinement model with an additional constraint on the SVG path parameters. As illustrated in Fig. 2, for an input superpixel x, we first use the coarse-stage model to predict a coarse level path sequence $S = \{s_1, s_2 \cdots s_n\}$. Then, we randomly choose a value $k \in (1, n-m)$ and split the predicted path sequence into two subsequences: $S_1 = \{s_1, s_2, \cdots, s_k\}$ and $S_2 = \{s_{k+1}, s_{k+2}, \cdots, s_n\}$. The subsequence S_1 is then rendered into $c_1 = R(S_1)$ and used as the input canvas for the refinement model E_r, while the remaining subsequence S_2 can be regarded as a pseudo ground truth for the output path sequence of E_r. Specifically, when training E_r, in addition to the previous constraints that operate in the pixel space, we add a new constraint on the *path parameter space*, which minimizes the distance between path sequences S' and S_2.

In detail, during training, the refinement model E_r takes both the canvas $c_1 = R(S_1)$ and the target image x as inputs, and outputs the path sequence $S' = \{s'_1, s'_2, \cdots, s'_m\}$.

Algorithm 1 Forward pass to efficiently compute $\mathbf{dpw}_\gamma(\mathbf{S}, \mathbf{S}')$.

Input: S, S', smoothing $\gamma \geq 0$, distance function d

$p_{0,j} = 0; p_{i,0} = q_{i,0} = q_{0,j} = \infty, i \in [\![n]\!], j \in [\![m]\!]$
for $j = 1, ..., m$ **do**
  for $i = 1, ..., n$ **do**
    $p_{i,j} = d_{i,j} + \min^\gamma(q_{i,j-1}, p_{i,j-1})$
    $q_{i,j} = \min^\gamma(q_{i-1,j}, p_{i-1,j})$
  end for
end for

Output: $(\min^\gamma(p_{n,m}, q_{n,m}), P, Q)$

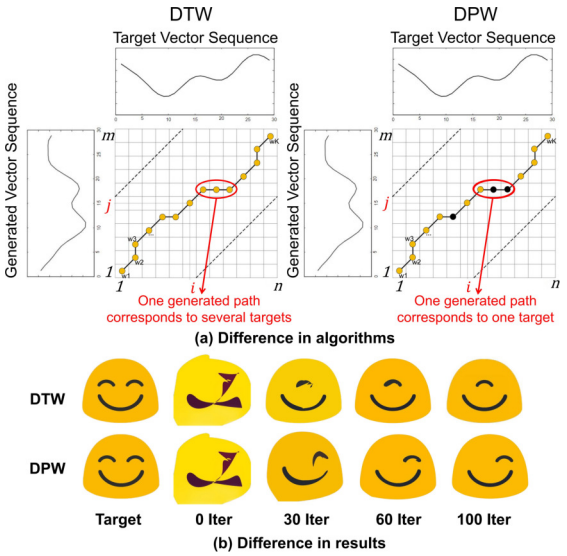

Figure 5. Difference between DTW and our DPW. a) Both the DTW and DPW loss calculate the sum of distances of elements colored yellow. The difference is that one generated path can only match one target path in DPW to avoid averaging several target paths. b) The comparison between the training processes.

We minimize both the distance in pixel space and path parameter space, with the total loss formulated as:

$$E_r^* = \arg\min_{E_r} \mathcal{L}_2 + \lambda_{DPW}\mathcal{L}_{DPW} + \lambda_{Bound}\mathcal{L}_{Bound}, \quad (7)$$

where $\mathcal{L}_{DPW}$ (detailed in Sec. 3.3) is the distance between the path sequences S_2 and S' in the path parameter space.

3.3. Dynamic Path Warping

To calculate the distance between the target path sequence $S = \{s_1, s_2, \cdots s_n\}$ and the predicted path sequence $S' = \{s'_1, s'_2 \cdots s'_m\}$, Dynamic Time Warping (DTW) [4] is a commonly used metric, which finds an optimal matching $match$ between the two ordered sequences (the yellow points in Fig. 5(a)) to minimize the accumulated distance $\sum_i^n \|s_i - s'_{match(i)}\|^2$ where $match(i+1) \geq match(i)$. Note that the monotonicity of the matching function cannot be ignored since the path sequences are well-ordered where latter paths are overlaid on former paths.

In our coarse paths guided framework, we expect the generated path sequence to be a subsequence of the target path sequence. However, in DTW, one generated path s'_j can correspond to several target paths $s_{i1}, ..., s_{il}$. Therefore, when trained with DTW, one generated path tends to become the average of several target paths. As shown in Fig. 5(b), when optimizing 3 paths to match the target emoji (with 4 paths), one path becomes the average of the two eyebrow paths, which is not our desired case.

To address this issue, we propose Dynamic Path Warping (DPW), where each generated path should match one and only one target path, and some target paths can be skipped (to learn a subsequence), as shown in Fig. 5(a), each horizontal line only passes through one matching point (yellow). To compute the DPW, we define $p_{i,j}$ as the minimum accumulated distance when s_i matches s'_j, and $q_{i,j}$ as the minimum accumulated distance when s'_j has been matched to one path before s_i (not including s_i). We employ dynamic programming to compute the final DPW loss $p_{n,m}$ as shown in Alg. 1. For each $p_{i,j}$, the distance $d_{i,j}$ between s_i and s'_j is added to the smaller one of $q_{i,j-1}$ and $p_{i,j-1}$. And for each $q_{i,j}$, its value takes the smaller one between $q_{i-1,j}$ and $p_{i-1,j}$ (more explanations are provided in the supplementary material). Moreover, to make Alg. 1 differentiable, we follow SoftDTW [4] to substitute the $\min(\cdot)$ operation:

$$\min^{\gamma}(a_0, a_1, ..., a_n) = \begin{cases} \min_{i \le n} a_i & \gamma = 0, \\ -\gamma \log \sum_{i=1}^{n} e^{-a_i/\gamma} & \gamma > 0. \end{cases}$$

4. Experiments

4.1. Experiment Setting

Implementation Details. We use SVG paths composed by cubic Bézier curves as the vector primitive, where each SVG path is closed, composed of 4 cubic Bézier curves connected end-to-end and has a fill color. Each SVG path has 28 parameters (24 for shape, 3 for color, and 1 for visibility). The coarse-stage model is trained to predict 128 paths for each superpixel first. Then, the refinement model is trained to predict 8 paths at one time. We train both the coarse-stage model and refinement model on ImageNet dataset [6]. We set batch size as 64 and learning rate as 2.5×10^{-4}. We train the coarse-stage model for 200K iterations with 5K warm up iterations, and train the refinement model for 200K iterations with λ_{DPW} decreasing from 1×10^{-3} to 0 in 10K iterations uniformly. In the following experiments, we implement our model with two versions: **1) SuperSVG-B,** that decomposes the image into superpixels and vectorizes them by the coarse-stage model and refinement model, and **2) SuperSVG-F,** which finetunes the SVG parameters from SuperSVG-B with $\mathcal{L}_2$ loss, which takes around 10 seconds for optimization. All experiments are carried out on an NVIDIA GeForce RTX 4090 24GB GPU.

Evaluation Details. For quantitative evaluation and comparison, we test our model on 1,000 images randomly selected from ImageNet test set, and convert each image into SVGs with 500, 2,000 and 4,000 paths respectively. With the predicted SVG paths, we evaluate the reconstruction accuracy of the output SVG with the following 4 metrics: ***1)*** *MSE Distance* and ***2)*** *PSNR* to measure the pixel distance between the input image and the rendering from SVG; ***3)*** *LPIPS* [35] to evaluate the perceptual distance, and ***4)*** *SSIM* [31] to measure the structural distance. We further compare with Im2Vec [21] on EMOJIS dataset [2].

4.2. Image to SVG Comparison

State-of-the-art methods. The state-of-the-art methods can be classified into 3 categories: 1) Algorithm-based methods: *Potrace* [24] employs edge tracing to vectorize binary images. To process color images, we follow Color Trace[1] to first quantize color images into different layers and then convert each layer to SVG using Potrace. *Adobe Illustrator* [1] is a widely-used commercial software which converts an image into SVG through image tracing. 2) Deep-learning-based methods: *Im2Vec* [21] encodes the target image into latent and predicts the vector paths with LSTM (#suppl.); and 3) Optimization-based methods: *DiffVG* [17] optimizes path parameters with random initialization and *LIVE* [20] employs layer-wise optimization to ensure a good vectorization structure. We use the official codes of these methods and default settings for comparison.

Qualitative Comparison on ImageNet. We compare with the state-of-the-art vectorization methods in reconstruction accuracy on ImageNet. Specifically, we conduct the comparison experiments under path numbers 500, 2,000 and 4,000[2]. The qualitative results are shown in Fig. 6. It can be seen that Potrace cannot reconstruct the image well. LIVE loses a lot of details in relatively smooth areas due to its emphasis on regions with substantial color variations. DiffVG and Adobe work better when the path number increases, but they reconstruct fewer details than our SuperSVG. In comparison, our SuperSVG-B reconstructs most of the details with a short inference time. And by optimizing the SVG parameters from SuperSVG-B with only 10 seconds, our SuperSVG-F achieves the best reconstruction accuracy under different SVG path numbers.

Quantitative Comparison on ImageNet. We further conduct quantitative comparison on 1,000 images randomly sampled from ImageNet [6] dataset (50 images for LIVE due to the extremely long optimization time: each input image takes about 6 GPU hours to optimize under 500 SVG paths). The quantitative results are presented in Tab. 1. Our SuperSVG achieves the best image vectorization results.

[1] https://github.com/HaujetZhao/color-trace

[2] Since Adobe and Potrace outputs have different number of parameters in each path, we keep their output parameter number the same as ours.

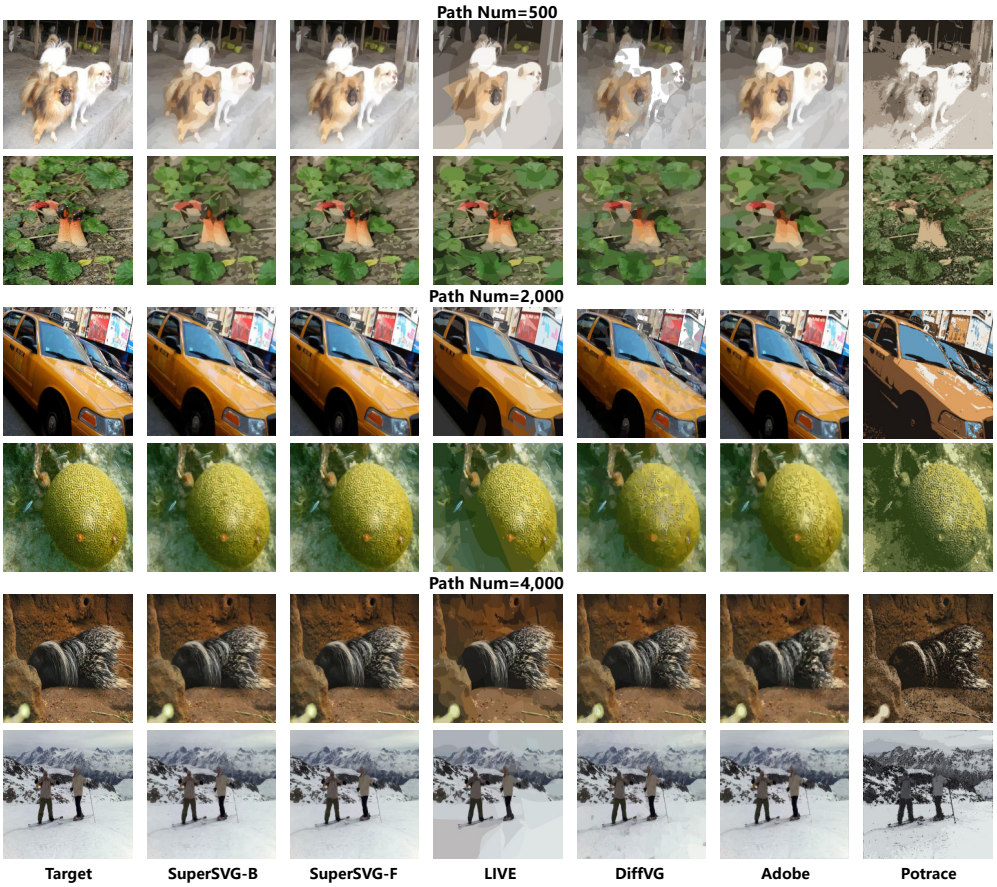

Figure 6. Qualitative comparison with the state-of-the-art methods in image vectorization with different number of SVG paths.

Table 1. Quantitative comparison between the state-of-the-arts and Ours. **Bold** and underline for the best and the second best results.

#Paths	Method	Time (s) ↓	MSE ↓	PSNR ↑	LPIPS ↓	SSIM ↑
500	LIVE [20]	20,000	0.0039	24.10	0.4467	0.7983
	DiffVG [17]	19.29	0.0069	21.42	0.5319	0.6671
	Adobe [1]	0.87	0.0067	21.82	0.5595	0.6939
	Potrace [24]	0.98	0.0208	17.85	0.5115	0.6920
	SuperSVG-B (Ours)	**0.31**	0.0044	24.80	0.4452	0.7687
	SuperSVG-F (Ours)	10.00	**0.0026**	**27.46**	**0.3558**	**0.8383**
2,000	LIVE [20]	120,000	0.0025	26.98	0.3994	0.8431
	DiffVG [17]	73.65	0.0036	25.88	0.4683	0.7710
	Adobe [1]	2.15	0.0033	26.23	0.3961	0.7229
	Potrace [24]	3.10	0.0160	19.65	0.4355	0.6997
	SuperSVG-B (Ours)	**0.71**	0.0024	27.25	0.3648	0.8446
	SuperSVG-F (Ours)	10.00	**0.0017**	**29.12**	**0.2931**	**0.8828**
4,000	LIVE [20]	300,000	0.0024	26.80	0.3981	0.8446
	DiffVG [17]	140.34	0.0025	27.29	0.3858	0.8492
	Adobe [1]	3.12	0.0035	25.93	0.3408	0.7297
	Potrace [24]	5.11	0.0113	20.68	0.4380	0.7374
	SuperSVG-B (Ours)	**1.04**	0.0019	28.45	0.3187	0.8757
	SuperSVG-F (Ours)	10.00	**0.0014**	**29.96**	**0.2496**	**0.9028**

4.3. Ablation Study

Ablation Study on the Superpixel-based Framework. We first validate the effectiveness of the superpixel-based vectorization framework. We train a model that directly predicts the SVG paths for an input image, without superpixel segmentation. Then, we test the model in two ways: **1)** predict the SVG paths for the whole input image and **2)** uniformly divide the input image into 4×4 blocks and vectorize each block separately. We compare our model with these two models with the same number of paths (1,000). The results are shown in Fig. 7 and Tab. 2(a). The model without superpixel segmentation loses many image details. By introducing the block division, the model enriches the details, but the regions near block boundaries are discontinuous (shown in red box), producing unnatural results. In comparison, our superpixel-based SuperSVG-B reconstructs most details and keeps the image structures well.

Ablation Study on the Coarse-stage Model. We then evaluate the effectiveness of the boundary loss $\mathcal{L}_{Bound}$ and the path efficiency loss $\mathcal{L}_{PE}$ in the coarse-stage model. We train 2 ablated models: **1)** the model without $\mathcal{L}_{Bound}$; and **2)** the model without $\mathcal{L}_{PE}$, and compare them with our coarse-stage model under **500** SVG paths. In this comparison, we only compare vectorization using the coarse-stage model, without using the refinement model. The results are shown in Fig. 8 and Tab. 2(b). The model without L_{Bound} predicts some paths that cross superpixel boundaries, re-

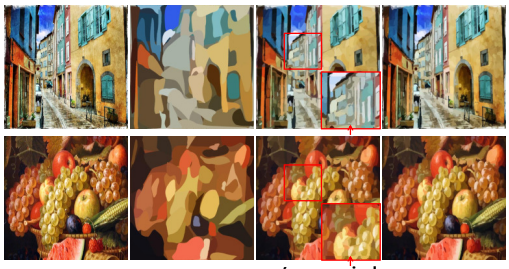

Figure 7. Ablation study on the superpixel-based image vectorization framework. The ablated models either cannot reconstruct most details or suffer from boundary inconsistency problem.

Figure 8. Ablation study on the coarse-stage model. The ablated models either predict paths crossing superpixel boundaries or reconstruct less details than ours.

sulting in worse reconstruction. The model without L_{PE} has a poorer performance, which is validated by the metric results. In comparison, our model outperforms the two ablated models, validating the effectiveness of the losses $\mathcal{L}_{Bound}$ and $\mathcal{L}_{PE}$ in the coarse-stage model.

Ablation Study on the Refinement-stage Model. Finally, we validate the effectiveness of the refinement stage and the DPW loss. We train 3 ablated models: **1)** the model without the refinement stage[3]; **2)** the model without the DPW loss $\mathcal{L}_{DPW}$ (*i.e.,* without coarse paths guidance, with pixel-wise loss only); and **3)** the model replacing DPW loss $\mathcal{L}_{DPW}$ with $\mathcal{L}_2$ loss in path parameter space. The results are shown in Fig. 9 and Tab. 2(c). The model without refinement-stage cannot reconstruct as many details as ours. The ablated model without $\mathcal{L}_{DPW}$ predicts paths with a very small area or even invisible, as explained in Sec.3.2, thus the results look similar to the coarse-stage results. For the ablated model replacing DPW loss $\mathcal{L}_{DPW}$ with $\mathcal{L}_2$ in path parameter space, which enforces one-to-one alignment between two paths' control points, since the constraint is too strict, the loss cannot function well in experiments, and the results look alike the results without $\mathcal{L}_{DPW}$. In comparison, our model outperforms the ablated models, validating the effectiveness of the refinement and DPW loss.

[3]Since the model without refinement does not contain refinement paths, we increase the number of coarse paths to keep the path number consistent.

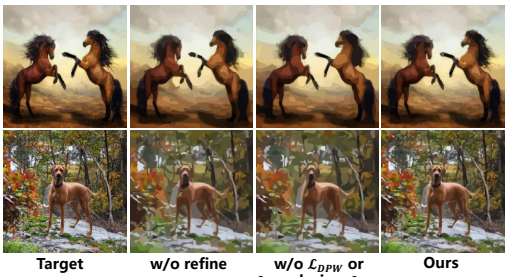

Figure 9. Ablation study on the refinement-stage model. The ablated models either lose details or converge to the poor local optimum described in Sec. 3.2.

Table 2. Quantitative results of ablation studies.

Method	MSE ↓	PSNR ↑	LPIPS ↓	SSIM ↑
w/o superpixel	0.0083	21.72	0.5057	0.6900
4×4 blocks	0.0038	29.50	0.4299	0.7834
Ours	**0.0032**	**26.04**	**0.4075**	**0.8111**

(a) Ablation on superpixel-based framework.

Method	MSE ↓	PSNR ↑	LPIPS ↓	SSIM ↑
w/o $\mathcal{L}_{Bound}$	0.0534	13.38	0.5041	0.6210
w/o $\mathcal{L}_{PE}$	0.0063	22.98	0.4830	0.7219
Ours (Coarse)	**0.0045**	**24.49**	**0.4452**	**0.7673**

(b) Ablation on coarse-stage model (results are obtained by the coarse-stage model alone, without refinement model).

Method	MSE ↓	PSNR ↑	LPIPS ↓	SSIM ↑
w/o refine	0.0041	25.02	0.4375	0.7770
w/o $\mathcal{L}_{DPW}$	0.0045	24.49	0.4452	0.7673
$\mathcal{L}_2$ replacing $\mathcal{L}_{DPW}$	0.0045	24.49	0.4452	0.7673
Ours	**0.0032**	**26.04**	**0.4075**	**0.8111**

(c) Ablation on refinement-stage model.

5. Conclusion

We propose SuperSVG, a novel superpixel-based vectorization model that decomposes a raster image into superpixels and then vectorizes each separately, achieving fast and accurate image vectorization. We propose a two-stage self-teaching framework, where a coarse-stage model reconstructs main structure and a refinement model enriches details, with a novel dynamic path warping loss that guides the refinement model by inheriting knowledge from coarse paths. Extensive experiments demonstrate that SuperSVG achieves the state-of-the-art performance on vectorization.

Acknowledgments

This work was supported by National Natural Science Foundation of China (62302297, 72192821, 62272447), Shanghai Sailing Program (22YF1420300), Young Elite Scientists Sponsorship Program by CAST (2022QNRC001), Beijing Natural Science Foundation (L222117), the Fundamental Research Funds for the Central Universities (YG2023QNB17), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Science and Technology Commission (21511101200).

References

[1] Adobe Illustrator. https://www.adobe.com/products/illustrator.html. 2, 6, 7

[2] Noto Emoji. https://github.com/googlefonts/noto-emoji. 6

[3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 2274–2282, 2012. 2, 3

[4] Marco Cuturi and Mathieu Blondel. Soft-DTW: a differentiable loss function for time-series. In *International conference on machine learning*, pages 894–903. PMLR, 2017. 5, 6

[5] Wen Dai, Tao Luo, and Jianbing Shen. Automatic image vectorization using superpixels and random walkers. In *2013 6th International Congress on Image and Signal Processing (CISP)*, 2013. 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 6

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16×16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 4

[8] Maria Dziuba, Ivan Jarsky, Valeria Efimova, and Andrey Filchenkov. Image vectorization: a review. *arXiv preprint arXiv:2306.06441*, 2023. 3

[9] Jun Gao, Chengcheng Tang, Vignesh Ganapathi-Subramanian, Jiahui Huang, Hao Su, and LeonidasJ. Guibas. DeepSpline: Data-driven reconstruction of parametric curves and surfaces. 2019. 1, 2

[10] David Ha and Douglas Eck. A neural representation of sketch drawings. *Learning,Learning*, 2017. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[12] Teng Hu, Ran Yi, Haokun Zhu, Liang Liu, Jinlong Peng, Yabiao Wang, Chengjie Wang, and Lizhuang Ma. Stroke-based neural painting and stylization with dynamically predicted painting region. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 7470–7480, 2023. 1

[13] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. *arXiv: Computer Vision and Pattern Recognition*, 2018. 2

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[15] Yu-Kun Lai, Shi-Min Hu, and Ralph R. Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transactions on Graphics*, 28(3):1–8, 2009. 1, 2

[16] A. Levinshtein, A. Stere, K.N. Kutulakos, D.J. Fleet, S.J. Dickinson, and K. Siddiqi. TurboPixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 2290–2297, 2009. 2

[17] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15, 2020. 1, 2, 3, 4, 6, 7

[18] Zicheng Liao, H. Hoppe, D. Forsyth, and Yizhou Yu. A subdivision-based representation for vector image editing. *IEEE Transactions on Visualization and Computer Graphics*, 18(11):1858–1867, 2012. 1, 2

[19] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Raster-to-vector: Revisiting floorplan transformation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2

[20] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16314–16323, 2022. 1, 2, 3, 6, 7

[21] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J Mitra. Im2Vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351, 2021. 1, 2, 3, 6

[22] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[23] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. 2

[24] Peter Selinger. Potrace: a polygon-based tracing algorithm, 2003. 1, 2, 6, 7

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[26] Bohao Tang, Teng Hu, Yuzhen Du, Ran Yu, and Lizhuang Ma. Curved-stroke-based neural painting and stylization through thin plate spline interpolation. *Scientia Sinica Informationis*, 54(2):301–315, 2024. 1

[27] Yizhe Tang, Yue Wang, Teng Hu, Xin Tan, and Ran Yi. AttnPainter: A scalable stroke predictor for any scene painting. *arXiv preprint*, 2024. 4

[28] Sebastian Thrun and James Diebel. Bayesian image vectorization: the probabilistic inversion of vector image rasterization. 2008. 1, 2

[29] Wei-Chih Tu, Ming-Yu Liu, Varun Jampani, Deqing Sun, Shao-Yi Chien, Ming-Hsuan Yang, and Jan Kautz. Learning superpixels with segmentation-aware affinity loss. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 568–576, 2018. 2

[30] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin De Capitani, and Luc Van Gool. SEEDS: Superpixels

extracted via energy-driven sampling. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision (ECCV)*, pages 13–26. Springer, 2012. 2

[31] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6

[32] Guofu Xie, Xin Sun, Xin Tong, and Derek Nowrouzezahrai. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Transactions on Graphics*, 33(6):1–11, 2014. 2

[33] Ming Yang, Hongyang Chao, Chi Zhang, Jun Guo, Lu Yuan, and Jian Sun. Effective clipart image vectorization through direct optimization of bezigons. *IEEE Transactions on Visualization and Computer Graphics*, page 1063–1075, 2016. 1

[34] Zipeng Ye, Ran Yi, Minjing Yu, Yong-Jin Liu, and Ying He. Fast computation of content-sensitive superpixels and supervoxels using q-distances. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3770–3779, 2019. 2

[35] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 6

[36] Shuang Zhao, Frédo Durand, and Changxi Zheng. Inverse diffusion curves using shape optimization. *IEEE Transactions on Visualization and Computer Graphics*, 2016. 2

[37] Hailing Zhou, Jianmin Zheng, and Lei Wei. Representing images using curvilinear feature driven subdivision surfaces. *IEEE Transactions on Image Processing*, 23(8):3268–3280, 2014. 1, 2

[38] 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. *arXiv preprint arXiv:2311.05276*, 2023. 1, 2