EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

Image Vectorization via Gradient Reconstruction

Souymodip Chakraborty[1], Vineet Batra[1], Ankit Phogat[1], Vishwas Jain[1], Jaswant Singh Ranawat[1]
Michal Lukáč[1], Matthew Fisher[1] and Kevin Wampler[1]

[1]Adobe Systems

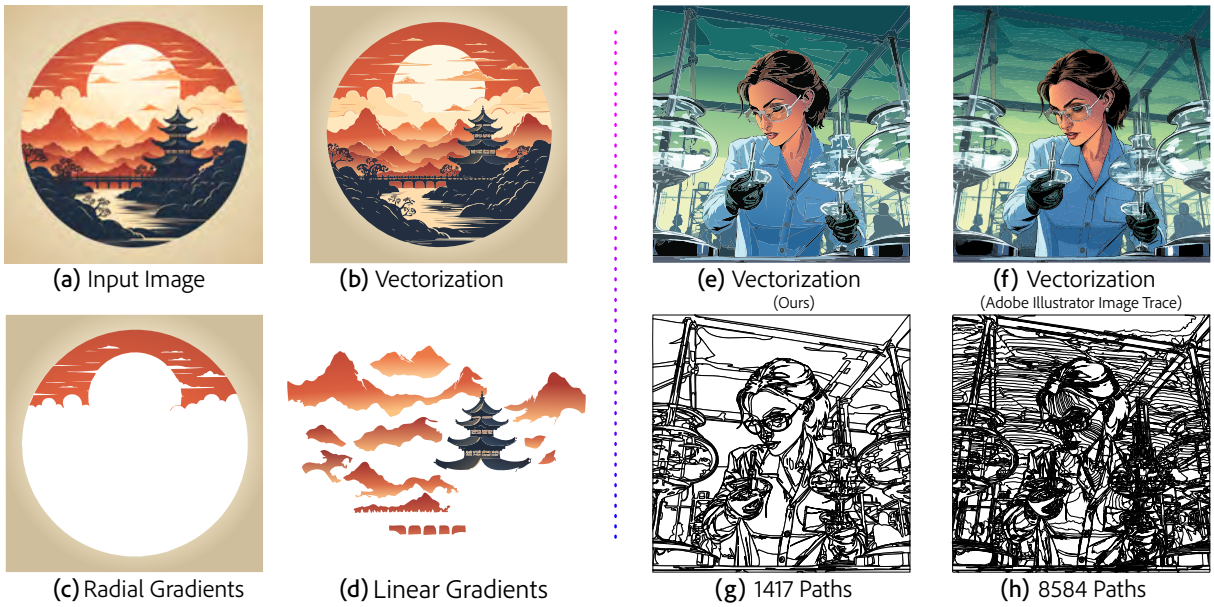


Figure 1: Illustration of our vectorization approach: (a) displays the original input image. Vectorization output (b) with our method using a blend of solid fills, radial (c), and linear (d) gradients, achieving precise reconstructions with simple geometry. (e) and (g) depict vectorization with our method and its corresponding geometry, juxtaposed with vectorization using a popular traditional method (f), (h), which relies exclusively on solid fills, resulting in more complex vector paths and reduced accuracy.

Abstract

As complex color shading becomes increasingly prevalent in contemporary illustration, vectorization techniques must keep up to facilitate vector processing of such images. This paper introduces a method that autonomously identifies regions with smooth color variations, classifies and reconstructs them as linear gradients, radial gradients, or solid fills as appropriate for vectorization, and then determines their geometry. We demonstrate the efficacy, accuracy, and robustness of this method, particularly in managing images characterized by intricate color shading. The outcome is a vectorized representation of the input image that preserves visual accuracy while minimising geometric complexity and improving editability.

CCS Concepts

• ***Computing methodologies*** → *Image processing;*

1. Introduction

Vectorization is a crucial process in graphic design and animation, transforming raster images into vector graphics to provide precision, compactness, and scalability. Unlike raster images, which are composed of a grid of pixels, vector graphics are defined by structured geometry and procedurally colored regions. This makes vector graphics resolution-independent and facilitates straightforward manipulation.

Despite recent advancements in text-to-image technology, which have revolutionized image creation and editing workflows, these

developments primarily focus on raster images. For applications requiring the advantages of vector graphics, vectorization remains the practical bridge for approximating a raster image with a vector graphic.

Effective vectorization requires a balance: the representation must accurately reconstruct the input image while maintaining minimal geometric complexity. Traditional vectorization methods often struggle with this balance, particularly when dealing with images containing intricate shading and color gradients.

Addressing these challenges, we introduce a fully automated solution capable of crafting compact, editable vector representation from a raster image. Our approach stands in contrast to recent solutions [DKT*23], [FLB17] that necessitate additional segmentation maps or explicit user inputs. Our system autonomously segments images into meaningful regions, each reconstructed using an appropriate fill type (solid, linear, or radial gradients). It can handle images of arbitrary size and complexity, and is uniquely equipped to reconstruct gradients with any number of color stops, as well as transformations applied to linear and radial gradient fills. Additionally, our method generates results within a few seconds (see Table 1).

Our method demonstrates robustness through its consistent performance across a diverse range of input images, from simple illustrations to complex photographs (Figure 13, Table 2). It maintains predictable processing times and accuracy regardless of the input complexity, number of colors, or presence of intricate shading. Moreover, our approach operates effectively with a single, fixed set of parameters across all image types, eliminating the need for manual adjustments or tuning based on the input's characteristics.

In summary, our main contributions are:

- A method for automatically segmenting the pixels of a rasterized image into regions with associated fill colors, ensuring that the resulting vectorization has both low reconstruction error and minimal geometry.
- A procedure for efficiently fitting linear and radial gradients to a region with an arbitrary number of gradient stops.

2. Related Work

Algorithmic Image Vectorization Traditional image vectorization research predominantly focuses on representing images with specific parametric primitives, such as cubic Bézier regions [Die08; LL06; Sel03]. Typically, these methods initiate with an image segmentation process to define regions of uniform color [GB05]. However, they often fall short of accurately representing smooth shadings in reconstructions. Diffusion curves [OBW*08; ZDZ17] and gradient meshes [SLWS07] have been used to model smooth shading, but can be difficult to manipulate and have not yet been widely adopted in the graphics design community. Several other methods specialize in vectorizing specific kinds of images such as line drawings [BS19; FLB16; KWÖG18; PNCB21; SBBB20] (see [YVG20] for a recent survey), clip arts [DSG*20], and textures [LPB*13; SWWW16].

Machine-Learning Image Vectorization Recent advancements have seen the development of machine-learning approaches for vectorization, employing diverse models such as autoregressive SVG decoders [LHES19], transformer-based systems [RBCP20], and progressive patch optimization techniques [MZX*22; XZW*23]. Hirschorn et al. [HJA24] propose a top-down vectorization method that balances compactness, reconstruction quality, and runtime efficiency, addressing challenges of editability and optimization. Dziuba et al. [DJEF23] provides an extensive review of these methodologies. Despite being promising, these machine-learning methods struggle with the highly variable encoding rates of vector images, require additional information such as the number of paths and the order of processing them and are compute-intensive. As a result, current state-of-the-art methods are limited to handling at most several hundred regions of uniform color, and often prove too slow for practical applications. For rasterized images generated by diffusion models, which typically require thousands of regions for high-quality reconstruction, these methods fail to scale both in terms of capacity and quality.

Layer Decomposition Several techniques have been devised for decomposing raster images into semi-transparent vector gradient layers [RLB*14; FLB17]. These approaches often necessitate user intervention or additional inputs to achieve effective layer segmentation. More recently, a rule-based method was proposed to streamline the process of determining optimal layer orderings [DKT*23]; however, it also requires a pre-defined segmentation mask as input. These layer decomposition methods focus on ideal images that can be reconstructed with very low error with a small number of layers, such as clip art. In addition to requiring segmentation masks, these methods do not work well on most raster images created by recent text-to-image models because they are not sufficiently robust to small deviations and imperfections in the raster image. Our approach addresses the challenge of segmenting the input image into salient vector regions.

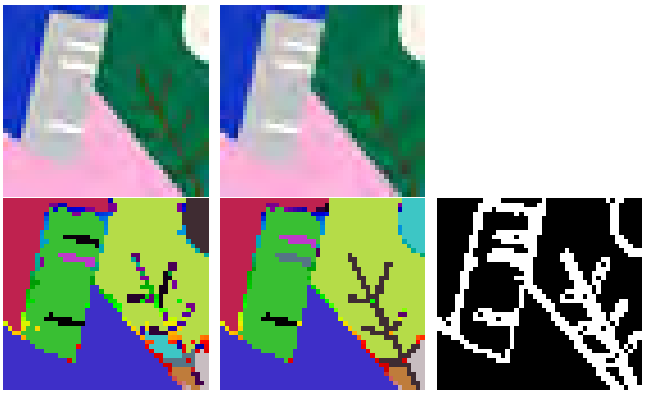
Mesh-Based Vectorization He et al. [HRK24] proposed a sparse patch-based method using Coons patches for feature alignment, while another framework [WCFC24] integrated cubic Bézier curves into curved triangle meshes for feature reconstruction. These approaches rely on mesh-based representations, which can be challenging to edit due to structural complexity. In contrast, our method directly vectorizes raster images into Bézier curves, offering a more intuitive and editable representation.

Segmentation in Sketch Processing Parakkat et al. [PMC22] introduced a Delaunay-based segmentation method for raster sketches requiring color hints, and Scrivener et al. [SCC24] used winding number features for segmenting vector sketches based on pre-existing stroke orientations. While these methods include segmentation and region decomposition, our approach offers a fully automated pipeline to convert raster images into Bézier curves with gradient support, without additional inputs or manual intervention.

3. Method

To vectorize a raster image I, we segment its pixels into connected regions, represented by a 2D segmentation map. We assign a fill function to each segment. Our method consists of the following key steps:

- **Preprocessing:** This step involves noise suppression, edge sharpening, and initial partitioning of the image into smooth regions and discontinuous (edge) regions.



(a) Original image and its segmentation. (b) Smoothened image and its segmentation. (c) Discontinuity $\mathcal{D}$ of the smoothened image.

Figure 2: Applying Mumford-Shah smoothing alleviates antialiasing and resampling artefacts in input images while preserving discontinuities: (a) original input image with its color-difference based segmentation below (184 segments), (b) smoothened image preserving key edges using parameters $\alpha = 1.0, \lambda = 1.5$ and its corresponding segmentation (106 segments), demonstrating how edge-preserving smoothing reduces over-segmentation while maintaining important boundaries, and (c) the resulting discontinuity map $\mathcal{D}$ identifying significant color transitions.

- **Discontinuity-Aware Segmentation:** Creates a segmentation of the smooth image regions which honors the discontinuity.
- **Function Parameter Estimation:** Assigns optimal parameterized functions minimizing L_1 error for each segment.
- **Partition and Merge Discontinuous Pixels:** Segments the pixels in the discontinuous region and merges them with existing partitions for final segmentation.
- **Curve Fitting:** Converts the segmentation into a gapless set of closed curves.

3.1. Preprocessing

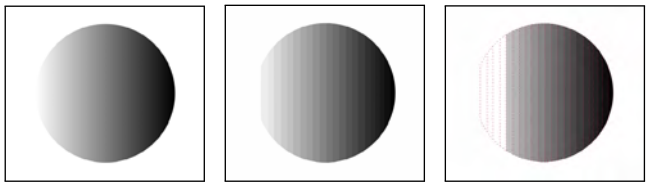
The preprocessing step aims to sharpen edges and reduce noise in the image to facilitate segmentation. We have observed that smoothing based on the Mumford-Shah functional [MS89] provides excellent control over the smooth and discontinuous regions of an image. For this purpose, we employ the discrete Mumford-Shah functional as described in [SC14]. Given an image I, the functional is defined as:

$$u^* \triangleq \underset{u}{\operatorname{argmin}} \sum_{x \in \Omega} \|u(x) - I(x)\|^2 + [\nabla u]_\lambda^\alpha, \quad \text{where } [g]_\lambda^\alpha \triangleq \min(\alpha \|g\|^2, \lambda). \tag{1}$$

The optimal solution, u^*, not only smooths the image but also produces the discontinuity map $\mathcal{D}$, defined as:

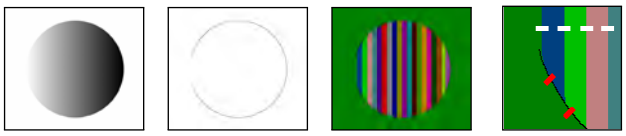
$$\mathcal{D} \triangleq \{x \mid [\nabla u^*(x)]_\lambda^\alpha = \lambda\}. \tag{2}$$

The parameters λ and α control the behavior of the functional. A higher value of λ enforces greater smoothing but risks losing critical discontinuity information, whereas a higher value of α enforces stronger quantization of color. In our experiments, we use conservative values of α and λ (1.0 and 1.5, respectively) that work well across a wide range of images (Figure 2; for detailed analysis, refer to [SC14]).



(a) A raster image with color gradient (b) Color based segmentation. (c) Merged segmentation result.

Figure 3: Challenges in greedy segment merging: (a) original raster image with smooth color gradients, (b) initial color-based segmentation, and (c) result after merging segments based on reconstruction error. Red dotted lines indicate original segment boundaries, demonstrating how naive merging can incorrectly combine segments across important color discontinuities.



(a) Image (b) $\mathcal{D}$ (c) S_0 (d) A crop of S_0.

Figure 4: Color-difference-based segmentation process: (a) input image, (b) discontinuity map $\mathcal{D}$ shown in black, (c) initial segmentation S_0 with distinct colors for each segment, and (d) a zoomed region showing the graph structure where white lines indicate segment adjacency and red lines represent discontinuity segments pairs.

3.2. Discontinuity-Aware Segmentation

We perform color-difference-based segmentation on smooth regions and merge segments. However, a purely greedy clustering approach can yield suboptimal results. As illustrated in Figure 3, merging segments solely by reconstruction error may group regions across discontinuities due to the transitive nature of the operation. To overcome this, we propose a graph-based method called *discontinuity-aware segmentation*, consisting of the following steps:

1. Color-Difference Segmentation of smooth image regions where neighboring pixel color differences remain below a threshold (τ_s).
2. Construction of the weighted graph $\mathcal{G}$ and the discontinuity relation $\mathcal{A}$.
3. Solve the multi-terminal min-cut problem to obtain the final segmentation S.

Color-Difference-Based Segmentation For an image I with a discontinuity map $\mathcal{D}$, let $\bar{\mathcal{D}}$ be the set of pixels not in $\mathcal{D}$, i.e. the set of smooth pixels. We divide the image into $\mathcal{D}$ and $\bar{\mathcal{D}}$ to aid in the segmentation process. These regions correspond to areas of the image with highly varying ($\mathcal{D}$) and smoothly varying ($\bar{\mathcal{D}}$) colors, respectively. We segment the pixels in $\bar{\mathcal{D}}$ based on color differences, computed in the CIELAB color space to effectively capture perceptual variations.

Starting from a segmentation where each pixel is a segment by itself, the algorithm merges neighboring segments if the difference

in their color value is less than the specified threshold τ_s, which yields S_0. See figure 4c.

Weighted Undirected Graph Construction Let the final color-difference-based segmentation be denoted by S_0. Define $\partial(u)$ as the set of edges of pixels in segment u that are shared with pixels from another distinct segment. We construct a graph $\mathcal{G}(V,E,W)$, where the set of vertices V corresponds to the segments in S_0:

$$V := S_0$$

(here, we use "segment" and "vertex" interchangeably). The edge set E consists of unordered pairs of vertices such that two segments u and v are connected if they share a boundary:

$$E = \{(u,v) \mid \partial(u) \cap \partial(v) \neq \emptyset\}.$$

The weight function $W : E \to \mathbb{R}$ is defined as:

$$W(u,v) \triangleq \exp\left(-\|I(u) - I(v)\|_2^2\right),$$

where $I(x)$ represents the mean color of segment $x \subseteq S_0$ (with slight abuse of notation).

We construct the discontinuity relation $A \subseteq V \times V$ by defining a function $f_A : V \times V \to \mathbb{N}$. Consider any pixel p in the discontinuity map $\mathcal{D}$. Define three directions: right ($\hat{r}$), down ($\hat{d}$), and diagonal right-down ($\hat{rd}$). Additionally, let σ represent a small but fixed distance. For each direction $\hat{t} \in \{\hat{r}, \hat{d}, \hat{rd}\}$, we check for the pairs of segments $u, v \in S_0$ whether:

1. There exists a pixel $q \in u$ lying along the direction $\hat{t}$ from p at a distance less than σ.
2. There exists another pixel $q' \in v$ lying in the *opposite* direction along $-\hat{t}$ from p at a distance less than σ.

The function $f_A(u,v)$ is defined as the count of pixels $p \in \mathcal{D}$ for which such a pair of pixels (q,q') exists. The discontinuity relation A is defined as:

$$A \triangleq \{(u,v) \mid f_A(u,v) > \tau_a \cdot \|\partial(u) \cap \partial(v)\|_0\}, \tag{3}$$

where τ_a is a parameter that quantifies the degree of discontinuity between two segments, $\|.\|_0$ is counting norm. See Figure 4d for a visual reference. As highlighted by the issue of arbitrarily merging shown in Figure 3, our goal is to create clusters of segments such that segments across discontinuities, as defined by the set A, are assigned to different clusters. Additionally, we aim to minimize the total number of such clusters.

Multicut Problem The constraint optimization problem is to find the cut set $C \subseteq E$,

$$\begin{aligned} C^* &\triangleq \operatorname*{argmin}_{C \subseteq E} \sum_{e \in C} w(e) \\ \text{s.t.} &\underset{(u,v)\in\mathcal{A}}{\forall}, \; u \text{ and } v \text{ is disconnected in } \mathcal{G}(V, E \setminus C, W) \end{aligned} \tag{4}$$

The corresponding decision problem asks whether such a C with $w(C) \leq k$ exists for a given k. This decision problem is NP-hard, reducible to the *Multi-Terminal Cut problem for planar graphs* [DJP*94].

The problem becomes a min-cut (max flow) problem when the size of $\mathcal{A}$ is 1, solved using the *augmentation graph technique* [CLRS09]. Given the set $\mathcal{A} := \{a_1, \ldots, a_n\}$, we order the pairs based on a heuristic. Then, for each pair in the ordered set $\mathcal{A}$, we progressively call the min-cut algorithm, at most n times, where n is the size of $\mathcal{A}$. The algorithm works in the parameterized complexity $O(n\Phi)$, where Φ is the complexity of the min-cut subroutine, which has a lower bound of $O(m \log\log m)$ [MNNW18], where m is the size of the graph.

Using this approach, we derive a segmentation S for the pixels in the smooth region $\bar{\mathcal{D}}$, where each segment exhibits regular geometry, and similar color regions spanning a discontinuity are assigned to distinct segments. See figure 7a.

3.3. Function Parameter Estimation

An image can be represented as a discrete function mapping the set of pixels Ω to a color space $\mathbb{R}^c$, where $c = 3$ corresponds to the three color channels (RGB). We utilize the following family of parameterized functions:

- **Constant Functions:** These functions assign a constant color value to all pixels within a segment:
$$\mathcal{K}_0 \triangleq \{f_\phi : \Omega \to \{\phi\} \mid \phi \in \mathbb{R}^3\}.$$
- **Linear Functions:** These functions define a linear gradient across a region:
$$\mathcal{K}_1 \triangleq \{f_\phi : \Omega \to \mathbb{R}^3 \mid \phi := (\mathbf{d}, S)\}.$$
Here, $\mathbf{d}$ specifies the direction of the gradient, and S represents a sequence of stops $\{(r_i, c_i)\}_{i \geq 1}$. Each $r_i \in \mathbb{R}$ represents positions ordered linearly along direction $\mathbf{d}$, with $r_{i+1} > r_i$, and $c_i \in \mathbb{R}^3$ is the corresponding color. For a pixel x, if the projection $\mathbf{d}^\top x \in [r_i, r_{i+1})$, the function $f_\phi(x)$ performs a linear interpolation of the colors c_i and c_{i+1}.
- **Radial Functions:** These functions define radial gradients centered around a focal point:
$$\mathcal{K}_2 \triangleq \{f_\phi : \Omega \to \mathbb{R}^3 \mid \phi := (S, \varphi, e, o, T)\}.$$
In this case, S is a sequence of stops as previously defined, φ represents the focal center, e is the eccentricity, o is the center, and T is an affine transform. For a pixel x, if the radius r_x of the circle passing through x satisfies $r_x \in [r_i, r_{i+1})$, the function $f_\phi(x)$ computes the color c by linear interpolation between c_i and c_{i+1}. For further details, refer to [W3C21].

The objective is to approximate the image using a minimal number of parameterized functions from $\mathcal{K}_0$, $\mathcal{K}_1$, and $\mathcal{K}_2$, ensuring that the reconstructed image closely resembles the original while minimizing the reconstruction error.

For each segment s of the image segmentation S obtained thus far, we determine the parameters for the three families of functions ($\mathcal{K}_0$, $\mathcal{K}_1$, and $\mathcal{K}_2$).

3.3.1. Constant Functions

For parameterized constant functions $f_\phi \in \mathcal{K}_0$, the color is the mean of the pixels in the segment. However, due to anti-aliasing, the color of the pixels near the boundary are influenced by neighboring segments. Hence, we use weighted mean, where the weights of the pixels are normalized based on their distances from the boundary.

3.3.2. Linear Functions

To fit a parameterized linear function, we choose the direction $\mathbf{d}$ that best aligns with the color gradient ∇I of pixels in segment s. We first calculate the structure tensor $\mathcal{T}$ for each point $p \in s$,

defined as $\mathcal{T}(p) \triangleq \nabla I(p) \cdot \nabla I(p)^\top$. The direction $\mathbf{d}$ is obtained by maximizing the quadratic form:

$$E_s(\mathbf{d}) \triangleq \mathbf{d}^\top \cdot \sum_{p \in s} \mathcal{T}(p) \cdot \mathbf{d}$$

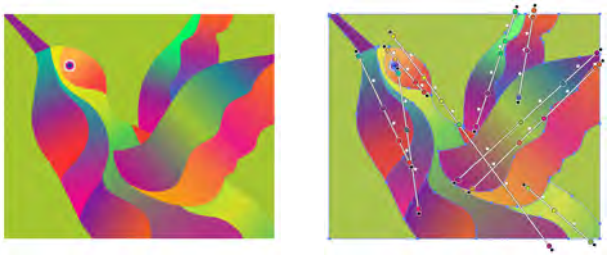
which is the eigenvector corresponding to the largest eigenvalue of $\sum_{p \in s} \mathcal{T}(p)$. Given a direction $\mathbf{d}$, we create a function $\rho : \mathbb{R} \to \mathbb{R}^3$ to capture the color variations along $\mathbf{d}$:

$$\rho(x) = \text{mean}\{I(p) \mid p \in s \text{ and } \mathbf{d}^\top p = x\}$$

The sequence of stops is modeled as a piecewise linear approximation of $\rho(x)$, obtained from the optimal Mumford-Shah 1D functional on the gradient of ρ (Figure 5):

$$u^* \triangleq \text{argmin}_u \sum_{x \in \Omega} \|u - \nabla\rho\|^2 + [\nabla u]_\lambda^\alpha \tag{5}$$

Here, $\nabla\rho$ is the discrete gradient of ρ. The sequence of stop positions $\{r_i\}$ are the discontinuity points of u^*, i.e., $\{x \mid [\nabla u^*(x)]_\lambda^\alpha = \lambda\}$, and the color sequence is $\{\rho(r_i)\}$.



(a) Original raster Image. (b) Vectorized with $\alpha = \infty, \lambda = 1.0$.

Figure 5: Visualization of gradient stop estimation: using parameters $\alpha = \infty$ (enforcing linear interpolation) and $\lambda = 1.0$ (balancing stop count and accuracy), resulting in 41 gradient stops while achieving an average L_1 error of 6.69 per pixel.

3.3.3. Radial Functions

To approximate the colors of a segment s using a radial function $f_\theta \in \mathcal{K}_2$, where $\theta \triangleq (S, \varphi, e, o, T)$, we start with the case when the transform T is the identity and o is the origin. The level sets of f_θ are circles (Figure 6), and the color gradient at a point p is towards the center of that circle:

$$\nabla f_\theta(p) \simeq (p - \varphi_p)$$

Here, φ_p is the center of the level sets (circle) passing through p, defined relative to the focus φ as:

$$\varphi_p \triangleq \varphi + r_p \cdot e$$

The radius r_p is obtained from the root of the quadratic $\|p - (\varphi + r_p \cdot e)\|^2 = r_p^2$. We determine φ and e by minimizing the sum of misalignment between the actual gradient direction and ∇f_θ, which is a function of φ and e:

$$E_s(\varphi, e) \triangleq \sum_{p \in s} (\nabla I(p) \times \nabla f_\theta(p))^2 \tag{6}$$

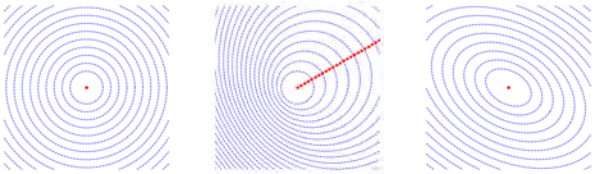
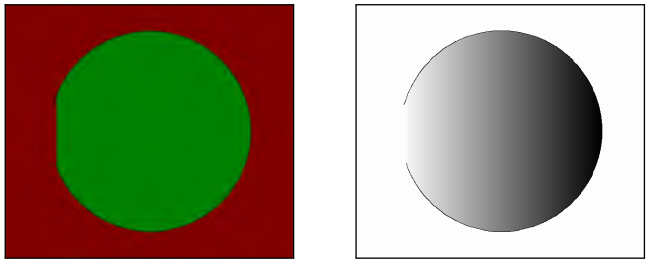


Figure 6: Visualization of radial gradient level sets ($\mathcal{K}_2$ functions): (a) concentric circles with focus at center ($e = 0$), (b) eccentric circles showing non-zero eccentricity e, and (c) transformed level sets under an affine transformation T.



(a) Segmentation S (b) Rendering of Fill function of S

Figure 7: Visualization of our segmentation and fill function assignment: (a) shows the segmentation S of smooth image regions, with each color representing a distinct segment, and (b) displays the reconstruction using our computed fill functions (constant color, linear gradients). Black pixels indicate the discontinuity map $\mathcal{D}$ that separates smooth regions.

where $\times$ denotes the cross product.[†] As with the functions in $\mathcal{K}_1$, we construct the function $\rho : \mathbb{R} \to \mathbb{R}^3$ and use a piecewise linear approximation to determine the color stops:

$$\rho(x) \triangleq \text{mean}\{I(p) \mid p \in s \text{ and } \|p - \varphi_p\|_2 = x\}$$

We use a weighted mean, giving more weight to points in the direction of e to compensate for the fact that the level sets (circles) are closer to each other in the opposite direction of e (see figure 6 middle). When a linear transform T is applied, each gradient vector $(p - \varphi_p)$ is transformed by T. For computation, T can be adjusted such that o is the origin, with inverse transformation applied to the computed values.

3.4. Final Segmentation

Finally, for each segment, we estimate the loss for each function from the three classes and choose the one with the minimum L_1 error, provided the error is below a specified threshold. If none of the function classes approximate the given segment well, we revert to the prior color-based segmentation of this region from S_0 and assign $\mathcal{K}_0$ functions to each sub-part.

Using this approach, we obtain a segmentation $S_{\tilde{\mathcal{D}}}$ of the pixels in the smooth region $\tilde{\mathcal{D}}$, along with a fill function associated with each segment (see figure 7b).

For the pixels in the discontinuity map $\mathcal{D}$, we perform color-difference-based segmentation. This yields a segmentation $S_{\mathcal{D}}$ and an estimated constant fill function $\mathcal{K}_0$ for each segment.

[†] We sum over all three channels, omitted from the equation to avoid clutter.

The segments in $S_{\mathcal{D}}$ primarily consist of anti-aliased pixels and should be merged with adjacent segments. However, in such cases, the color difference between these segments and their neighboring segments often exceeds the color similarity threshold τ_s. Instead, the colors of these segments are better approximated as a linear combination of the fill functions of the neighboring segments.

To address this, consider a segment $u \in S_{\mathcal{D}}$ and its constant fill function c_u, along with its neighbor segments $\{v_1, \dots v_k\}$ in $S_{\bar{\mathcal{D}}}$ and $\{f_1, \dots, f_k\}$ be the neighbor's respective fill functions. For every pixel $p \in u$, let $F_u = \langle f_1(p), \dots, f_k(p) \rangle$ be the evaluation of the fill functions $\{f_1, \dots, f_k\}$. We calculate $d_u(p)$, which is the minimum distance of c_u from the best linear approximation of F_u:

$$d_u(p) \triangleq \min_{\|W\|=1} \|c_u - W^\top F_u\|_2^2,$$

where $W \in \mathbb{R}^k_{\geq 0}$. If the average projection for all pixels in u is below the threshold τ_s, the pixels in u are merged with the nearest neighboring segment (by spatial distance). The final segmentation S_f is the union of $S_{\bar{\mathcal{D}}}$ and the updated $S_{\mathcal{D}}$:

$$S_f \triangleq S_{\bar{\mathcal{D}}} \cup S_{\mathcal{D}}.$$

3.5. Curve Fitting

To generate the output vector graphics, Bézier curves are fitted to the region boundaries obtained from the final segmentation. Initially, each region boundary is represented as a *path*, consisting of a polyline formed by pixel-sized line segments. These paths are then approximated using a combination of line segments and Bézier curves for a compact and accurate representation.

We employ a piecewise Bézier curve fitting algorithm, inspired by the method described in [BLP10]. This algorithm achieves a tight fit, balancing smoothness and fidelity to the original input. The curve fitting process comprises of the following steps:

1. Trace the boundaries of segmented regions to construct a planar straight line graph of pixel-sized line segments separating the different regions in the segmentation.
2. Collect sequences of line segments connected by valence-2 vertices into longer paths, forming a *network of edge paths*. Each such path is either a loop or starts and ends at vertices with a valence which is not 2, and contains only valence-2 vertices in its interior. We call any vertex where two different edge paths meet a *junction node*.
3. Connect pairs of different paths meeting at a junction node to maximize *continuity and smoothness* of the new combined path crossing though the node.
4. Simplify the resulting paths individually using a dynamic programming algorithm, ensuring that the position of any junction nodes remain fixed in this process. This yields a simplified Bézier path for each input polyline path.
5. Combine the simplified paths together into a network of interconnected Bézier curves.

Our method ensures smooth fitting across junctions and efficient performance. For more details, refer to the supplementary material.

4. Implementation Details and Results

Discontinuity aware segmentation requires the initial color-based segmentation and discontinuity pairs. The granularity of segmentation is controlled by the merging threshold τ_s, figure 8 shows

Figure 8: Effect of color-difference based merging with threshold τ_s: (a) $\tau_s = 5$ preserves fine details but yields more segments, (b) $\tau_s = 10$ provides a balanced segmentation (our default value), and (c) $\tau_s = 20$ produces fewer segments but may lose important color transitions. Lower thresholds maintain finer color distinctions while higher values promote segment merging.

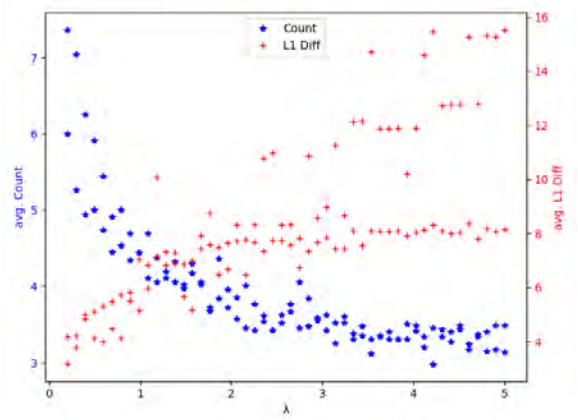


Figure 9: Trade-off between vectorization complexity and accuracy: relationship between the parameter λ (x-axis) and both the average number of gradient stops (left y-axis) and reconstruction error measured by L1 loss (right y-axis). Lower λ values achieve better accuracy at the cost of more gradient stops, with $\lambda = 1.0$ providing an optimal balance.

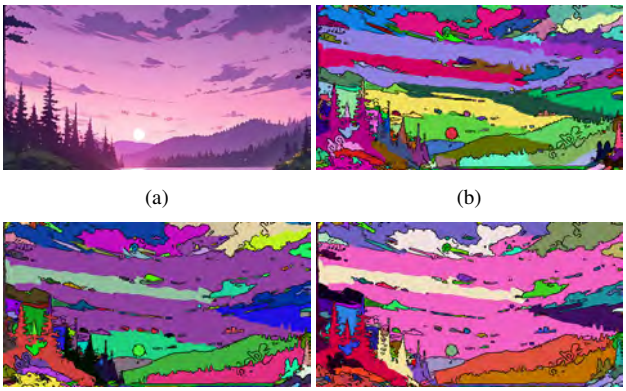
the effect of S_0 for different thresholds. Empirically, we find that $\tau_s = 10.0$ provides an effective balance for typical images. For constructing $f_{\mathcal{A}}$ used in defining the discontinuity pair $\mathcal{A}$, we search for discontinuity pairs within a fixed distance σ, along the specified directions. We set it to 5 pixels, which defines the maximum range for searching discontinuity pairs.

From equation (3), $\mathcal{A}$ is determined by the parameter τ_a, which controls how strictly we add pairs to $\mathcal{A}$. A lower value of τ_a commits more pairs to $\mathcal{A}$, increasing the size of the resulting segmentation. See Figure 10 for reference. We use $\tau_a = 0.25$.

The gradient stops are calculated using (5). We set $\alpha = \infty$ to enforce linear interpolation between stops. As shown in figure 9, lower λ values reduce reconstruction error at the cost of increased stop counts. $\lambda = 1.0$ offers a balanced trade-off between average L_1 error and stop count.

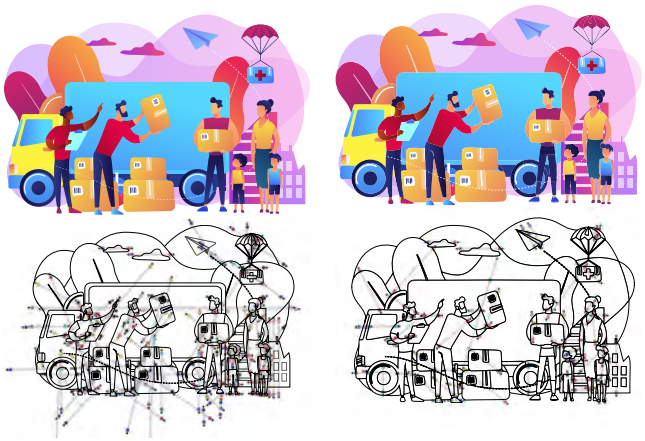
Size	Time (s)	L1 (CIELAB)	SSIM
512x512	0.51	0.03989	0.7247
1024x1024	2.05	0.03532	0.8166
2048x2048	9.71	0.01774	0.9269

Table 1: Average execution time, reconstruction quality using average L1 loss in CIELAB and SSIM, for varying image sizes.



(a) (b) (c) (d)

Figure 10: Effect of discontinuity pair threshold τ_a on segmentation: (a) input image, (b-d) segmentations with $\tau_a = 0, 0.25$, and 0.5 respectively, yielding 363, 334, and 317 segments. Higher τ_a values lead to fewer segments due to fewer discontinuity pairs.



(a) Designer-created vector files with the outline and gradient stops. (b) Our reconstruction with the outline and gradient stops.

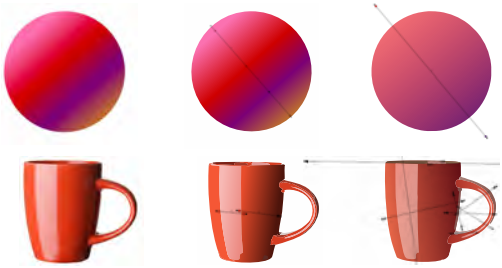
Figure 11: Vectorization of designer-created art with 368 paths and 208 gradient fills. Our method generated 516 paths and 67 gradient fills, achieving an L_1 difference of 0.011 and an SSIM of 0.951.

4.1. Evaluation and Comparisons

We measure *fidelity* by *average* L_1-loss and SSIM, *performance* by evaluating runtime as a function of image size, and *editability* based on the number of paths (which corresponds to the number of regions) and gradients in the output vector. Table 1 shows a summary of our experiments, and a more comprehensive table is provided in the supplementary material.

4.1.1. Ground Truth Vectors

To evaluate the accuracy of our reconstruction technique, we conducted tests on a collection of designer-created vector files, each featuring a significant use of gradients. These vector graphics were rasterized and then processed through our reconstruction method. The results demonstrate high fidelity in reconstruction, with most gradients from the original images accurately recovered, as shown in Figure 11.



(a) (b) (c)

Figure 12: Comparison with layered gradient reconstruction ([DKT*23]): (a) input image, (b) our result, and (c) result using [DKT*23]. Our method achieves more accurate reconstruction without requiring an additional segmentation map (average L_1 loss 0.03209 vs 0.07286 and SSIM 0.886 vs 0.8029).

4.1.2. Generative Workflows

We evaluated our vectorization method on images generated by various generative AI workflows, including text-to-image generation, demonstrating robustness against typical artifacts found in generated images. Furthermore, we applied our method to *generative expand* (outpainting) which extend the boundaries of existing images. The results presented in Figure 14, demonstrate the consistency of our method, as the expanded inputs produce identical outputs for the unchanged regions.

4.1.3. Layered Gradient Reconstruction

The latest advancements in layered gradient reconstruction, notably by [DKT*23], propose a solution for optimizing groupings and layers. However, the method needs a segmentation map, which is a challenging problem in itself. We contrast our results with those from [DKT*23] (see figure 12) and provide a direct comparison using a simplified example. Our comparison reveals that our method achieves more faithful reconstructions without the need for an additional segmentation map.

4.1.4. ARDECO: Automatic Region DEtection and COnversion

[LL06] introduces an automatic segmentation method using an efficient two-level numerical scheme to minimize Mumford and Shah's energy functional to effectively segment images into regions with linear and higher-order gradients.

However, the representation of linear gradients through three independent linear equations, totaling nine coefficients, contrasts with standard vector file formats that use a single gradient direction for all channels. This discrepancy can result in errors when translating the results to standard SVG format. Additionally, translating quadratic gradients is even more challenging, as SVG support is limited to only radial gradients.

The method proposed by [LL06] also leverages salient feature maps to improve segmentation quality. We compared their results that uses salient feature maps, with our method under default parameter settings. As shown in Table 2, our approach demonstrates superior quality and better runtime performance.

4.1.5. Machine Learning Methods

With the advent of machine learning in vectorization, several promising methods have emerged. However, these are still nascent and often struggle with complex artworks requiring the number of

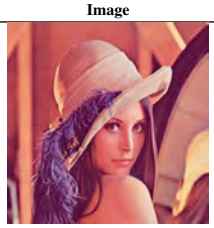
Image	Ours	ARDECO [LL06]	LIVE [MZX*22]	O&R [HJA24]
 (a)	L1: 0.031 SSIM: 0.768 Time: 0.525s Paths: 347	L1: 0.037 SSIM: 0.697 Time: 72-83s Path: 243	L1: 0.032 SSIM: 0.760 Time: 1820s Paths: 256	L1: 0.056 SSIM: 0.635 Time: 266s Paths: 256
 (b)	L1: 0.033 SSIM: 0.830 Time: 0.485s Paths: 309	L1: 0.050 SSIM: 0.730 Time: 72-83s Paths: 688	L1: 0.034 SSIM: 0.821 Time: 1853s Paths: 256	L1: 0.048 SSIM: 0.709 Time: 262s Paths: 256
 (c)	L1: 0.022 SSIM: 0.777 Time: 0.465s Paths: 254	L1: 0.023 SSIM: 0.749 Time: 72-83s Paths: 335	L1: 0.019 SSIM: 0.787 Time: 1853s Paths: 256	L1: 0.029 SSIM: 0.700 Time: 276s Paths: 256
 (d)	L1: 0.027 SSIM: 0.815 Time: 0.495s Paths: 209	L1: 0.029 SSIM: 0.800 Time: 72-83s Paths: 181	L1: 0.022 SSIM: 0.855 Time: 1914s Paths: 256	L1: 0.034 SSIM: 0.773 Time: 268s Paths: 256
 (e)	L1: 0.050 SSIM: 0.641 Time: 0.643s Paths: 426	L1: 0.066 SSIM: 0.483 Time: 72-83s Paths: 349	L1: 0.057 SSIM: 0.576 Time: 1855s Paths: 256	L1: 0.083 SSIM: 0.441 Time: 280s Paths: 256
 (f)	L1: 0.050 SSIM: 0.767 Time: 0.111s Paths: 1927	L1: 0.078 SSIM: 0.467 Time: 72-83s Paths: 2043	L1: 0.066 SSIM: 0.544 Time: 1861s Paths: 256	L1: 0.086 SSIM: 0.378 Time: 262s Paths: 256

Table 2: Quantitative comparison of different vectorization methods. We evaluate reconstruction quality using average L1 loss in CIELAB colorspace and structural similarity (SSIM), while assessing computational efficiency through runtime measurement. For [LL06], results and timings are sourced from the original publication. Results for [MZX*22] and [HJA24] are obtained using the authors' original implementations. The number of paths in the output vector graphics serves as a measure of geometric complexity.

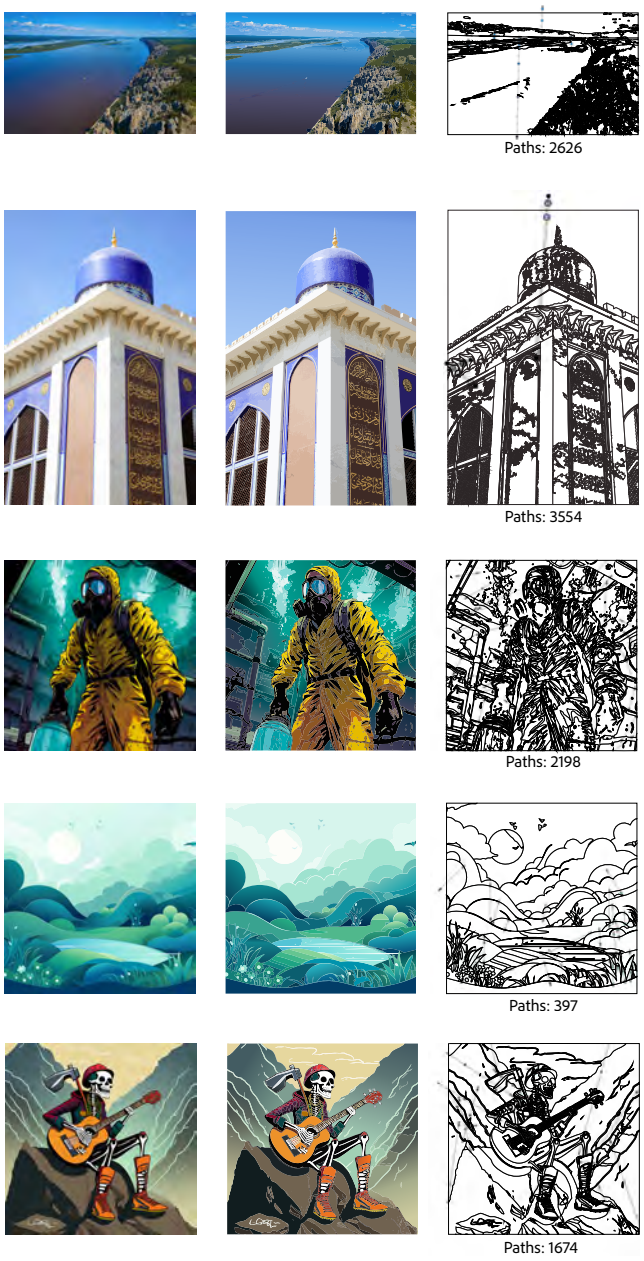


Figure 13: Results demonstrating our method on various input images. For each row: (left) original raster image, (middle) our vector reconstruction, and (right) the paths used in reconstruction.

paths a priori. Our comparison (Table 2) with [MZX*22; HJA24] reveals that our method consistently produces better structured geometry with significantly faster runtime, highlighting its efficacy over current machine learning approaches.

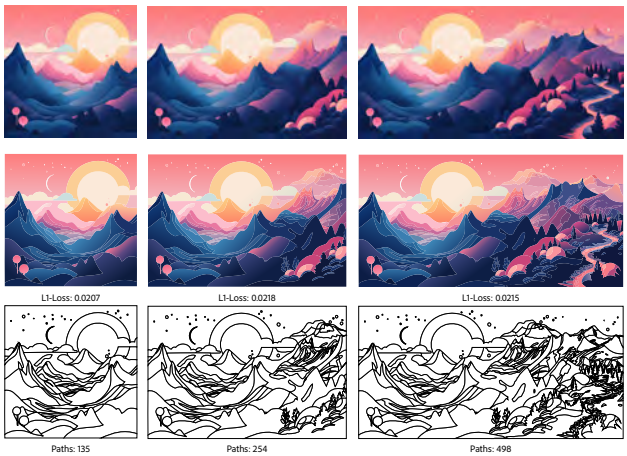


Figure 14: Vectorization for out-painted images (top row), using our method (middle row), and the corresponding outlines (bottom row). Note that expanded inputs produce identical outputs for the unchanged regions.

5. Conclusion and Future Work

Our current approach focuses on vectorizing raster images to SVG-compatible representation using primitives such as solid fills, linear gradients, and radial gradients. While these primitives are effective for many applications, they may be insufficient to capture complex shadings with high accuracy and reasonable geometric complexity. Consequently, our method may not perform satisfactorily for some natural images with intricate shading details.

Additionally, SVG includes numerous primitives that could potentially enhance the quality and usability of vector outputs, but they are beyond the scope of this paper. For instance, incorporating stroked paths and other primitives could improve the user-friendliness and versatility of vector representations. Furthermore, layer-wise decomposition, which could yield better and more usable outputs, is not addressed in our current work.

Despite these limitations, our method efficiently and effectively processes a wide range of images, highlighting its potential and practicality. The rapid advancement and increasing accessibility of image generation technologies underscore the need for vectorization methods that not only preserve the fidelity of the original images but also produce clean, efficient geometric structures. In this paper, we introduced a vectorization method employing both linear and radial gradients with multiple gradient stops, effectively approximating intricate shadings and complex color transitions.

Looking ahead, we aim to expand our work to incorporate higher-order fill functions, such as free-form gradients, which have the potential to capture more intricate coloring nuances. By exploring additional primitives and techniques, and continuously refining our approach, we aspire to enhance the versatility and robustness of vectorization methods, ultimately contributing to the ever-evolving demands of digital art and design in the modern era.

References

[BLP10] BARAN, ILYA, LEHTINEN, JAAKKO, and POPOVIC, JOVAN. "Sketching Clothoid Splines Using Shortest Paths". *Computer Graphics Forum* (2010). ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2009.01635.x 6.

[BS19] BESSMELTSEV, MIKHAIL and SOLOMON, JUSTIN. "Vectorization of Line Drawings via Polyvector Fields". *ACM Trans. Graph.* 38.1 (Jan. 2019). ISSN: 0730-0301. DOI: 10.1145/3202661. URL: https://doi.org/10.1145/3202661 2.

[CLRS09] CORMEN, THOMAS H., LEISERSON, CHARLES E., RIVEST, RONALD L., and STEIN, CLIFFORD. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844 4.

[Die08] DIEBEL, JAMES RICHARD. *Bayesian Image Vectorization: the probabilistic inversion of vector image rasterization*. Stanford University, 2008 2.

[DJEF23] DZIUBA, MARIA, JARSKY, IVAN, EFIMOVA, VALERIA, and FILCHENKOV, ANDREY. "Image Vectorization: a Review". *arXiv preprint arXiv:2306.06441* (2023) 2.

[DJP*94] DAHLHAUS, E., JOHNSON, D. S., PAPADIMITRIOU, C. H., et al. "The Complexity of Multiterminal Cuts". *SIAM Journal on Computing* 23.4 (1994), 864–894. DOI: 10.1137/S0097539792225297. eprint: https://doi.org/10.1137/S0097539792225297. URL: https://doi.org/10.1137/S0097539792225297 4.

[DKT*23] DU, ZHENG-JUN, KANG, LIANG-FU, TAN, JIANCHAO, et al. "Image vectorization and editing via linear gradient layer decomposition". *ACM Transactions on Graphics (TOG)* 42.4 (2023), 1–13 2, 7.

[DSG*20] DOMINICI, EDOARDO ALBERTO, SCHERTLER, NICO, GRIFFIN, JONATHAN, et al. "PolyFit: perception-aligned vectorization of raster clip-art via intermediate polygonal fitting". *ACM Trans. Graph.* 39.4 (Aug. 2020). ISSN: 0730-0301. DOI: 10.1145/3386569.3392401. URL: https://doi.org/10.1145/3386569.3392401 2.

[FLB16] FAVREAU, JEAN-DOMINIQUE, LAFARGE, FLORENT, and BOUSSEAU, ADRIEN. "Fidelity vs. simplicity: a global approach to line drawing vectorization". *ACM Trans. Graph.* 35.4 (July 2016). ISSN: 0730-0301. DOI: 10.1145/2897824.2925946. URL: https://doi.org/10.1145/2897824.2925946 2.

[FLB17] FAVREAU, JEAN-DOMINIQUE, LAFARGE, FLORENT, and BOUSSEAU, ADRIEN. "Photo2clipart: Image abstraction and vectorization using layered linear gradients". *ACM Transactions on Graphics (TOG)* 36.6 (2017), 1–11 2.

[GB05] GAO, SONG and BUI, TIEN D. "Image segmentation and selective smoothing by using Mumford-Shah model". *IEEE Transactions on Image Processing* 14.10 (2005), 1537–1549 2.

[HJA24] HIRSCHORN, OR, JEVNISEK, AMIR, and AVIDAN, SHAI. "Optimize & reduce: a top-down approach for image vectorization". *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN: 978-1-57735-887-9. DOI: 10.1609/aaai.v38i3.27987. URL: https://doi.org/10.1609/aaai.v38i3.27987 2, 8, 9.

[HRK24] HE, K., ROERDINK, J.B.T.M., and KOSINKA, J. "Image vectorization using a sparse patch layout". *Graphical Models* 135 (2024), 101229. ISSN: 1524-0703. DOI: https://doi.org/10.1016/j.gmod.2024.101229. URL: https://www.sciencedirect.com/science/article/pii/S1524070324000171 2.

[KWÖG18] KIM, BYUNGSOO, WANG, OLIVER, ÖZTIRELI, A., and GROSS, MARKUS. "Semantic Segmentation for Line Drawing Vectorization Using Neural Networks". *Computer Graphics Forum* 37 (May 2018), 329–338. DOI: 10.1111/cgf.13365 2.

[LHES19] LOPES, RAPHAEL GONTIJO, HA, DAVID, ECK, DOUGLAS, and SHLENS, JONATHON. "A learned representation for scalable vector graphics". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, 7930–7939 2.

[LL06] LECOT, GREGORY and LEVY, BRUNO. "ARDECO: Automatic region detection and conversion". *17th Eurographics Symposium on Rendering-EGSR'06*. 2006, 349–360 2, 7, 8.

[LPB*13] LOPEZ-MORENO, JORGE, POPOV, STEFAN, BOUSSEAU, ADRIEN, et al. "Depicting Stylized Materials with Vector Shade Trees". *ACM Transactions on Graphics (TOG)* 32 (July 2013). DOI: 10.1145/2461912.2461972 2.

[MNNW18] MOZES, SHAY, NIKOLAEV, KIRILL, NUSSBAUM, YAHAV, and WEIMANN, OREN. "Minimum Cut of Directed Planar Graphs in o(n Log Log n) Time". *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '18. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2018, 477–494. ISBN: 9781611975031 4.

[MS89] MUMFORD, DAVID BRYANT and SHAH, JAYANT. "Optimal approximations by piecewise smooth functions and associated variational problems". *Communications on pure and applied mathematics* (1989) 3.

[MZX*22] MA, XU, ZHOU, YUQIAN, XU, XINGQIAN, et al. "Towards layer-wise image vectorization". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, 16314–16323 2, 8, 9.

[OBW*08] ORZAN, ALEXANDRINA, BOUSSEAU, ADRIEN, WINNEMÖLLER, HOLGER, et al. "Diffusion curves: a vector representation for smooth-shaded images". *ACM Transactions on Graphics (TOG)* 27.3 (2008), 1–8 2.

[PMC22] PARAKKAT, AMAL DEV, MEMARI, POORAN, and CANI, MARIE-PAULE. "Delaunay Painting: Perceptual Image Colouring from Raster Contours with Gaps". *Computer Graphics Forum* 41 (2022). URL: https://api.semanticscholar.org/CorpusID:249026796 2.

[PNCB21] PUHACHOV, IVAN, NEVEU, WILLIAM, CHIEN, EDWARD, and BESSMELTSEV, MIKHAIL. "Keypoint-driven line drawing vectorization via PolyVector flow". *ACM Trans. Graph.* 40.6 (Dec. 2021). ISSN: 0730-0301. DOI: 10.1145/3478513.3480529. URL: https://doi.org/10.1145/3478513.3480529 2.

[RBCP20] RIBEIRO, LEO SAMPAIO FERRAZ, BUI, TU, COLLOMOSSE, JOHN, and PONTI, MOACIR. "Sketchformer: Transformer-based representation for sketched structure". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, 14153–14162 2.

[RLB*14] RICHARDT, CHRISTIAN, LOPEZ-MORENO, JORGE, BOUSSEAU, ADRIEN, et al. "Vectorising bitmaps into semi-transparent gradient layers". *Computer Graphics Forum*. Vol. 33. 4. Wiley Online Library. 2014, 11–19 2.

[SBBB20] STANKO, TIBOR, BESSMELTSEV, MIKHAIL, BOMMES, DAVID, and BOUSSEAU, ADRIEN. "Integer-Grid Sketch Simplification and Vectorization". *Computer Graphics Forum* 39 (Aug. 2020). DOI: 10.1111/cgf.14075 2.

[SC14] STREKALOVSKIY, EVGENY and CREMERS, DANIEL. "Real-Time Minimization of the Piecewise Smooth Mumford-Shah Functional". *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II*. Ed. by FLEET, DAVID J., PAJDLA, TOMÁS, SCHIELE, BERNT, and TUYTELAARS, TINNE. Vol. 8690. Lecture Notes in Computer Science. Springer, 2014, 127–141. DOI: 10.1007/978-3-319-10605-2_9. URL: https://doi.org/10.1007/978-3-319-10605-2%5C_9 3.

[SCC24] SCRIVENER, DANIEL, COLDREN, ELLIS, and CHIEN, EDWARD. "Winding Number Features for Vector Sketch Colorization". *Comput. Graph. Forum* 43 (2024), i–x. URL: https://api.semanticscholar.org/CorpusID:271515118 2.

[Sel03] SELINGER, PETER. *Potrace: a polygon-based tracing algorithm*. 2003 2.

[SLWS07] SUN, JIAN, LIANG, LIN, WEN, FANG, and SHUM, HEUNG-YEUNG. "Image vectorization using optimized gradient meshes". *ACM Transactions on Graphics (TOG)* 26.3 (2007), 11–es 2.

[SWWW16] SONG, YING, WANG, JIAPING, WEI, LI-YI, and WANG, WENCHENG. "Vector Regression Functions for Texture Compression". *ACM Trans. Graph.* 35.1 (Dec. 2016). ISSN: 0730-0301. DOI: 10.1145/2818996. URL: https://doi.org/10.1145/2818996 2.

[W3C21] W3C. *Paint Servers: Solid Colors, Gradients, and Patterns - SVG 2*. https://www.w3.org/TR/SVG2/pservers.html#RadialGradientNotes. Accessed: 2024-05-17. 2021 4.

[WCFC24] WANG, WANYI, CHEN, ZHONGGUI, FANG, LINCONG, and CAO, JUAN. "Curved Image Triangulation Based on Differentiable Rendering". *Computer Graphics Forum* 43 (Oct. 2024). DOI: 10.1111/cgf.15232 2.

[XZW*23] XING, XIMING, ZHOU, HAITAO, WANG, CHUANG, et al. "SVGDreamer: Text Guided SVG Generation with Diffusion Model". *arXiv preprint arXiv:2312.16476* (2023) 2.

[YVG20] YAN, CHUAN, VANDERHAEGHE, DAVID, and GINGOLD, YOTAM. "A benchmark for rough sketch cleanup". *ACM Trans. Graph.* 39.6 (Nov. 2020). ISSN: 0730-0301. DOI: 10.1145/3414685.3417784. URL: https://doi.org/10.1145/3414685.3417784 2.

[ZDZ17] ZHAO, SHUANG, DURAND, FRÉDO, and ZHENG, CHANGXI. "Inverse diffusion curves using shape optimization". *IEEE transactions on visualization and computer graphics* 24.7 (2017), 2153–2166 2.