Layered Image Vectorization via Semantic Simplification

Zhenyu Wang[1] Jianxi Huang[1] Zhida Sun[1] Yuanhao Gong[1] Daniel Cohen-Or[2] Min Lu[1*]
[1] Shenzhen University [2] Tel Aviv University

Abstract

This work presents a progressive image vectorization technique that reconstructs the raster image as layer-wise vectors from semantic-aligned macro structures to finer details. Our approach introduces a new image simplification method leveraging the feature-average effect in the Score Distillation Sampling mechanism, achieving effective visual abstraction from the detailed to coarse. Guided by the sequence of progressive simplified images, we propose a two-stage vectorization process of structural buildup and visual refinement, constructing the vectors in an organized and manageable manner. The resulting vectors are layered and well-aligned with the target image's explicit and implicit semantic structures. Our method demonstrates high performance across a wide range of images. Comparative analysis with existing vectorization methods highlights our technique's superiority in creating vectors with high visual fidelity, and more importantly, achieving higher semantic alignment and more compact layered representation.

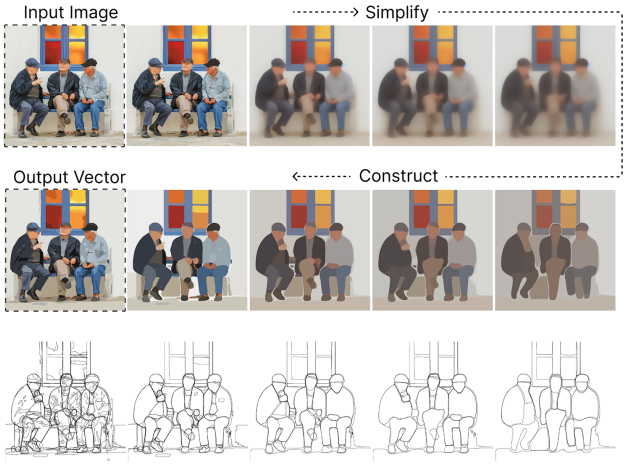

Figure 1. Layered vectorization: by generating a sequence of progressive simplified images (top row), our technique reconstructs vectors layer by layer, from macro to finer details (middle row). Our approach maintains the vectors compactly aligned within the boundaries of explicit and implicit semantic objects (bottom row).

1. Introduction

Vector graphics represent images at the object level rather than the pixel level, enabling them to be scaled without quality loss[16]. This object-based structure allows vectors to efficiently store and transmit complex visual content using minimal data, which is ideal for easy editing and integration across various visual design applications.

Creating effective vector representation, such as SVG, however, often demands considerable artistic effort, as it involves carefully designing shapes and contours to capture an image's essence. Recently, deep-learning-based methods have been introduced to generate vectors, such as from text descriptions [26, 54, 61]. While promising, these methods are constrained by the limitation of pre-trained models, which struggle to produce accurate representation of out-of-domain examples.

An alternative approach is to generate vectors from raster images, a process known as *image vectorization*. Current state-of-the-art vectorization techniques mainly target at visual-faithful reconstruction[20, 35, 67], often produce overly complex and intricate shapes, highlighting the ongoing challenge in achieving a vectorization method that balances visual fidelity with manageability.

In this paper, we introduce a novel image vectorization approach that progressively creates compact layered vector representations, from macro to fine levels of detail. Unlike existing work that takes the input image as the single target [20, 35], our approach features the process of image semantic simplification, i.e., to generate a sequence of progressively simplified versions for the input image (see the top row in Figure 1). This sequence of simplified images serves as stepping-stone targets, guiding the vector reconstruction with incremental and manageable complexity (see the middle row of Figure 1). Notably, our approach enables the effective deduction of vectors that capture the underlying implicit semantic objects from these simplified abstract images, achieving a layered and semantically aligned vector representation that is highly manageable (Figure 2).

The key of our approach is a new image simplification

*Corresponding author, Email: lumin.vis@gmail.com

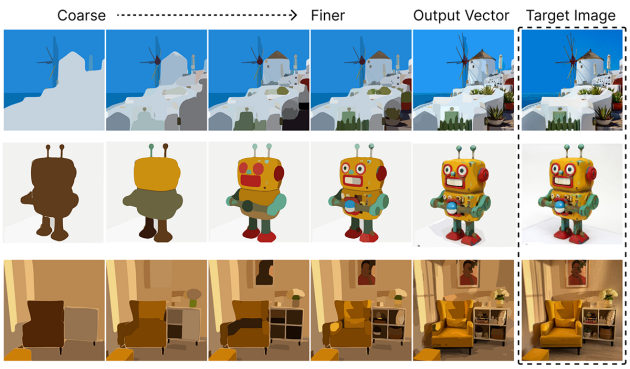

Figure 2. Vectors with levels of detail generated with our method: from left to right, vector primitives from macro to finer details are added layer by layer.

technique leveraging Score Distillation Sampling (SDS) in image generation [38]. By muting the conditioned estimated noise in Classifier-Free Guidance (CFG) [22], our method succeeds in harnessing the *feature-average effect* [19] of SDS for image abstraction, which effectively reduces diverse details while preserving the overall macro structure. Guided by the sequence of progressively simplified images, semantic masks for explicit and implicit objects are detected and layered, based on which vectors are optimized in a two-stage framework. The first stage focuses on building *structure-wise vectors*, optimized towards back-to-front segmented masks by the proposed structure loss. The second stage is to adjust additional vectors to enhance the visual fidelity.

Our method has been tested on a range of vector-style images (e.g., clipart, emojis), and realistic images. Compared to state-of-the-art methods, our approach demonstrates higher visual fidelity, more compact layer-wise representation, and significant improvement in semantic alignment.

2. Related work

Image Vectorization Image vectorization, also known as image tracing, has been a subject of research for decades [12, 51]. Early work relied on segmentating images into non-overlapping 2D patches, such as triangular [3, 7, 49], or other irregular shapes [30]. Later research focused on refining the boundaries of these decompositions, introducing curved boundaries to better capture curvilinear features [34, 55, 59] and gradients [48].

Another approach to image vectorization tackles it as a curve or polygon fitting problem. This includes methods like diffusion curves, which represent images at the extrema of the gradient1 and then render them through the Poisson equation [37, 56, 66]. Other vectorization techniques are particularly suited for non-photorealistic images, such as clip art [8, 11, 15, 23], line drawings [14, 58], cartoons [64], gray-scale manga [47], and pixel art [28].

The advent of deep learning has led researchers to approach vectorization through neural networks. A key enabler is the development of differentiable rasterizers [32], which bridge the vector and raster domains. Im2Vec [41] employs a variational auto-encoder (VAE) on Fonts and Emoji datasets, to map the input image to a latent space and generate a similar vector. ClipGen [43] trains an LSTM on clipart vectors of ten categories to optimize vector primitives. Other network-based vectorization methods focus on line-drawing images [13]. Chen et al. [5] leverage a transformer model to assemble vectorizations from simple primitives. Some approaches avoid model learning. LIVE [35] progressively optimize closed cubic Bézier curves to the large difference regions between the rendered SVG and target image. SAMVG [68] initializes primitives based on segmented masks from the target raster image. SuperSVG [25] trains a model to predict vectors from the superpixel-based segmentation of images. SGLIVE [67] extends the capability of LIVE to support radial gradients via a gradient-aware segmentation. Chen et al. [6] encapsulate texture in the vector optimization. Hirschorn et al. [20] introduce an iterative process that adds primitives based on pixel clustering, and removes primitives with low-ranking scores after optimization.

Our vectorization method is also model-free, leveraging differential rendering. Unlike prior works [20, 35, 67, 68] that optimize towards a single target image, our method generates a sequence of progressively simplified intermediate images as optimization targets. This sequence effectively captures its underlying topology structure from coarse to fine, guiding vectors toward a hierarchical representation of objects and their topologies.

Layer Decomposition Layers are an efficient structure for image manipulation and editing [4]. Other image tasks, such as image matting [45] and image reflection separation [24, 31, 65], stroke decomposition [18, 57], video recoloring [10, 36] are also closely related to this topic.

A range of works decomposes images into layered bitmaps, such as single-color layers with varying opacities [50], layers with soft colors modeled by a normal distribution [2], layers with user-specified colors [29], or color palettes[53, 62, 63]. More recently, deep models have been trained to decompose images into transparent layers [60]. However, those layers are RGBA semi-transparent layers in raster image format.

Another bunch of works decomposes images into vectorized layers. Several work vectorize an image into shapes with linear color gradient [11, 15, 42], based on the *alpha compositing* [39]. Those methods mainly deal with clip art images, which may become prone to errors when applied

to natural images. Other work decomposes an image into sequences of brushstrokes to recover the step-by-step painting process [46, 69]. However, these strokes usually lack semantic object-level representation.

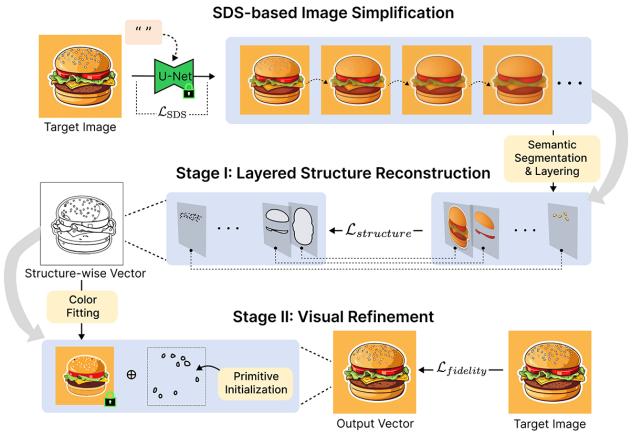

Figure 3. Layered vectorization pipeline: with the input of a target image, its sequence of progressive simplified images is generated using the SDS diffusion model. Vectors are then reconstructed in two stages: structure construction via layer-wise shape optimization to match segmented masks and visual refinement for high fidelity.

3. Overview

Figure 3 shows the pipeline of our method. Taking the target image as input, our process begins with *Progressive Image Simplification* that employs the generative diffusion model based on score distillation sampling [38], to generate a sequence of simplified images, referred to as the *simplified image sequence*. Taking this simplified image sequence as guidance, our method goes through a two-stage vector reconstruction.

The first stage involves constructing structure-wise vectors that capture both implicit and explicit semantic structures. To begin, semantic segmentation [27] is applied to each image in the simplified sequence, to extract pixels of semantic areas, referred to as *masks*, at various levels of detail. Those segmented masks are sorted and organized into back-to-front layers based on their overlap relationships. Using the raw boundaries of segmented masks as structure-wise vectors can be intricate; therefore, optimizing structure-wise vectors is employed. Specifically, structure-wise vectors are initialized per mask and optimized via differential rendering [32] with the *layer-wise structure loss*, which measures the shape alignment between structure-wise vectors and their corresponding masks, to ensure the vector representation accurately maintains semantic structures.

The second stage involves visual refinement. Color fitting is first performed on the structure-wise vectors, serving as the visual basis and frozen. Then taking the rasterized image of frozen structure-wise vectors, we compute its visual differences to the target image. Then visual-wise vector primitives are initialized to regions with large visual differences and optimized toward the target image to minimize the *visual fidelity loss*. During optimization, vector clean-up operations—such as merging and removing redundant vectors—are periodically performed to maintain a neat and efficient vector representation.

4. Progressive Image Simplification

The premise of our vectorization method is to use a sequence of progressively simplified images to guide the optimization. Existing vectorization methods mainly rely on pixel-level analysis of a single target image to decide where to add and optimize vectors, such as large connected areas identified in the target image in LIVE [35] or clusters via DBSCAN in Optimize & Reduce (O&R) method [20]. In contrast, our approach takes the series of simplified images as intermediate targets to optimize.

Figure 4(a) shows an example of the image sequence simplified using our SDS-based method. As shown, the image sequence exhibits varying levels of simplicity, from the original one with many intricate details and textures to a simplified and overall outline on the right.

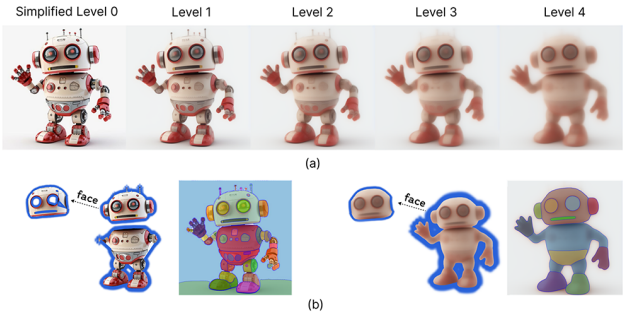

Figure 4. Example of SDS-based image simplification: (a) a sequence of progressively simplified images, with the original image (level 0) on the left; (b) comparison of segmented masks between levels 0 and 4, showing that more simplified image captures more macro structures, such as the 'whole body of the robot' detected in level 4 but not in level 0.

The intuition behind using simplified images to guide layered vectorization is to prioritize the capture of the overall structure before addressing subtle changes. This approach offers two notable benefits. Firstly, by decomposing the vectorization process into manageable levels, it becomes more tangible and achievable compared to optimizing it as a whole. The incremental improvement from one abstract level to the next is relatively small, allowing for effective optimization at each step. Secondly, the pro-

gressive abstraction establishes a macro-to-fine hierarchy, enabling holistic optimization of delicate paths within the image. This approach becomes particularly advantageous when dealing with shapes that exhibit pixel variations due to occlusion, shadows, or textures yet remain integral components of a larger entity. For example, in Figure 4(b), the 'robot' is abstracted as a unit shape at the most simplified level, while segmentation of the original target image fails to capture the entire robot shape. Also, its 'face' can be represented as a unit shape without holes when the simplified image is applied.

4.1. Feature-average Effect in SDS

Our image simplification method takes advantage of the *feature-averaged effect* in SDS [19, 33]. The feature-average effect of SDS can be explained with the gradient of SDS loss, as follows:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\theta) \approx \mathbb{E}_{t,\epsilon,c}\left[\omega(t)\underbrace{(\epsilon_\phi(\mathbf{x}_t,t,y)-\epsilon)}_{\text{SDS update direction}}\frac{\partial g(\theta,c)}{\partial\theta}\right], \tag{1}$$

where g is a differentiable generator that maps optimization variables θ to an image $x = g(\theta)$. For instance, in the original DreamFusion framework [38], g is a NeRF volume renderer. In our case, $g(\theta) = \theta$, meaning the optimization variables θ are the image pixels themselves. The term $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ is the random noise added at timestep t, while $\epsilon_\phi(\mathbf{x}_t, t, y)$ is predicted noise with given condition y. Consequently, $(\epsilon_\phi(\mathbf{x}_t, t, y) - \epsilon)$ implies the update direction in the Denoising Diffusion Probabilistic Model (DDPM) [22].

It has been observed that the pre-trained DDPM is sensitive to the input, often predicting feature-inconsistent noise, even when conditioning input y remains the same. This causes image pixels θ to be updated in inconsistent directions, leading to a feature-averaged result. Figure 5 illustrates the effect using the prompt "a photo of a robot". As the number of optimization steps increases, the images progressively lose detailed features, resulting in a more feature-averaged appearance.

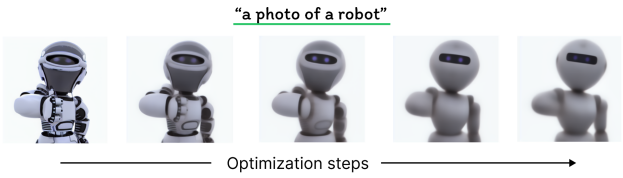

Figure 5. Example of the feature-averaging effect in SDS: as optimization progresses, the 'robot' loses fine details, such as 'fingers' and 'helmet', showing the smoothed and simplified appearance.

4.2. SDS-based Simplification

It is crucial to manage the feature-average effect to ensure progressive simplification without significant shape distortion. For instance, as shown in Figure 5, while simplification is achieved, it leads to noticeable shape alterations.

In text conditional diffusion models, Classifier-Free Guidance (CFG) [21] is introduced to combine two predicted noises into one:

$$\epsilon_\phi^\omega(\mathbf{z}_t, y, t) = (1+\omega)\epsilon_\phi(\mathbf{z}_t, y, t) - \omega\epsilon_\phi(\mathbf{z}_t, t), \tag{2}$$

where $\epsilon_\phi(\mathbf{z}_t, y, t)$ is the noise predicted with the conditioned text input and $\epsilon_\phi(\mathbf{z}_t, t)$ is that with unconditioned input. The CFG scale controls how closely the conditional prompt should be followed during sampling in DDPM.

Inspired by CFG, we found that simplification without importing significant shape distortion can be achieved by increasing the portion of unconditional noise. Two possible approaches can be used to achieve this. As shown in Figure 6, one approach is to set the CFG scale to zero. The other approach is to set the conditioned text prompt to empty (i.e., " "). Both alternatives are effective; we use the latter one in this work.

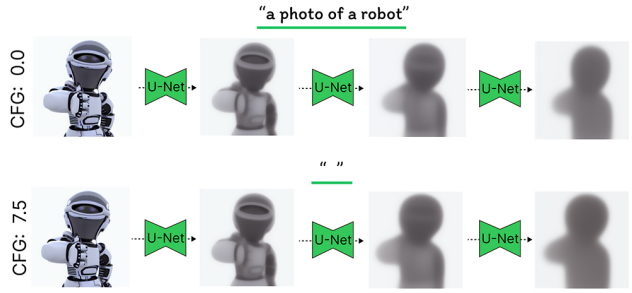

Figure 6. Two approaches for progressive image simplification with macro structure preserved: (top) CFG scale set to zero, (bottom) conditional text prompt set to empty.

With the target image as input, a sequence of progressively simplified images is obtained every N optimization steps ($N = 20$ in our work). Figure 7 gives an example. Compared to other image simplification methods, e.g., Superpixel [1], Bilateral filter [52], and Gaussian filter, our SDS-based method effectively abstracts the image while maintaining the macro shapes. Also, the boundaries of detected object masks (the rightmost column of Figure 7) are smoothed and compatible with vector-based graphics.

5. Vector Reconstruction

Guided by the sequence of simplified images, vectors are initialized and optimized via differential rendering [32] in two stages: structural construction and visual refinement. Below, we introduce the key components and loss functions.

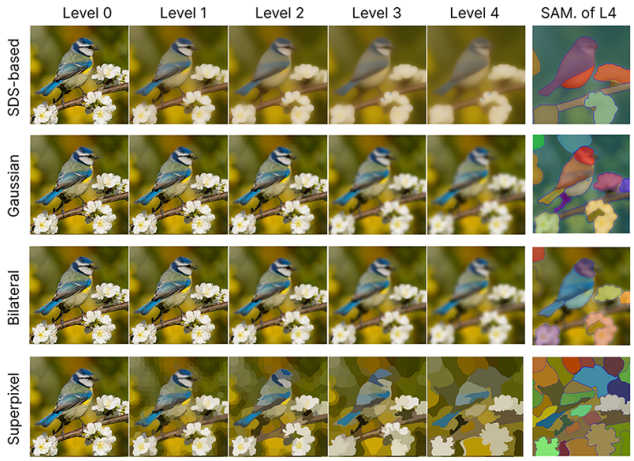

Figure 7. Example of SDS-based image simplification compared to other methods: using the SDS-based method, the macro semantic structures (e.g., 'flower') are obtained with smooth boundaries. The rightmost column shows the semantic segmentation of the simplified level 4.

5.1. Stage I: Structural Construction

Layering of Segmented Masks The sequence of simplified images is semantically segmented to detect object masks. Subsequently, the masks are organized into layers, with large masks on the back layer and small masks overlaid on the front. Masks within a layer are not intersected with each other. As illustrated in Figure 8, masks are added progressively from the most simplified image to the least simplified (i.e., the input target), with each mask placed in the furthest back layer where it does not overlap with other masks already positioned on the same layer.

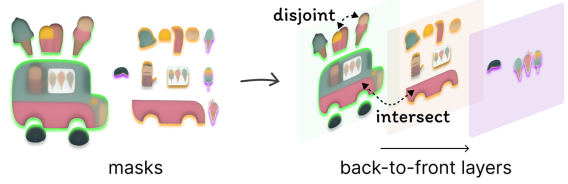

Figure 8. Mask layering: iterating from the most simplified to the least, a mask is only added into the layer from back to front, provided it does not intersect with other masks already in that layer.

Once the masks are layered, each mask's boundary is traced and simplified using the Douglas–Peucker algorithm [9]) to reduce the number of points in the boundary. For each mask, a structure-wise vector is initialized as a closed shape of cubic Bézier curves, with control points set to points in the simplified boundary of the mask. Then structure-wise vectors are rendered by layers and optimized together to minimize the structure loss.

Layer-wise Structure Loss The structure loss is computing with two components. One component is the layer-wise MSE loss $\mathcal{L}_{mse}$, which measures the image differences between each pair of the mask layer I_{mask_j} and vector layer I_{vector_j}, defined as follows:

$$\mathcal{L}_{\text{mse}} = \sum_{j=1}^{n} \|I_{\text{mask}_j} - I_{\text{vector}_j}\|_2^2, \tag{3}$$

where n is the number of layers. In the structural construction phase, the focus is on optimizing shape rather than other visual properties such as color. Therefore each pair of mask and corresponding vector are rendered with the same randomly assigned color.

The second component is the overlap loss $\mathcal{L}_{overlap}$, which penalizes the overlap among structure-wise vectors within a layer, defined as follows:

$$\mathcal{L}_{\text{overlap}} = \sum_{j=1}^{n} \sum_{p \in I_{layer_j}} ReLU(\theta - \alpha(p)), \tag{4}$$

where vectors are rendered with the same semi-transparent gray in I_{layer_j}, the $\alpha(p)$ is the transparency value of pixel p, and θ is the transparency threshold. Using these components, the structure loss function is a joint loss with weights $w_1 = 1$ and $w_2 = 1e{-}8$:

$$\mathcal{L}_{\text{structure}} = w_1 \mathcal{L}_{\text{mse}} + w_2 \mathcal{L}_{\text{overlap}}. \tag{5}$$

5.2. Stage II: Visual Refinement

Color Fitting After optimizing the structure-wise vectors to desirable mask shapes, a color fitting is applied to assign a color to each vector. There can be different coloring strategies. In this work, we introduce two. One is to assign the most dominant color from the visible pixels the vector covers in the target image. Another is to fit colors by minimizing the MSE loss between rasterized vectors and the target image. Once the color fitting is accomplished, structure-wise vectors are frozen during subsequent visual refinement.

To achieve high visual fidelity with the target image, visual-wise vectors are initialized following a strategy similar to LIVE [35]. Vectors are initialized to the top-K largest connected areas with pixel-level differences between the rasterized vectors and the target image, and optimized to minimize visual fidelity loss.

Visual Fidelity Loss The visual fidelity loss $\mathcal{L}_{\text{fidelity}}$ measures the faithfulness of all vectors to the input image, i.e., how visually similar the rasterized vectors I_{vector} is to the input target I_{target}. Therefore We define $\mathcal{L}_{\text{fidelity}}$ as their RBG error under L_2 norm:

$$\mathcal{L}_{\text{fidelity}} = \|I_{\text{target}} - I_{\text{vector}}\|_2^2. \tag{6}$$

6. Implementation

We implemented this method using PyTorch with the Adam optimizer. By default, a sequence of five simplified images (including the original input image) is generated at intervals of 20 SDS iterations. The learning rates for optimizing primitive points and their colors are set to 1.0 and 0.01 respectively. All examples and experiments in this paper were conducted on a system running Ubuntu 20.04.6 LTS, equipped with an Intel Xeon Gold 5320 CPU operating at 2.20 GHz and four NVIDIA A40 GPUs. Each GPU features 48 GB of GDDR6 memory with ECC.

7. Evaluation

In this section, we first report the results of the ablation study, and then elaborate on the comparison between our method and four state-of-the-art methods.

7.1. Ablation Study

Ablation on the Guide of Simplified Image Sequence We investigated the impact of incorporating *a sequence of simplified images* in the vectorization process, compared to an ablated version that relies solely on a single input image, i.e., *without the sequence.* As shown in Figure 9, using the sequence of simplified images as the intermediate targets enables our approach to create more implicit semantic vectors, such as the 'entire body of Captain America', and the 'grassland', which are missed when the sequence is not used. These richer layers allow for fine-grained management of vector elements.

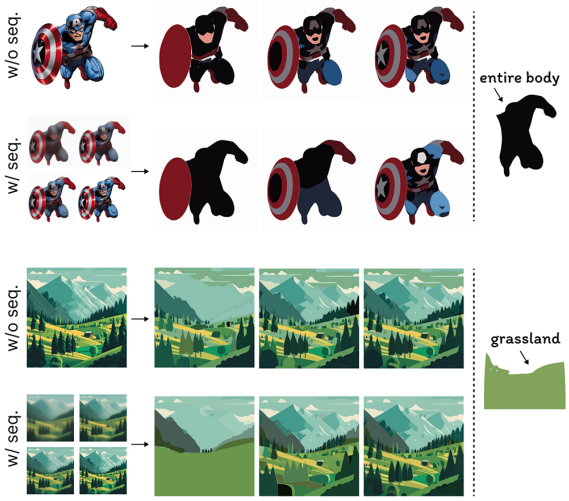

Figure 9. Comparison of structure-wise vectors with and without the simplification sequence: incorporating the simplification sequence creates more implicit semantic vectors, e.g., the 'entire body of Captain America', and 'grassland'.

Ablation on SDS-based Image Simplification We evaluated the effectiveness of *SDS-based simplification* against three conventional image simplification methods: Bilateral filtering, Gaussian filtering, and Superpixel-based Simplification. As shown in Figure 10(a), the SDS-based method preserves clear boundaries, such as the round shape of 'the ladybird', more effectively than the other three methods. Also, the SDS-based method smartly removes less featured elements, such as the 'trees in front of the house', and recovers the occluded parts of the house in Figure 10(b). The complete constructed vector sequences are provided in the supplementary materials.

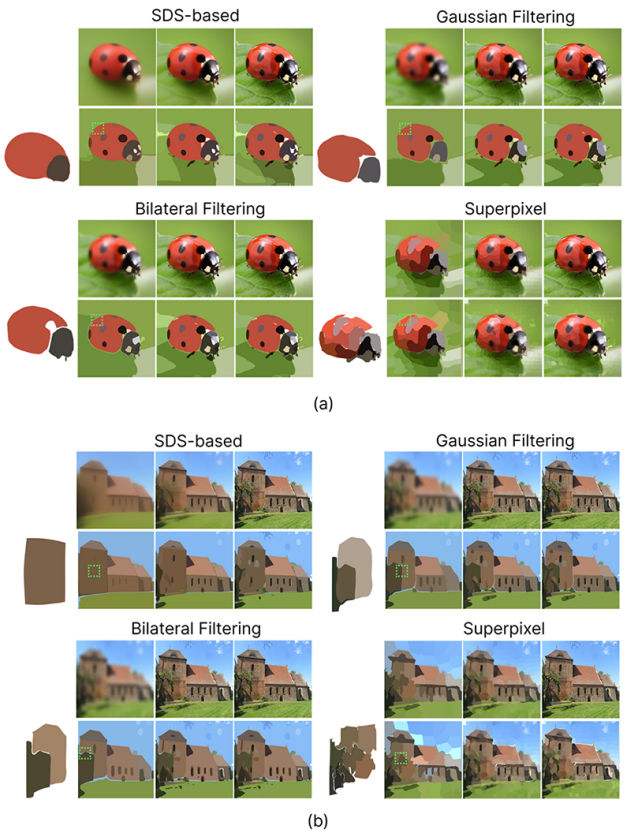

Figure 10. Comparison of structure-wise vectors between SDS-based method and three conventional image simplification methods: (a) SDS-based method retains the round boundary of the 'ladybird', (b) and semantically recovers the 'front wall of the house'.

7.2. Comparison Experiment

We compared our method to existing methods, including DiffVG [32], LIVE [35], O&R [20], and SGLIVE [67]. To make the comparison fair, we use the same visual primitives for all five methods. More details on the experiment set-up and results are provided in the supplementary materials.

Visual Quality We examined the visual rendering fidelity of the generated vectors, specifically how closely they resemble the original input image. Figure 11 shows the visual comparison with the four state-of-the-art methods. Clearly, our method demonstrates a more faithful reconstruction with clean, compact, and semantically aligned boundaries, while the other four methods exhibit artifacts in colors and shapes.

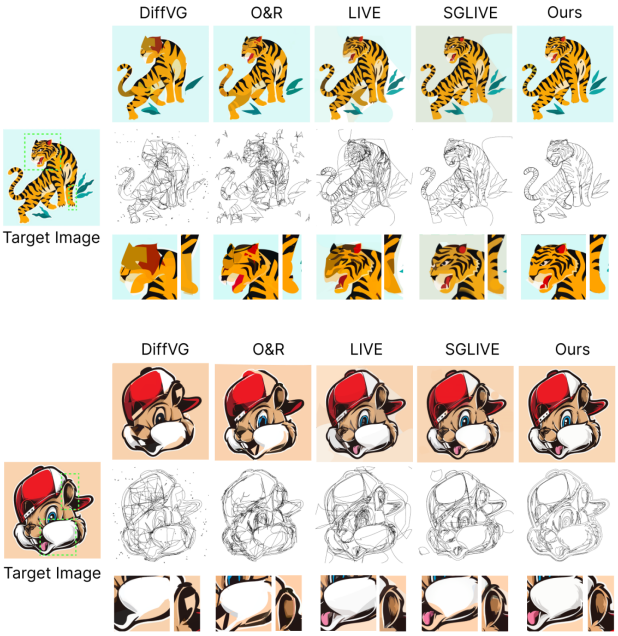

Figure 11. Qualitative reconstruction comparison: both examples are vectorized with 128 vector primitives. Our method reconstructs more faithful, clean, and semantic-aligned vectors.

To quantify this, we calculated the pixel MSE and LPIPS computed based on VGG [44] between the rasterized image of vectors and the original image. We collected a testing dataset of 100 images, including realistic photos, clipart images, emojis. Figure 12 shows the result on this testing dataset. Our method reconstructs with lower MSE and LPIPS than the other four methods.

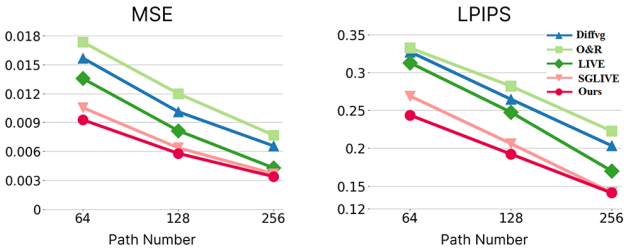

Figure 12. MSE and LPIPS comparison: our method reconstructs more faithful vectors across different numbers of vector primitives.

Layer-wise Representation To quantify this, we introduce the metric *Vector Compactness (VeC)* to assess how well the primitives are contained within the semantic object boundaries. Given a semantic mask (e.g., the area of pixels segmented as 'butterfly'), VeC is defined as the ratio of vectors highly contained within the mask (i.e., exceeding 85% area overlap) to the total number of vectors interacting with the mask.

Table 1 reports the average VeC of all images in our testing dataset. For each image, we sampled four masks randomly from its semantic segmentation. As can be seen, our method maintains significantly higher compactness of primitives compared to the other four methods, with approximately 73.8%.

Table 1. Comparison of the average VeC and standard deviation of the 100 testing images.

VeC (%)	DiffVG	LIVE	O&R	SGLIVE	Ours
Avg.	41.9	43.4	39.9	65.9	**73.8**↑
Std.	15.1	17.4	20.4	18.5	11.9

As visually evident in Figure 13(a), vector primitives in methods like LIVE and O&R tend to exhibit scattering and inter-region intersections (highlighted in red). Our method demonstrates superior performance by retaining a large portion of the vector primitives within coherent semantic structures (green). This compact layered construction facilitates vector editing tasks, such as image recoloring. For example, in Figure 13(b), upper-layer primitives can be easily selected using the underlying layer structures and recolored by applying a specified hue shift. Our method effectively preserves the details of texture and lighting, even after recoloring.

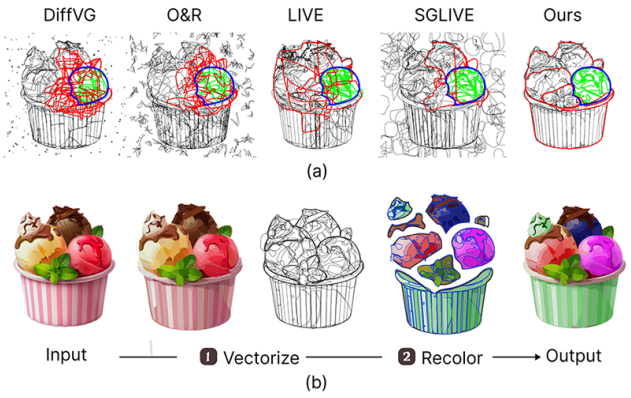

Figure 13. Layered representation and editing: (a) looking at the mask of 'ice cream ball' (highlighted in blue), vectors highly contained in the ice cream are colored in green, those intersected but not contained are in red. (b) our compact layer-wise vector representation facilitates recoloring.

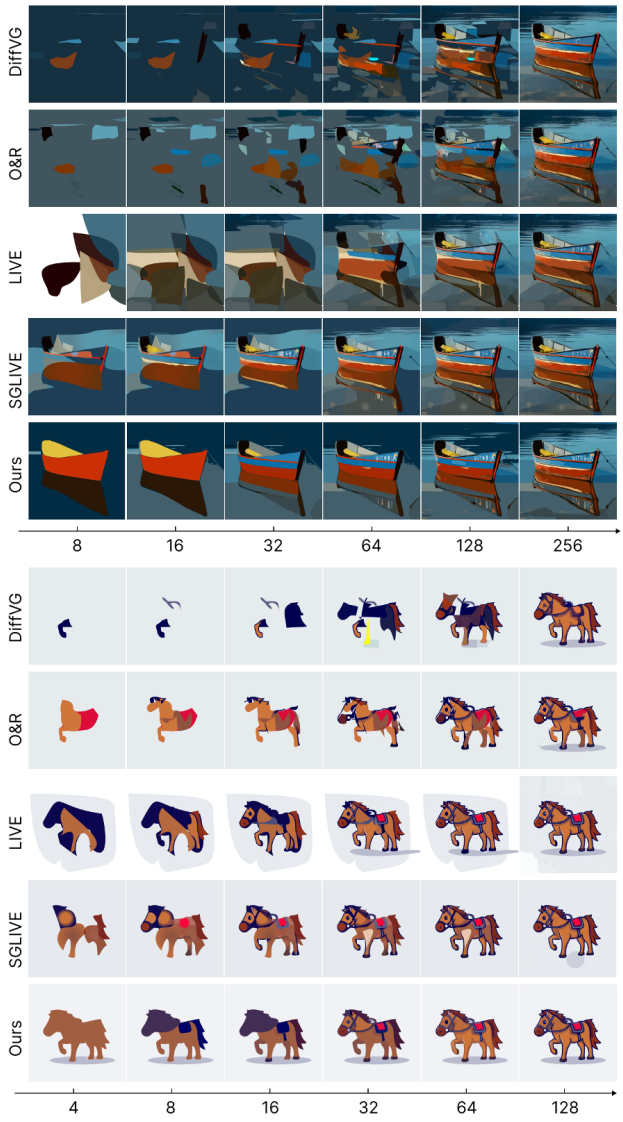

Figure 14. Vector layers generated by LIVE [35], DiffVG [32], O&R [20], SGLIVE [68], and our method.

Semantic Alignment We examined how well the underlying vector primitives align with meaningful structures in the image. Figure 14 shows the two examples (128 paths in 'horse' and 256 in 'boat'), comparing the sequence of intermediate accumulated vectors from back to front layers. As shown, our method outperforms others, generating primitives that progress from macro to fine detail and align more accurately with the underlying semantic structures.

Figure 15 shows some captions generated for coarse vector layers of our vectorization method. Florence-2 model is applied to infer captions from images. It can be seen the descriptive text generated from coarse layers aligns well with the contents in the original target image.

To quantify this, we used the CLIP score [40] to measure the semantic similarity between the generated caption of the input target image and the rasterized SVG images. As shown in Figure 16, at every sampled path number, our method produces vectorized images that retain a closer match to the original image's content, particularly at the beginning stages where there are only a few vector paths.

Figure 15. Captioning of macro structural vectors generated by our vectorization method: for each example, the caption of the coarse image is generated by Florence-2 model [17].

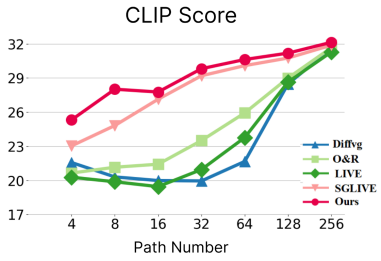

Figure 16. Comparison of CLIP similarity score: our method manages to preserve key semantic features of the original image better than other methods, especially with a few vectors.

8. Conclusion

In this work, we present a novel image vectorization technique that utilizes a sequence of progressively simplified images to guide vector reconstruction. Our approach introduces an SDS-based image simplification method that achieves effective visual abstraction. Through a two-stage vector reconstruction process, our approach emphasizes both visual fidelity and structural manageability, producing a layered vector representation that captures the target image's structure from macro to fine details. Our method demonstrates superior performance in multiple areas, including visual fidelity, layered representation, and semantic alignment of vectors with underlying structures, greatly enhancing usability for further editing and modification.

Acknowledgments This work is supported in parts by the National Natural Science Foundation of China (NSFC) Program (62472288, 12471502), Shenzhen Science and Technology Program (20231122121504001, 20231121165649002), Israel Science Foundation (3441/21), Scientific Development Funds of Shenzhen University (000001032518), and Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), MNR Key Laboratory for Geo-Environmental Monitoring of Great Bay Area and Guangdong Key Laboratory of Urban Informatics.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012. 4

[2] Yağiz Aksoy, Tunç Ozan Aydin, Aljoša Smolić, and Marc Pollefeys. Unmixing-based soft color segmentation for image manipulation. *ACM Trans. Graph.*, 36(4), 2017. 2

[3] Sebastiano Battiato, Giovanni Gallo, and Giuseppe Messina. Svg rendering of real images using data dependent triangulation. In *Proceedings of the 20th Spring Conference on Computer Graphics*, page 185–192, New York, NY, USA, 2004. Association for Computing Machinery. 2

[4] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. Graph.*, 33(4), 2014. 2

[5] Ye Chen, Bingbing Ni, Xuanhong Chen, and Zhangli Hu. Editable image geometric abstraction via neural primitive assembly. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 23457–23466, 2023. 2

[6] Ye Chen, Bingbing Ni, Jinfan Liu, Xiaoyang Huang, and Xuanhong Chen. Towards high-fidelity artistic image vectorization via texture-encapsulated shape parameterization. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15877–15886, 2024. 2

[7] Laurent Demaret, Nira Dyn, and Armin Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604–1616, 2006. 2

[8] Edoardo Alberto Dominici, Nico Schertler, Jonathan Griffin, Shayan Hoshyari, Leonid Sigal, and Alla Sheffer. Polyfit: perception-aligned vectorization of raster clip-art via intermediate polygonal fitting. *ACM Trans. Graph.*, 39(4), 2020. 2

[9] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10 (2):112–122, 1973. 5

[10] Zheng-Jun Du, Kai-Xiang Lei, Kun Xu, Jianchao Tan, and Yotam I Gingold. Video recoloring via spatial-temporal geometric palettes. *ACM Trans. Graph.*, 40(4):150–1, 2021. 2

[11] Zheng-Jun Du, Liang-Fu Kang, Jianchao Tan, Yotam Gingold, and Kun Xu. Image vectorization and editing via linear gradient layer decomposition. *ACM Transactions on Graphics (TOG)*, 42(4):1–13, 2023. 2

[12] Maria Dziuba, Ivan Jarsky, Valeria Efimova, and Andrey Filchenkov. Image vectorization: a review, 2023. 2

[13] Vage Egiazarian, Oleg Voynov, Alexey Artemov, Denis Volkhonskiy, Aleksandr Safin, Maria Taktasheva, Denis Zorin, and Evgeny Burnaev. Deep vectorization of technical drawings. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII*, page 582–598, Berlin, Heidelberg, 2020. Springer-Verlag. 2

[14] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Trans. Graph.*, 35(4), 2016. 2

[15] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Photo2clipart: image abstraction and vectorization using layered linear gradients. *ACM Trans. Graph.*, 36 (6), 2017. 2

[16] Jon Ferraiolo, Fujisawa Jun, and Dean Jackson. *Scalable vector graphics (SVG) 1.0 specification*. iuniverse Bloomington, 2000. 1

[17] Florence. MS Windows NT kernel description, 2024. 8

[18] Hongbo Fu, Shizhe Zhou, Ligang Liu, and Niloy J Mitra. Animated construction of line drawings. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–10, 2011. 2

[19] Amir Hertz, Kfir Aberman, and Daniel Cohen-Or. Delta denoising score. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2328–2337, 2023. 2, 4

[20] Or Hirschorn, Amir Jevnisek, and Shai Avidan. Optimize & reduce: A top-down approach for image vectorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2148–2156, 2024. 1, 2, 3, 6, 8

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 4

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 4

[23] Shayan Hoshyari, Edoardo Alberto Dominici, Alla Sheffer, Nathan Carr, Zhaowen Wang, Duygu Ceylan, and I-Chao Shen. Perception-driven semi-structured boundary vectorization. *ACM Trans. Graph.*, 37(4), 2018. 2

[24] Qiming Hu and Xiaojie Guo. Single image reflection separation via component synergy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13138–13147, 2023. 2

[25] Teng Hu, Ran Yi, Baihong Qian, Jiangning Zhang, Paul L Rosin, and Yu-Kun Lai. Supersvg: Superpixel-based scalable vector graphics synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24892–24901, 2024. 2

[26] Ajay Jain, Amber Xie, and Pieter Abbeel. Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1911–1920, 2023. 1

[27] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer White-

head, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 3

[28] Johannes Kopf and Dani Lischinski. Depixelizing pixel art. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2011)*, 30(4):99:1 – 99:8, 2011. 2

[29] Yuki Koyama and Masataka Goto. Decomposing images into layers with advanced color blending. In *Computer Graphics Forum*, pages 397–407. Wiley Online Library, 2018. 2

[30] Gregory Lecot and Bruno Levy. Ardeco: Automatic region detection and conversion. In *17th Eurographics Symposium on Rendering-EGSR'06*, pages 349–360, 2006. 2

[31] Anat Levin, Assaf Zomet, and Yair Weiss. Separating reflections from a single image using local features. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, pages I–I. IEEE, 2004. 2

[32] Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph.*, 39(6), 2020. 2, 3, 4, 6, 8

[33] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. *arXiv preprint arXiv:2311.11284*, 2023. 4

[34] Zicheng Liao, Hugues Hoppe, David Forsyth, and Yizhou Yu. A subdivision-based representation for vector image editing. *IEEE transactions on visualization and computer graphics*, 18(11):1858–1867, 2012. 2

[35] 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 16293–16302, 2022. 1, 2, 3, 5, 6, 8

[36] Abhimitra Meka, Mohammad Shafiei, Michael Zollhöfer, Christian Richardt, and Christian Theobalt. Real-time global illumination decomposition of videos. *ACM Trans. Graph.*, 40(3), 2021. 2

[37] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics (TOG)*, 27(3):1–8, 2008. 2

[38] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 2, 3, 4

[39] Thomas Porter and Tom Duff. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 253–259, 1984. 2

[40] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 8

[41] 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. 2

[42] C. Richardt, J. Lopez-Moreno, A. Bousseau, M. Agrawala, and G. Drettakis. Vectorising bitmaps into semi-transparent gradient layers. In *Proceedings of the 25th Eurographics Symposium on Rendering*, page 11–19, Goslar, DEU, 2014. Eurographics Association. 2

[43] I-Chao Shen and Bing-Yu Chen. Clipgen: A deep generative model for clipart vectorization and synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 28 (12):4211–4224, 2022. 2

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, San Diego, CA, USA, 2015. 7

[45] Alvy Ray Smith and James F Blinn. Blue screen matting. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 259–268, 1996. 2

[46] Yiren Song, Shijie Huang, Chen Yao, Xiaojun Ye, Hai Ci, Jiaming Liu, Yuxuan Zhang, and Mike Zheng Shou. Processpainter: Learn painting process from sequence data. *arXiv preprint arXiv:2406.06062*, 2024. 3

[47] Hao Su, Xuefeng Liu, Jianwei Niu, Jiahe Cui, Ji Wan, Xinghao Wu, and Nana Wang. Marvel: Raster gray-level manga vectorization via primitive-wise deep reinforcement learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(4):2677–2693, 2024. 2

[48] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. *ACM Trans. Graph.*, 26(3):11–es, 2007. 2

[49] Sriram Swaminarayan and Lakshman Prasad. Rapid automated polygonal image decomposition. In *35th IEEE Applied Imagery and Pattern Recognition Workshop (AIPR'06)*, pages 28–28, 2006. 2

[50] Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. Decomposing images into layers via rgb-space geometry. *ACM Trans. Graph.*, 36(1), 2016. 2

[51] Xingze Tian and Tobias Günther. A survey of smooth vector graphics: Recent advances in representation, creation, rasterization, and image vectorization. *IEEE Transactions on Visualization and Computer Graphics*, 30(3):1652–1671, 2024. 2

[52] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Sixth international conference on computer vision (IEEE Cat. No. 98CH36271)*, pages 839–846. IEEE, 1998. 4

[53] Yili Wang, Yifan Liu, and Kun Xu. An improved geometric approach for palette-based image decomposition and recoloring. *Computer Graphics Forum*, 38:11–22, 2019. 2

[54] Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Iconshop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics (TOG)*, 42(6): 1–14, 2023. 1

[55] Tian Xia, Binbin Liao, and Yizhou Yu. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics (TOG)*, 28(5):1–10, 2009. 2

[56] Guofu Xie, Xin Sun, Xin Tong, and Derek Nowrouzezahrai. Hierarchical diffusion curves for accurate automatic image vectorization. *ACM Trans. Graph.*, 33(6), 2014. 2

[57] Songhua Xu, Yingqing Xu, Sing Bing Kang, David H Salesin, Yunhe Pan, and Heung-Yeung Shum. Animating chinese paintings through stroke-based decomposition. *ACM Transactions on Graphics (TOG)*, 25(2):239–267, 2006. 2

[58] Chuan Yan, Yong Li, Deepali Aneja, Matthew Fisher, Edgar Simo-Serra, and Yotam Gingold. Deep sketch vectorization via implicit surface extraction. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024. 2

[59] 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*, 22(2):1063–1075, 2016. 2

[60] Lvmin Zhang and Maneesh Agrawala. Transparent image layer diffusion using latent transparency. *arXiv preprint arXiv:2402.17113*, 2024. 2

[61] Peiying Zhang, Nanxuan Zhao, and Jing Liao. Text-to-vector generation with neural path representation. *ACM Transactions on Graphics (TOG)*, 43(4):1–13, 2024. 1

[62] Qing Zhang, Chunxia Xiao, Hanqiu Sun, and Feng Tang. Palette-based image recoloring using color decomposition optimization. *IEEE Transactions on Image Processing*, 26 (4):1952–1964, 2017. 2

[63] Qing Zhang, Yongwei Nie, Lei Zhu, Chunxia Xiao, and Wei-Shi Zheng. A blind color separation model for faithful palette-based image recoloring. *IEEE Transactions on Multimedia*, 24:1545–1557, 2021. 2

[64] Song-Hai Zhang, Tao Chen, Yi-Fei Zhang, Shi-Min Hu, and Ralph R. Martin. Vectorizing cartoon animations. *IEEE Transactions on Visualization and Computer Graphics*, 15 (4):618–629, 2009. 2

[65] Xuaner Zhang, Ren Ng, and Qifeng Chen. Single image reflection separation with perceptual losses. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4786–4794, 2018. 2

[66] Shuang Zhao, Frédo Durand, and Changxi Zheng. Inverse diffusion curves using shape optimization. *IEEE Transactions on Visualization and Computer Graphics*, 24(7):2153–2166, 2018. 2

[67] Hengyu Zhou, Hui Zhang, and Bin Wang. Segmentation-guided layer-wise image vectorization with gradient fills. *arXiv preprint arXiv:2408.15741*, 2024. 1, 2, 6

[68] Haokun Zhu, Juang Ian Chong, Teng Hu, Ran Yi, Yu-Kun Lai, and Paul L Rosin. Samvg: A multi-stage image vectorization model with the segment-anything model. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4350–4354. IEEE, 2024. 2, 8

[69] Zhengxia Zou, Tianyang Shi, Shuang Qiu, Yi Yuan, and Zhenwei Shi. Stylized neural painting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15689–15698, 2021. 3