Easy-editable Image Vectorization with Multi-layer Multi-scale Distributed Visual Feature Embedding

Ye Chen[1] Zhangli Hu[1] Zhongyin Zhao[1] Yupeng Zhu[1] Yue Shi[1] Yuxuan Xiong[1] Bingbing Ni[1,2*]
[1]Shanghai Jiao Tong University, Shanghai 200240, China
[2]USC-SJTU Institute of Cultural and Creative Industry
{chenye123, nibingbing}@sjtu.edu.cn

Abstract

Current parameterized image representations embed visual information along the semantic boundaries and struggle to express the internal detailed texture structures of image components, leading to a lack of content consistency after image editing and driving. To address these challenges, this work proposes a novel parameterized representation based on hierarchical image proxy geometry, utilizing multi-layer hierarchically interrelated proxy geometric control points to embed multi-scale long-range structures and fine-grained texture details. The proposed representation enables smoother and more continuous interpolation during image rendering and ensures high-quality consistency within image components during image editing. Additionally, under the layer-wise representation strategy based on semantic-aware image layer decomposition, we enable decoupled image shape/texture editing of the targets of interest within the image. Extensive experimental results on image vectorization and editing tasks demonstrate that our proposed method achieves high rendering accuracy of general images, including natural images, with a significantly higher image parameter compression ratio, facilitating user-friendly editing of image semantic components.

1. Introduction

The utilization of implicit functions for encoding image shape and texture information has gained significant attention in recent years [4, 7, 23, 27, 29]. This is primarily due to the fact that implicit neural network models represent images as functions defined over a continuous domain, independent of grid-based image sampling/rendering, which allows for highly precise representation of image content. Additionally, because the implicit parameters are much sparser than the original pixel-wise *RGB* values, such approach also achieves image compression. However, implicit image representations inherently function as black-box mapping models, lacking alignment between the image's semantic components and corresponding representation parameters. As a result, such representations are not conducive to image content editing, as users are unable to manipulate specific local components of the image.

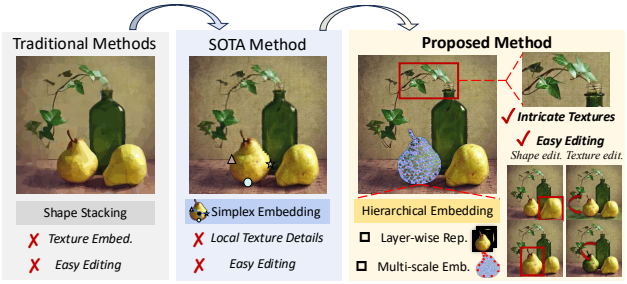

Figure 1. **Illustration of our motivation.** Traditional methods heavily rely on stacking a large number of shapes to approximate textures due to the lack of shape-aware texture modeling, which not only results in low quality texture representation but also makes image editing extremely challenging. Chen *et al.* [6] propose to embed textures at shape contour points for improving texture representation, but this approach is still limited to simple artistic images. Additionally, due to the lack of layer-wise modeling and the sparse texture interpolation, it fails to support precise image editing. This work explores a multi-layer multi-scale texture embedding with the help of hierarchically related proxy geometric nodes, tailored for high quality reconstruction of complex natural images with lightweight parameters, which also ensures high-fidelity image shape/texture editing.

To achieve editable representations, recent works propose vectorized image representation (image vectorization) frameworks [6, 11, 12, 15, 19, 21, 30, 33]. The principle behind this approach is to use a collection of vectorized geometric primitives like Bézier shapes to represent the image explicitly. Traditional vectorization algorithms [11, 12, 18, 21, 24, 30, 31], due to the lack of shape-aware texture modeling, require stacking of numerous shapes to approximate image textures, which not only reduces the representation accuracy but also makes subse-

*Corresponding author: Bingbing Ni

quent image editing extremely difficult. To achieve more precise texture representation and editability, recent scholars [10] propose to decouple images into shape and texture parameters through spatially distributing implicit texture parameters into multiple control points along Bézier shape contors, allowing for the interpolation-based implicit mapping of textures within the shape. Unfortunately, although the implicit representation enhanced by distributed explicit control nodes improves the accuracy of texture sampling, the texture codes are primarily embedded along the contours (or semantic boundaries), making it difficult to establish a stable (coordinate-to-feature) mapping support domain for the internal areas of image components. This leads to two significant challenges: 1) Due to weak texture expression capacity, the above framework can only model textures for simple artistic images, performing poorly on categories like natural images with more complex texture and structures; 2) The coupling of texture mapping network parameters with the sparse spatial distribution of control points leads to the texture being overfitted to fixed spatial locations, resulting in significant distortion when resampling internal textures after shape editing (*e.g.*, stretching and deformation), which does not support general image editing. Additionally, most existing vectorized image representations share the common issue of lacking a semantic layer concept, making it challenging to reconstruct reasonable images after editing a single visual target.

To address these challenges, this work proposes a new representation framework based on multi-layer hierarchical image proxy geometry. First, we utilize the capabilities of currently available large semantic understanding models [17, 32] to achieve semantic layering of the input image, facilitating image editing and image reassembly after image manipulation. Second, for each semantic component, we sample multiple hierarchically interrelated image proxy nodes (*i.e.*, from control points on the external boundary of the target component to internal mesh points) through adaptive Bézier control points fitting and dynamic multi-scale mesh refinement constrained by boundary Bézier curves and the internal distribution of geometric information. Then we embed implicit representation parameters of image information at various scales into these distributed multi-scale proxy geometric nodes. During image decoding and reconstruction, the implicit parameters provided by the relevant proxy geometric nodes are fused to generate a mapping from coordinates to pixel values, achieving more continuous and smoother interpolation results with multi-scale image content integration. Compared to previous distributed embedding representation frameworks, our approach offers a more stable support domain for the implicit mapping function thanks to the proxy nodes' ability to integrate multi-scale envelope shape structure and internal details of semantic components. As a result, our method supports accurate texture representation for natural images with complex textures, maintains high-quality texture consistency within image components during editing, and facilitates more precise image texture editing and target manipulation. Extensive experimental results on image vectorizaion and image editing tasks (across various benchmarks including ImageNet [28]) demonstrate the high rendering accuracy of the proposed framework for general image, including natural images, as well as its user-friendly capabilities for image shape/texture editing. In addition, we also make comparisons with neural implicit image representations, our method achieves significantly better reconstruction quality with a higher parameter compression ratio, which significantly demonstrates the effectiveness and efficiency of our representation.

2. Related Works

Image Vectorization. Image Vectorization, as a bridge between rasterized pixel images and scalable vector graphics, has evolved significantly over recent years. Empirical algorithms often capture image contours and model textures as iterative optimization problems, such as diffusion curves [24, 31] and gradient meshes [18, 28]. Diffvg [19] introduces a differentiable rasterizer that facilitates the use of machine/deep learning techniques for image vectorization. Subsequent methods [5, 6, 10, 11, 21, 25, 30] widely adopt closed Bézier shapes as fundamental primitives, obtaining a series of Bézier control points through neural network training or iterative parameter optimization. LIVE [21] is a remarkable work that achieves layer-wise image vectorization. However, the hierarchical structure in LIVE lacks semantic information, and each layer relies on the stacking of numerous irregular shapes, making it impossible for image editing. SuperSVG [12] creatively uses a ViT-based model [9] to predict shape parameters in a feedforward manner, achieving good reconstruction accuracy for natural images. However, neural network-based approaches require significant computational resources and are difficult to generalize to arbitrary images. In our work, we propose an optimization-based framework to decompose images into semantic-aware layers and explore complex image textures with multi-scale texture embedding, which achieves high-fidelity image vectorization results for arbitrary images, endowed with easy-editability.

Neural Implicit Image Parameterization. Most image vectorization algorithms rely on shape stacking to represent textures, lacking efficient modeling for complex textures. Alternatively, another category of image parameterization algorithms (*i.e.*, neural implicit representation [4, 6, 7, 23, 27, 29]) approach the problem from an implicit representation perspective, achieving efficient coordinate-to-texture mapping through joint optimization of texture features and a decoding function. Deep image prior [29] is a pioneer-

ing work to utilize a deep network with millions of parameters to accurately fit images. SIREN [27] uses MLPs with periodic activation functions to represent images, requiring highly redundant MLP parameters for reasonable image reconstruction. Chen *et al*. [6] propose a shape-anchored distributed texture representation, achieving significant parameter compression. However, texture features distributed along semantic boundaries struggle to capture the internal texture structure of image components. As a result, [6] is limited to simple artistic images and performs poorly on natural images. In addition, representations mentioned above do not possess image editing capabilities. Our method introduces hierarchical geometric nodes to embed image textures in a multi-layer multi-scale manner, enabling efficient reconstruction of natural images with minimal parameters, endowed with easy image editing ability.

Image Geometry. This work explores image geometry, particularly in relation to triangulation in 2D domain [3, 13, 14, 20, 26]. Triwild [14] introduces a robust 2D meshing method that generates curved triangles capable of reproducing smooth feature curves. This work utilizes Triwild constrained by Bézier shapes to explore local image geometry within each layer, enabling multi-scale texture embedding.

3. Methodology

3.1. Overview

Our framework is illustrated in Fig. 2. We parameterize any input image $\boldsymbol{I} \in \mathbb{R}^{W\times H\times 3}$ into texture embeddings distributed at multi-layer hierarchical geometric control points and a decoding function with parameters θ which decodes textures in a coord-to-rgb manner:

$$\boldsymbol{I} \sim \{\{\boldsymbol{R}_1, \boldsymbol{R}_2, ..., \boldsymbol{R}_L\}, \theta\}, \tag{1}$$

where L is the number of image layers. $\boldsymbol{R}_i$ denotes the i-th image layer and takes the form: $\boldsymbol{R}_i = \{\boldsymbol{B}_i, \boldsymbol{G}_i, \boldsymbol{F}_i\}$, where $\boldsymbol{B}_i$ means the Bézier shape to fit layer boundary, $\boldsymbol{G}_i$ denotes the multi-scale proxy nodes within $\boldsymbol{B}_i$ and $\boldsymbol{F}_i$ denotes the distributed texture features. The technical details are elaborated in the following sections.

3.2. Semantic-aware Image Layer Decomposition

Current image vectorization algorithms [12, 21, 25, 30] frequently stack and couple numerous redundant shapes, which obscures the relationships and hierarchies among image components, resulting in challenges for easy image editing. For example, changing the color or shape of a specific region may inevitably alter overlapping shapes, causing unintended disruptions.

The above observation motivates us to propose a semantic-aware image layer decomposition module that decomposes the input image $\boldsymbol{I}$ into a series of background regions and foreground regions. To achieve this, we first employ the Segment Anything Model (SAM) [17] to generate a series of separate image regions: $\{\boldsymbol{M}_1, \boldsymbol{M}_2..., \boldsymbol{M}_L\} = SAM(\boldsymbol{I})$, where $\boldsymbol{M}_i \in \{0,1\}^{W\times H}$ represents the $i-$th image layer (as mask). Next, we utilize the Depth Anything Model [32] to generate the depth map of the input image and calculate the average depth value $\bar{d}_i$ for each masked region $\boldsymbol{M}_i$. Then we cluster all regions into two sorted clusters (*i.e*., background and foreground areas) via k-means clustering [16]:

$$\{\{\boldsymbol{M}_1^f, ..., \boldsymbol{M}_{L_1}^f\}, \{\boldsymbol{M}_1^b, ..., \boldsymbol{M}_{L_2}^b\}\} = cluster(\{\bar{d}_1, ...\bar{d}_L\}). \tag{2}$$

Hence, we decompose the input image into L_1 foreground layers and L_2 background layers sorted by average depth values and can render the image layer by layer according to this depth ordering during subsequent texture parameter optimization process, enabling the separation of the background and various foreground semantic components.

3.3. Hierarchical Parameterized Image Geometry

For each image layer $\boldsymbol{M}_i$, we aim to efficiently fit its edges using a single Bézier shape (Sec. 3.3.1). Additionally, to capture complex local structures, we also employ hierarchically related proxy geometry within the shape (Sec. 3.3.2).

3.3.1. Adaptive Bézier Shape Fitting

Existing image vectorization methods [5, 6, 21, 25] often require a predefined number of segments per shape to simultaneously optimize a large number of Bézier shapes. However, the number of segments is difficult to define: too few segments limit the expressive power of the shapes, necessitating extensive shape stacking to represent complex geometries, while too many segments lead to parameter redundancy and optimization challenges. In contrast, we model each layer as a single shape with an adaptive control point optimization strategy, thus we can fit any shape with an arbitrary number of control points adaptively and avoid shape stacking, which not only enhances the accuracy and parameter efficiency of Bézier shapes, but also facilitates easy editing of image semantic components.

Specifically, we first detect the shape boundary points $\mathbb{E}_i = \{\boldsymbol{e}_k\}$ and sample $3m$ points from them as initialization of control points for an $m-$segment Bézier shape $\boldsymbol{B}_i = \{\boldsymbol{s}_{i1}, ..., \boldsymbol{s}_{im}\}$, where each segment $\boldsymbol{s}_{ij}$ is parameterized by four control points $\{\boldsymbol{p}_{ij}^0, \boldsymbol{p}_{ij}^1, \boldsymbol{p}_{ij}^2, \boldsymbol{p}_{ij}^3\}$ and we can sample points in this segment uniformly by:

$$\mathbb{B}_{ij} = \{\boldsymbol{b}_{ij}(t) = \sum_{o=0}^{3} \binom{3}{o} (1-t)^{3-o} t^o \boldsymbol{p}_{ij}^o\}, \tag{3}$$

where $t \in [0,1]$ denotes the sampling coefficient. The points sampled across the entire shape can be denoted as $\mathbb{B}_i = \cup_{j=1}^{m} \mathbb{B}_{ij}$. We then calculate the Chamfer distance

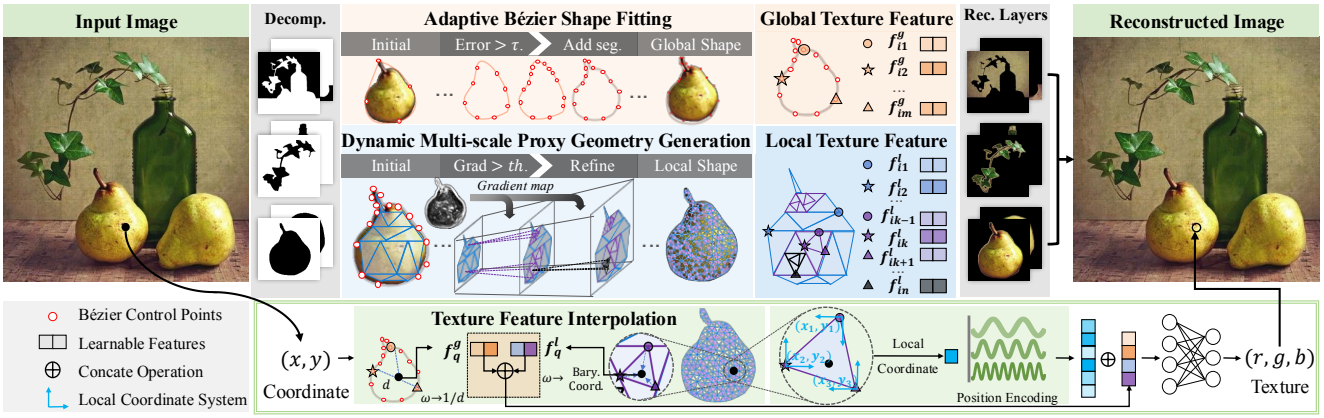

Figure 2. **Overview of our framework.** We propose a novel vectorized image representation to embed multi-scale image texture features on multi-layer hierarchically related geometric control points. With such representation, the texture at arbitrary position in the continuous image domain can be decoded with a geometry-aware interpolation method and a lightweight decoding function.

between $\mathbb{E}_i$ and $\mathbb{B}_i$ to optimize control point positions:

$$\mathcal{L}_i^{cd} = \sum_{\boldsymbol{b}\in\mathbb{B}_i} \min_{\boldsymbol{e}\in\mathbb{E}_i} \|\boldsymbol{b}-\boldsymbol{e}\|_2^2 + \sum_{\boldsymbol{e}\in\mathbb{E}_i} \min_{\boldsymbol{b}\in\mathbb{B}_i} \|\boldsymbol{e}-\boldsymbol{b}\|_2^2. \tag{4}$$

After optimizing for a number of epochs, we compute a per-segment error for the Bézier shape:

$$err(\boldsymbol{s}_{ij}) = \sum_t ||\boldsymbol{b}_{ij}(t) - \boldsymbol{e}_{ijt}||_2^2, \tag{5}$$

where $\boldsymbol{e}_{ijt} \in \mathbb{E}_i$ represents the boundary point that is closest to the shape point $\boldsymbol{b}_{ij}$. For each segment $\boldsymbol{s}_{ij}$ with $err(\boldsymbol{s}_{ij}) > \tau$, we sample three points $\{\boldsymbol{p}_{ij}^{11}, \boldsymbol{p}_{ij}^{12}, \boldsymbol{p}_{ij}^{13}\}$ along the segment and insert them into the control point sequence, thus splitting the segment into two segments $\boldsymbol{s}_{ij1}$ and $\boldsymbol{s}_{ij2}$, which are parameterized by control points $\{\boldsymbol{p}_{ij}^0, \boldsymbol{p}_{ij}^1, \boldsymbol{p}_{ij}^{11}, \boldsymbol{p}_{ij}^{12}\}$ and $\{\boldsymbol{p}_{ij}^{12}, \boldsymbol{p}_{ij}^{13}, \boldsymbol{p}_{ij}^2, \boldsymbol{p}_{ij}^3\}$, respectively. We repeat the above process until the error of any segment is less than τ and then obtain the optimized $\boldsymbol{B}_i$.

3.3.2. Dynamic Multi-scale Proxy Geometry Generation

Previous methods [6, 21] represent image geometry by fitting only the image edges, which not only fail to capture complex local geometry but also cannot embed intricate internal textures. To address this, we propose to place proxy triangles within shapes. Considering that uniformly distributing triangles internally struggles to balance the trade-off between representation accuracy and efficiency, we introduce an interrelated multi-scale triangulation method to capture both global and local geometric information efficiently with fewer parameters, facilitating stable shape/texture representation inside shapes.

Specifically, we first employ the Triwild algorithm [14], using the Bézier curves as boundary constraints, to triangulate the interior of the layer. During triangulation process, relative edge length l_r is a key parameter that controls the density of the generated triangles (please refer to [14] for more details). Considering that the texture of background layers tends to be smoother, while foreground layers often exhibit richer textures, we assign different base edge lengths for background and foreground layers respectively, *i.e.* l_r^b and l_r^f. Thus we obtain the *hierarchy-1* proxy geometry:

$$\boldsymbol{G}_i^1 = \{\boldsymbol{V}_i^1, \boldsymbol{T}_i^1\} = Triwild(\boldsymbol{M}_i, \boldsymbol{B}_i, l_r^*), \tag{6}$$

where $\boldsymbol{V}_i^1$ denotes the collection of the *hierarchy-1* proxy vertices and $\boldsymbol{T}_i^1$ represents the proxy triangle elements.

We believe that regions with higher geometric information density should be represented with more finer/smaller triangles. Hence, we perform dynamic multi-scale triangle refinement on the *hierarchy-1* triangle elements and generate hierarchical proxy geometric nodes. In our method, we use gradient magnitudes extracted by the Sobel filter to measure geometric information density. Specifically, we first generate a gradient map for $\boldsymbol{M}_i$. Next, for each triangle $\boldsymbol{t}_{ij}^1 \in \boldsymbol{T}_i^1$, we identify the pixels covered and generate pixel-level gradient magnitudes. If $\boldsymbol{t}_{ij}^1$ covers more than one pixel and the average gradient magnitude of $\boldsymbol{t}_{ij}^1$ exceeds a specified threshold, we refine $\boldsymbol{t}_{ij}^1$ by sampling a vertex on each of its three edges, thus dividing it into four smaller triangles. Generally, we set the threshold as the average gradient magnitude within $\boldsymbol{M}_i$. Performing the above operation in parallel for all triangles yields the *hierarchy-2* proxy geometric nodes, which can be described as:

$$\boldsymbol{G}_i^2 = \{\boldsymbol{V}_i^2, \boldsymbol{T}_i^2\} = Refine(\boldsymbol{V}_i^1, \boldsymbol{T}_i^1). \tag{7}$$

We can perform the above process recursively to obtain proxy geometric nodes with multiple hierarchical levels for each image layer $\boldsymbol{M}_i$. The generated hierarchical proxy geometry can be denoted as:

$$\boldsymbol{G}_i^* = \cup_{j=1}^{h^*} \boldsymbol{G}_i^{*,j}, \tag{8}$$

where $* \in \{b, f\}$ denotes foreground/background layers. $\boldsymbol{G}_i^{*,j} = \{\boldsymbol{V}_i^{*,j}, \boldsymbol{T}_i^{*,j}\}$ denotes the *hierarchy-j* proxy geometry for $\boldsymbol{M}_i$. In our experiments, we set $h^b = 2$ and $h^f = 3$.

3.4. Neural Texture Parameterization

To efficiently represent precise textures with fewer parameters, we follow distributed implicit representation methods [4, 6, 23] to explore image texture in a coordinate-to-color manner. In our framework, we abandon the approach that distributes texture features on fixed discrete grids or along Bézier curves which solely describe the image edges. In contrast, we distribute texture features on hierarchically interrelated geometric control points to embed multi-scale long-range structures and fine-grained texture details.

Specifically, to capture the global texture information of layer $\boldsymbol{M}_i$, we sample certain points from the boundary Bézier shape $\boldsymbol{B}_i$ as texture control points. To reduce the number of parameters and ensure a uniform distribution along the contour, we sample one point in each segment to store/embed global texture features $\boldsymbol{F}_i^g = [\boldsymbol{f}_{i1}^g, ..., \boldsymbol{f}_{im}^g]$. To capture the rich local textures within the layer, we also store/embed texture features at all the multi-scale proxy vertices (*i.e.*, $\boldsymbol{V}_i$) within the layer: $\boldsymbol{F}_i^l = [\boldsymbol{f}_{\boldsymbol{v}_i^1}^l, ..., \boldsymbol{f}_{\boldsymbol{v}_i^n}^l]$, where $\boldsymbol{v}_i^*$ means the vertices of all the hierarchical triangles mentioned in Sec. 3.3 and n denotes the number of vertices.

Consequently, we parameterize the texture information of each image as hierarchical texture feature vectors (*i.e.*, $\boldsymbol{F}_i = [\boldsymbol{F}_i^g, \boldsymbol{F}_i^l]$) embedded at geometric control points, along with a lightweight decoding function $\phi_\theta : [\boldsymbol{x}, \boldsymbol{f}_{\boldsymbol{x}}] \mapsto \boldsymbol{c}$ to map features to texture values, where $\boldsymbol{x}$ is the queried coordinate and $\boldsymbol{f}_{\boldsymbol{x}}$ denotes the corresponding feature vector at the queried point, which can be obtained through a geometry-aware interpolation method. Specifically, for arbitrarily positioned query point in $\boldsymbol{M}_i$ with coordinate $\boldsymbol{x}$, we follow [6] to use inverse distance weighting method to interpolate its global texture feature from features stored at Bézier control points:

$$\boldsymbol{f}_{\boldsymbol{x}}^g = \sum_{j=1}^{m} \frac{w(\boldsymbol{x}, \boldsymbol{p}_{ij}^s)}{\sum_{j=1}^{m} w(\boldsymbol{x}, \boldsymbol{p}_{ij}^s)} \boldsymbol{f}_{ij}^g, \tag{9}$$

where $\boldsymbol{p}_{ij}^s$ means the texture control point sampled from the j-th segment. $w(\cdot, \cdot)$ is the inverse distance weighting function. For local texture features, we first use the barycentric coordinate algorithm to locate the triangle element $\boldsymbol{t}_{ij} = [\boldsymbol{v}_{ij}^1, \boldsymbol{v}_{ij}^2, \boldsymbol{v}_{ij}^3]$ that covers point $\boldsymbol{x}$ within all the hierarchical proxy triangles (*i.e.*, $\boldsymbol{T}_i$). We then use barycentric coordinate interpolation to obtain the local texture feature of point $\boldsymbol{x}$:

$$\boldsymbol{f}_{\boldsymbol{x}}^l = \sum_{k=1}^{3} \lambda_{\boldsymbol{x}}^{ij,k} \boldsymbol{f}_{\boldsymbol{v}_{ij}^k}^l, \tag{10}$$

where $(\lambda_{\boldsymbol{x}}^{ij,1}, \lambda_{\boldsymbol{x}}^{ij,2}, \lambda_{\boldsymbol{x}}^{ij,3})$ denotes the barycentric coordinate vector of point $\boldsymbol{x}$ with respect to triangle element $\boldsymbol{t}_{ij}$ and $\boldsymbol{f}_*^l$ means the local feature vectors at each vertex of $\boldsymbol{t}_{ij}$. Thus, for any query point $\boldsymbol{x}$, we can interpolate its texture features: $\boldsymbol{f}_{\boldsymbol{x}} = [\boldsymbol{f}_{\boldsymbol{x}}^g, \boldsymbol{f}_{\boldsymbol{x}}^l]$. Then the texture value at point $\boldsymbol{x}$ can be predicted through a decoding function ϕ_θ.

Thanks to hierarchical texture embedding on both edge control points and multi-scale proxy geometric nodes within shapes, our implicit texture embedding is highly robust. Also, the texture of each layer is represented solely by the features stored at its own geometric points. Such layer-wise strategy significantly accelerates the convergence of texture parameters, and facilitates efficient image editing.

3.5. Representation Parameters Optimization

For geometric parameters, the coordinates of Bézier control points and multi-scale triangles are obtained as stated in Sec. 3.3. For texture parameters, we follow [4, 6] to treat each pixel $[i, j]$ as a coordinate $\boldsymbol{x}_{ij}$ in normalized continuous image space. For each point, we search its corresponding layer according to the layer order defined in Eqn. 2 and then obtain its hierarchical features $\boldsymbol{f}_{\boldsymbol{x}_{ij}}$ as described in Sec. 3.4. It is noted that we only need to assess the positional relationship between the pixel coordinates and the Bézier shapes, eliminating the need to store a binary mask for each layer. Then we can predict the texture value at pixel $[i, j]$ with the decoding function ϕ_θ. To capture high-frequency texture details, we follow [22] to incorporate position encoding at the input. Additionally, to avoid fitting texture information to absolute coordinates and to maintain hierarchical editability of the image, we encode the local coordinates of each pixel in the corresponding triangle:

$$\widetilde{\boldsymbol{x}}_{ij} = \mathcal{U}([\boldsymbol{x}_{ij} - \boldsymbol{v}_{\boldsymbol{x}_{ij}}^1, \boldsymbol{x}_{ij} - \boldsymbol{v}_{\boldsymbol{x}_{ij}}^2, \boldsymbol{x}_{ij} - \boldsymbol{v}_{\boldsymbol{x}_{ij}}^3]), \tag{11}$$

where $\mathcal{U}$ denotes the parameter-free encoding function as defined in [22]. $\boldsymbol{v}_{\boldsymbol{x}_{ij}}^*$ means the vertices of the triangle that covers $\boldsymbol{x}_{ij}$ and $[\cdot, \cdot]$ denotes concatenation. Hence the output image $\hat{\boldsymbol{I}}$ can be described as:

$$\hat{\boldsymbol{I}}[i, j] = \phi_\theta([\widetilde{\boldsymbol{x}}_{ij}, \boldsymbol{f}_{\boldsymbol{x}_{ij}}]). \tag{12}$$

Then we can use pixel-wise error to optimize the texture features of all image layers (*i.e.*, $\boldsymbol{F} = [\boldsymbol{F}_1, ..., \boldsymbol{F}_L]$) and the parameters θ of the decoding function simultaneously. The optimization objective can be denoted as:

$$\min_{\{\boldsymbol{F}, \theta\}} ||\boldsymbol{I} - \hat{\boldsymbol{I}}||_2^2. \tag{13}$$

After parameter optimization, we can efficiently perform shape and texture editing for natural images with complex texture, which are not achieved by current SOTAs [6, 12, 21]. For instance, we can perform image shape editing by manipulating the Bézier control points (*e.g.*, translating, rotating and scaling). We then map the pixels covered by the transformed shapes back to the coordinate space of corresponding geometric nodes to interpolate texture features, which effectively ensures in-shape texture consistency. Additionally, we can achieve texture transfer from $\boldsymbol{M}_j$ to $\boldsymbol{M}_i$ by mapping the pixels of $\boldsymbol{M}_i$ to the coordinate space of geometric nodes of $\boldsymbol{M}_j$ for new texture feature interpolation, while ensuring the consistency of structure inside $\boldsymbol{M}_i$. Please refer to Sec. 4.3 for more details.

4. Experiments

Datasets. Most image vectorization methods [5, 19, 25] are limited to images with extremely simple textures like Icon [2] and Emojis [1]. Recent works [6, 11, 21] extend their methods to Clipart images with slightly more complex textures. SuperSVG [12] utilizes the natural images of ImageNet [8] for training and evaluation. We compare our method with SOTAs on all the datasets mentioned above.
Implementation Details. We initialize $4-$segment Bézier shape for each layer, *i.e.*, $m = 4$. During adaptive Bézier shape fitting, we optimize for a total of 2000 epochs and calculate the per-segment error every 500 epochs. We set the error threshold $\tau = 2e - 3$. For Triwild algorithm, we set base edge lengths as $l_r^b = 0.2, l_r^f = 0.05$. We set the dimension of texture features as 16, the dimension of coordinate embedding as 66, and the ϕ_θ is a two-layer MLP with input dimension 82, hidden dimension 128 and output dimension 3. We optimize the texture parameters for 2000 epochs and the entire pipeline takes an average of 5 minutes for each natural image on ImageNet.

4.1. Comparison with Image Vectorization Methods

We compare our method with current SOTA methods (*i.e.*, DVG [19], LIVE [21], O&R [11], S-SVG [12] and Chen *et al*. [6]) on image vectorization task. For Emojis&Icon datasets, we use *MSE* distance as the metric. For Clipart and ImageNet benchmark, we use *MSE*, *PSNR*, *LPIPS* and *SSIM* as metrics. Note that S-SVG [12] only releases part of its source code, thus we directly use the best results reported in their paper (4000 shapes). In addition, increasing the shape number has only a limited effect on the performance of Chen *et al*. [6] (as demonstrated in their paper), thus we use their 512-shape version considering their GPU memory usage and optimization time. For other compared methods, we use 512 shapes on Clipart and 4000 shapes on ImageNet. Quantitative comparisons are shown in Tab. 1 and Tab. 2. We can observe that our method significantly outperforms other methods across all metrics. In particular, existing methods are significantly short in natural image reconstruction compared to ours, which can be attributed to that previous approaches either stack shapes [11, 12, 19, 21] to represent textures or encode textures into sparse edge control points [6]. In contrast, our method employs multi-layer multi-scale interrelated geometric control points to encode both global information and local texture details, achieving superior reconstruction quality. We also show some qualitative comparisons in Fig. 3. We can see that Chen *et al*. [6] can only reconstruct very smooth textures and fail to capture complex texture details due to lack in texture control points within shapes (*e.g.*, the intricate texture of eyes and bird's feathers). Our method can capture intricate and complex textures thanks to hierarchical texture embedding.

Dataset	DVG	Im2Vec	LIVE	O&R	Chen	**Ours**
Emoji	9.24	25.81	1.61	1.44	0.63	**0.12**
Icon	28.52	32.90	2.42	2.23	0.70	**0.11**

Table 1. **Image vectorization results on Emoji&Icon datasets.** *MSE* ($\times 10^{-3}$) results are reported. All compared methods use 10 shapes. Our method achieves significantly better results.

Dataset	Method	Time	MSE↓	PSNR↑	LPIPS↓	SSIM↑
Clipart.	DVG	7.5	2.18	26.60	0.2998	0.8542
	LIVE	90.2	1.92	27.17	0.2633	0.8742
	O&R	10.9	1.64	27.85	0.2599	0.8739
	Chen	29.2	0.55	32.59	0.2109	0.9182
	Ours	4.3	**0.17**	**37.70**	**0.0835**	**0.9590**
ImageNet.	DVG	89.8	2.63	25.80	0.3823	0.8459
	LIVE	>1000	2.91	25.36	0.4076	0.8443
	O&R	81.3	2.98	25.26	0.4058	0.8464
	S-SVG	-	1.40	29.96	0.2496	0.9028
	Chen	34.3	1.56	28.07	0.2593	0.8717
	Ours	5.2	**0.37**	**34.32**	**0.1082**	**0.9333**

Table 2. **Image vectorization results on Clipart and ImageNet.** *MSE* ($\times 10^{-3}$) and other image reconstruction metrics are reported. Our method achieves best results across all metrics, especially on complex natural images. S-SVG is based on deep network training, thus we do not compare its running time. The time (*mins*) is tested on an NVIDIA GeForce RTX 3090 GPU.

4.2. Comparison with Implicit Representations

We compare our method with implicit image representation algorithms (*i.e.*, SIREN [27], Grid-based method and Chen *et al*. [6]) on the ImageNet dataset. We use $3-$layer and $5-$layer version of SIREN for comparison. For grid-based method, we follow [6] to utilize the framework of LIIF [4] to perform zero-shot image reconstruction, which stores feature vectors on fixed grids and decodes the features with a $2-$layer MLP. We demonstrate the effectiveness and efficiency of our method by comparing reconstruction quality and the number of implicit parameters on the ImageNet dataset. Quantitative results are reported in Tab. 3. The results indicate that our method achieves significantly higher reconstruction metrics than counterparts with a highly compressed parameter count. Notably, compared to Chen *et al*. [6], although we place many proxy geometric points within each shape, our layer-wise strategy to model each image layer as a single shape allows us to save a substantial number of Bézier control points (Chen *et al*. [6] need several hundreds of $4-$segment shapes for a natural image while our method needs only a few shapes/image layers). In Fig. 4, we present a visual comparison with methods that have similar number of parameters to ours. SIREN [27] exhibits noticeable blurring and a lack of high-frequency details when the number of MLP layers is insufficient as it fits all information into MLP parameters, disregarding the explicit image structure and texture distribution. For grid-based method that stores features on fixed grids, a large number of features is needed to achieve acceptable

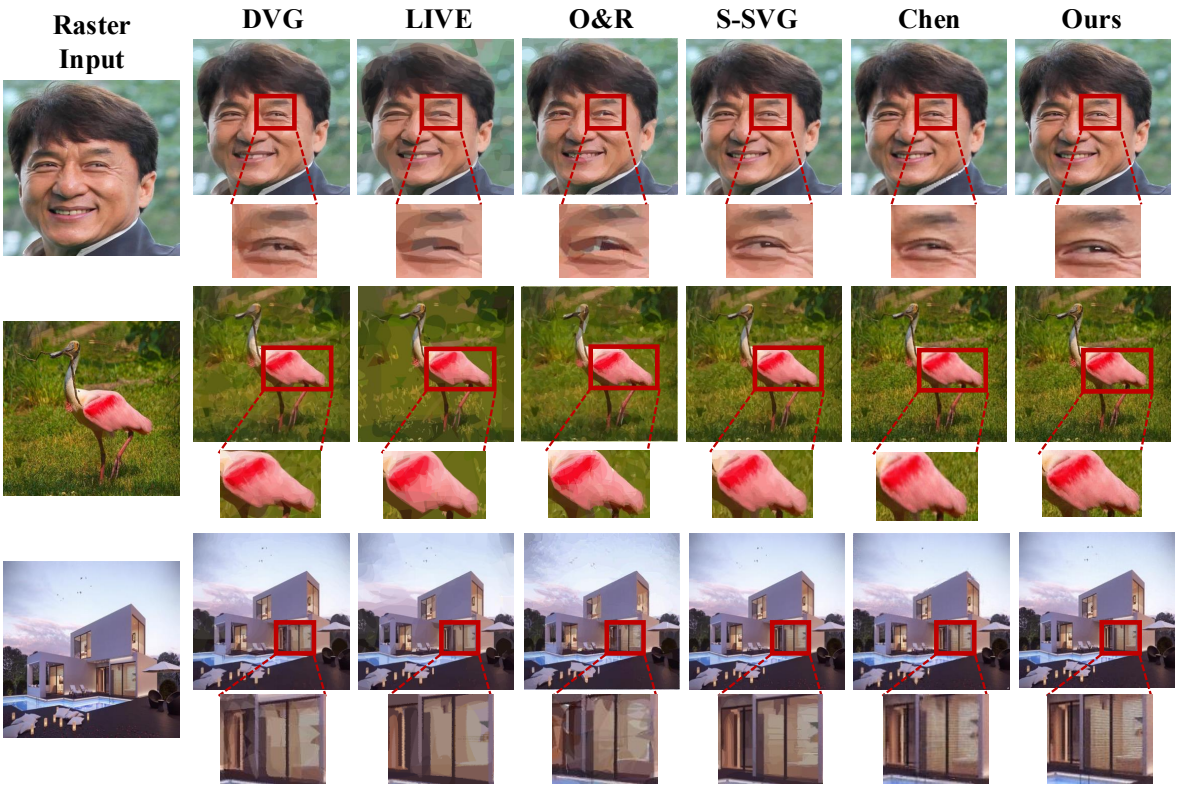

Figure 3. **Qualitative comparison on natural images.** We use red boxes to emphasize the differences. Our representation can express complex image details. The results of S-SVG [12] are directly obtained from their paper. Please zoom in for more details.

Method	MSE↓	PSNR↑	LPIPS↓	SSIM↑	Codes.↓	Params.↓
SIREN-3	2.50	26.02	0.2773	0.8267	-	50307
SIREN-5	1.70	27.70	0.2605	0.8688	-	83331
LIIF-/8	6.58	21.82	0.5059	0.6347	4096	68099
LIIF-/4	3.10	25.09	0.2920	0.8241	16384	264707
Chen-256	1.64	27.85	0.2595	0.8692	3072	58115
Chen-512	1.56	28.07	0.2593	0.8717	6144	113411
Ours	**0.37**	**34.32**	**0.1082**	**0.9333**	**2106**	**48919**

Table 3. **Comparison with general implicit image representations.** *MSE* ($\times 10^{-3}$) and other reconstruction results on ImageNet benchmark are reported. "Codes" denotes the number of feature vectors utilized. "Params" denotes the number of parameters. "SIREN" does not distribute feature vectors. "SIREN-$*$" means the SIREN version with $*$ layers. In "LIIF-/$*$", the "$*$" denotes the proportion of the spatial dimension of the original image to the grids storing feature vectors as defined in [6].

reconstruction results, which is very inefficient. Chen *et al*. [6] fail to model intricate texture details due to the sparse texture embedding. Our method achieves high parameter compression while preserving high reconstruction quality by utilizing multi-layer and multi-scale texture embedding.

4.3. Easy Image Editing

Easy image editing for natural images is a very challenging task in image vectorization area. Current methods such

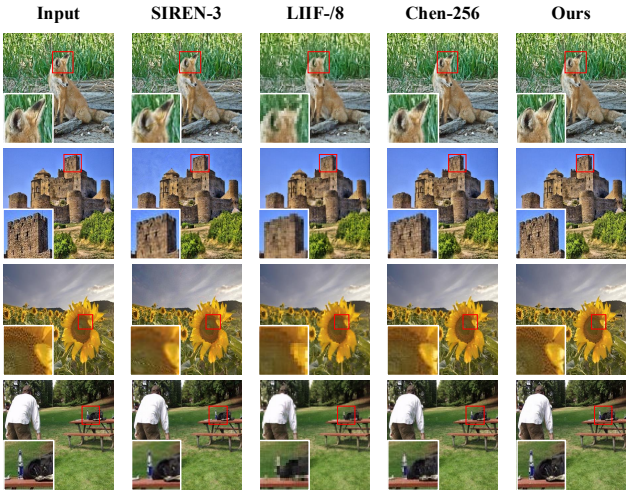

Figure 4. **Qualitative comparison on natural images.** We visualize examples of methods with the same order of magnitude of parameters. Please zoom in for more details.

as LIVE [21] and SuperSVG [12] lack texture modeling, requiring extensive shape stacking to represent natural images, which makes it impossible to precisely edit specific areas of interest. Chen *et al*. [6] decouple image geometry and texture and embed texture codes into image edge points, enabling the editing of simple images such as Emo-

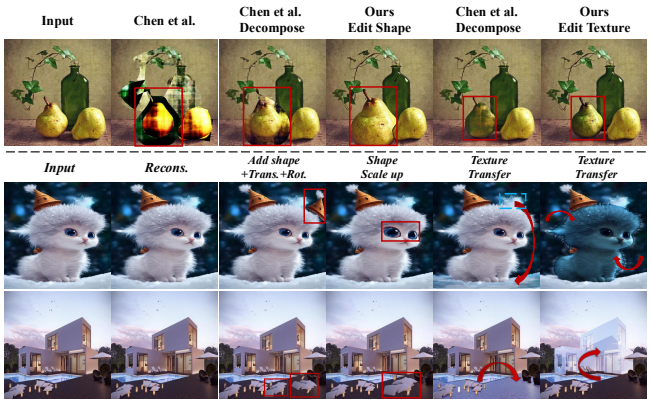

Figure 5. **Some examples on easy image editing task.** We compare editing results with Chen *et al*. [6] above the dashed line. Chen *et al*. [6] suffer from significant texture distortion during editing. We also show some examples of ours below the dashed line. We use red boxes to indicate the edited shapes and red arrows to indicate texture transfer. Please zoom in for details.

jis [1] and Icons [2]. In this section, we compare with [6] on image editing task and the results are shown in Fig. 5. We observe that the original method of [6] also results in shape stacking, making editing unreasonable. For fair comparisons, we modify the pipeline of [6] by first segmenting the region of interest and then fitting it individually with a single shape. We then transform shapes by adjusting control points (columns 3 and 4) and transfer textures by migrating features (columns 5 and 6) as depicted in Sec. 3.5. We can see that [6] results in severe texture distortion when shapes are deformed and does not support precise texture transfer between different shapes, which can be attributed to the sparse texture embedding and distance-based interpolation, which results in a discontinuous coordinate-to-texture mapping. In contrast, our method allows for easy shape editing while preserving detailed texture consistency and smooth texture transfer between shapes because of the stable texture representation with multi-layer multi-scale texture embedding and the layer-wise Bézier fitting strategy.

4.4. Ablation Study

Component Analyses. We first conduct experiments to explore the effectiveness of the crucial modules in the proposed framework. The results are shown in Tab. 4. We can see that when using only edge control points without internal triangles ("Edge-only"), reconstruction results on natural images deteriorate significantly. Employing a single level of internal triangles within shapes yields competitive results with relatively few parameters. Utilizing multi-scale triangles further enhances reconstruction quality with only a minimal increase in parameters, which demonstrates the effectiveness of our multi-scale scheme. We also conduct the ablated version without Bézier fitting, where we directly perform Trilwild algorithm on the entire image ("Tri-only"). With a similar parameter count, this version exhibits reduced reconstruction quality on natural images. This demonstrates the effectiveness of our layer-wise strategy and multi-level feature fusion at both shape edges and interiors. We also demonstrate the effectiveness of the position encoding scheme and the interpolation method based on barycentric coordinates, which further enhance our method.

	Edge -only	Edge+ SinTri	Edge+ MulTri	Tri -only	w/o Pos Enc.	IDW -Tri
MSE	1.13	0.54	**0.37**	0.65	0.93	0.43
Params.	15097	36067	48919	34501	40727	48919

Table 4. **Component analyses on feature embedding methods.** "SinTri" and "MulTri" denotes single-scale triangles and multi-scale triangles respectively. "IDW-Tri" means using inverse distance weighting interpolation within triangles. *MSE* ($\times 10^{-3}$) results and parameter counts on ImageNet are reported.

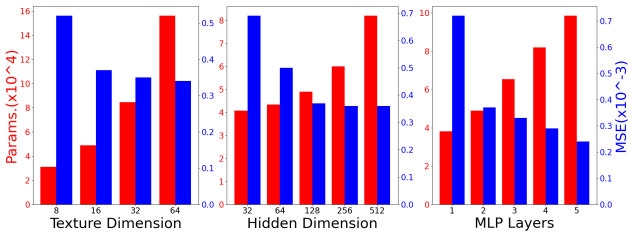

Figure 6. **Parameter analyses.** *MSE* ($\times 10^{-3}$) results (blue) and parameter counts (red) on ImageNet are reported.

Parameter Analyses. We investigate how the texture feature dimension, the hidden dimension and the number of MLP layers affect the reconstruction results. The results are shown in Fig. 6. We observe that, due to our efficient multi-layer multi-scale texture feature embedding, the improvement in reconstruction quality from higher texture feature dimensions or higher hidden dimensions is minimal. Additional MLP layers further enhance the performance of our method. Considering the trade-off between reconstruction quality and parameter counts, using a two-layer MLP proves to be highly efficient.

5. Conclusion

This work presents a novel efficient image representation to parameterize any image into layer-wise texture embeddings distributed at multi-layer geometric control points. Extensive experiments on various tasks and benchmarks demonstrate that our representation effectively reconstructs complex natural images with significant parameter compression, while enabling precise and easy image editing.

6. Acknowledgment

This work was supported by National Science Foundation of China (U20B2072). This work was also partially supported by Grant YG2021ZD18 from Shanghai Jiaotong University Medical Engineering Cross Research. This work was partially supported by STCSM 22DZ2229005.

References

[1] Note emoji. https://github.com/googlefonts/noto-emoji. Accessed: 2021-09-30. 6, 8

[2] creativestall. https://thenounproject.com/creativestall/. 6, 8

[3] Jean-Daniel Boissonnat, Olivier Devillers, Monique Teillaud, and Mariette Yvinec. Triangulations in cgal. In *Proceedings of the sixteenth annual symposium on Computational geometry*, pages 11–18, 2000. 3

[4] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021. 1, 2, 5, 6

[5] Ye Chen, Bingbing Ni, Xuanhong Chen, and Zhangli Hu. Editable image geometric abstraction via neural primitive assembly. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23514–23523, 2023. 2, 3, 6

[6] Ye Chen, Bingbing Ni, Jinfan Liu, Xiaoyang Huang, and Xuanhong Chen. Towards high-fidelity artistic image vectorization via texture-encapsulated shape parameterization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15877–15886, 2024. 1, 2, 3, 4, 5, 6, 7, 8

[7] Yinbo Chen, Oliver Wang, Richard Zhang, Eli Shechtman, Xiaolong Wang, and Michael Gharbi. Image neural field diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8007–8017, 2024. 1, 2

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[9] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[10] 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

[11] 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, 6

[12] 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. 1, 2, 3, 5, 6, 7

[13] Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Tetrahedral meshing in the wild. *ACM Trans. Graph.*, 37(4):60, 2018. 3

[14] Yixin Hu, Teseo Schneider, Xifeng Gao, Qingnan Zhou, Alec Jacobson, Denis Zorin, and Daniele Panozzo. Triwild: robust triangulation with curve constraints. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019. 3, 4

[15] Zhangli Hu, Ye Chen, Zhongyin Zhao, Jinfan Liu, Bilian Ke, and Bingbing Ni. Towards artist-like painting agents with multi-granularity semantic alignment. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 10191–10199, 2024. 1

[16] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002. 3

[17] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, 2023. 2, 3

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

[19] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020. 1, 2, 6

[20] Yaron Lipman. Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics (TOG)*, 31 (4):1–13, 2012. 3

[21] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16314–16323, 2022. 1, 2, 3, 4, 5, 6, 7

[22] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 5

[23] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)*, 41(4):1–15, 2022. 1, 2, 5

[24] 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. 1, 2

[25] 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, 3, 6

[26] Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Workshop on applied computational geometry*, pages 203–222. Springer, 1996. 3

[27] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural*

information processing systems, 33:7462–7473, 2020. 1, 2, 3, 6

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

[29] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 1, 2

[30] Zhenyu Wang, Jianxi Huang, Zhida Sun, Daniel Cohen-Or, and Min Lu. Layered image vectorization via semantic simplification. *arXiv preprint arXiv:2406.05404*, 2024. 1, 2, 3

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

[32] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10371–10381, 2024. 2, 3

[33] Zhongyin Zhao, Ye Chen, Zhangli Hu, Xuanhong Chen, and Bingbing Ni. Vector graphics generation via mutually impulsed dual-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4420–4428, 2024. 1