Vector Scaffolding: Inter-Scale Orchestration for Differentiable Image Vectorization

Jaerin Lee[1], Kanggeon Lee[1], and Kyoung Mu Lee[1,2]

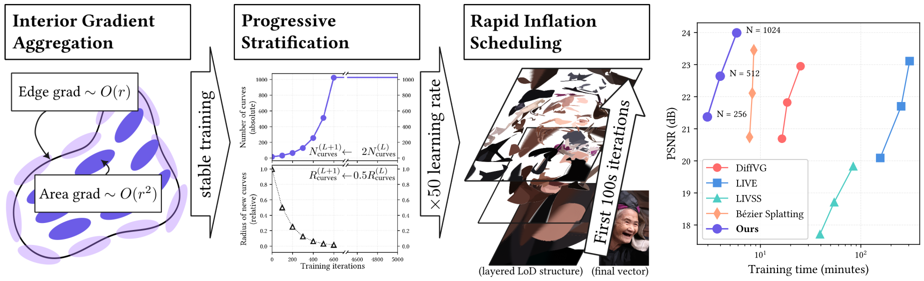
[1] Dept. of ECE&ASRI, Seoul National University, Korea
[2] IPAI, Seoul National University, Korea
{ironjr,dlrkdrjs97,kyoungmu}@snu.ac.kr

Abstract. Differentiable vector graphics have enabled powerful gradient-based optimization of vector primitives directly from raster images. However, existing frameworks formulate this as a flat optimization problem, forcing hundreds to thousands of randomly initialized curves to blindly compete for pixel-level error reduction. This disordered optimization leads to *topology collapse*, where macroscopic structures are distorted by internal high-frequency noise, resulting in a redundant and uneditable "polygon soup" that limits practical editability. To address this limitation, we propose **Vector Scaffolding**, a novel hierarchical optimization framework that shifts from flat pixel-matching to structured topological construction tailored for vector graphics. By identifying a key cause of topology collapse as the mathematical imbalance between area and boundary gradients, we introduce *Interior Gradient Aggregation* to stabilize the learning dynamics of multi-scale curve mixtures. Upon this stabilized landscape, we employ *Progressive Stratification* and *Rapid Inflation Scheduling* to progressively densify vector primitives with extremely high learning rates ($\times 50$). Experiments demonstrate that our approach accelerates optimization by $2.5\times$ while simultaneously improving PSNR by up to 1.4 dB over the previous state of the art.

Keywords: Differentiable vector graphics · hierarchical optimization · structure alignment · Bézier curves · Gaussian splatting · acceleration

1 Introduction

Raster images capture *what we see*; vector graphics encode *how we understand it.* Because they are resolution-independent and inherently editable through professional creative software relying on path-based tools, vector graphics have become indispensable for digital art, animation, and modern design workflows. Recent advancements in differentiable vector graphics [13, 14, 17, 23] bridge the gap between these two distinct domains. Their central idea is to allow gradients to flow through the rasterization process, enabling gradient-based optimization algorithms to directly synthesize or reconstruct vectors from pixel supervision. While pioneering works like DiffVG [13], and later LIVE [17], established the mathematical foundation for this task, their prohibitive computational costs make them impractical for real-world applications.



(a) High-level concept of Vector Scaffolding. **(b)** DIV2K benchmark

Fig. 1: Overview. We introduce **Vector Scaffolding**, a hierarchical optimization framework for fast and stable differentiable image vectorization. (a) Our framework consists of three ideas: *Interior Gradient Aggregation* stabilizes training vectors with large scale variance. *Progressive Stratification* further stabilizes training to enable huge learning rate. This allows *Rapid Inflation Scheduling* to accelerate training and outperform baselines by a large margin. (b) We achieve higher rendering fidelity in a fraction of wall clock time required by existing methods.

The slow speed of these early works is due to the sequential reconstruction of vectors, curve by curve. Bézier Splatting [14] introduced a parallelized optimization scheme for vectorization. A highly optimized differentiable rasterization pipeline powered by 2D Gaussian splatting [10, 12] along Bézier curves significantly accelerates the optimization process. However, it still inherits the *flat optimization problem* from the early works, where all curves are treated uniformly. Initializing hundreds of curves simultaneously on a single plane and letting them compete for per-pixel local loss reduction leads to disordered optimization dynamics. At best, this leads to a superpixel-like representation with low editability and representational fidelity. At worst, it results in an unstructured, uneditable "polygon soup," blocking creators from further editing and polishing the artwork. We name this phenomenon *topology collapse*. This undermines the key advantage of vector representation: editability.

Natural *and* artistic images, on the other hand, have an inherent hierarchy of scales and details. Later works like LIVSS [23] and Optimize & Reduce [8] were the first to leverage this hierarchical structure. Nevertheless, these methods depend heavily on external image priors such as diffusion models [9] to build the image hierarchy. The computational cost introduced by these priors fundamentally limits the scalability of the frameworks.

In our **Vector Scaffolding**, we instead propose a purely on-the-fly optimization-based framework for hierarchical vectorization, without relying on learned external priors. Following the wisdom of how artists create artwork from simple sketches and gradually add details, we construct vector graphics by progressively spawning smaller curves above base constructions. We observe that the commonly used splatting formulation [14] emphasizes boundary samples and omits positional gradients from interior samples. Our work starts with **Interior Gradient Aggregation** to add the missing gradients. This allows large base curves to be effectively pulled from their interior, greatly stabilizing the optimization process. We then establish a structural hierarchy through **Progressive Stratification**, inspired by the natural power law of image frequency. Once lower-frequency base layers stabilize, we stack a new layer with a larger number of smaller curves in places where the residual error concentrates. Empirically, our spawning rule can support multiple levels of detail without suffering from curve redundancy. The **Progressive Stratification** rule also prevents upper-layer high-frequency errors from distorting the lower layers, further stabilizing the overall optimization trajectory. This allows us to increase the learning rate by a factor of 50 compared to the previous state of the art [14], without causing instability. Finally, based on this fast optimization trajectory, we redesign the scheduling to further accelerate convergence. Our **Rapid Inflation Scheduling** strategy allocates new curves in the first few hundred iterations safely, while the rest of the iterations finalize the optimization process.

By restructuring the optimization dynamics, our *Vector Scaffolding* accelerates the entire optimization process by $2.5\times$ in wall-clock time compared to the state-of-the-art Bézier Splatting [14], with improved reconstruction fidelity by up to 1.4 dB in PSNR. The results provide strong empirical evidence that optimization structure is crucial in achieving good performance in vectorization.

In summary, our main contributions are as follows:

- We propose **Vector Scaffolding**, a novel hierarchical optimization framework for differentiable image vectorization that shifts from flat pixel-matching to structured construction for fast and better vectorization.
- We introduce **Interior Gradient Aggregation** to match the optimization target to the Reynolds transport theorem, greatly stabilizing the optimization process.
- We formulate **Progressive Stratification** with **Rapid Inflation Scheduling**, establishing a structured hierarchy that allows aggressive learning rates without causing instability.
- Experiments demonstrate that our method achieves a $2.5\times$ speedup in optimization measured in wall-clock time and up to 1.4 dB PSNR improvement at identical curve budgets compared to the state-of-the-art Bézier Splatting [14].

2 Related Work

2.1 Differentiable Vector Graphics and Optimization

Image vectorization has been significantly advanced by differentiable rendering techniques. DiffVG [13] introduced the first differentiable vector graphics (VG) rasterizer, enabling gradient-based optimization of vector primitives through the rasterization process. Building upon this foundation, subsequent works such as LIVE [17], Optimize & Reduce (O&R) [8], and layered vectorization approaches [23] explored layer-wise path initialization and hierarchical optimization strategies to better preserve image topology during vectorization. Other works focused on improving color representation within vector graphics, for example by modeling regional color distributions using linear gradients [4] or implicit neural representations [3]. Despite their effectiveness, these approaches often incur substantial computational overhead due to the complexity of differentiable rasterization [13, 17] or introduction of external priors [23]. As a result, optimizing vector representations for high-resolution images can remain computationally expensive.

Another line of work learns neural models to directly synthesize vector graphics from raster inputs. Early approaches such as Lopes et al. [15] adopt encoder–decoder architectures for vector generation. Subsequent works including Im2Vec [20] and SVGFormer [2] further improve structured vector synthesis by introducing sequential modeling and transformer-based architectures. More recently, diffusion-based generative models have been explored for vector graphics synthesis and text-to-VG generation [9,11,26,27,29]. While these learning-based approaches show promising results for stylized graphics such as icons or sketches, they often face challenges when applied to complex images in professional domains.

Recently, Bézier Splatting [14] was proposed to accelerate differentiable vectorization by combining Bézier curves with the efficient Gaussian splatting rendering pipeline [10,12]. Although this substantially improves rendering efficiency, the optimization process remains largely flat, with all curves optimized simultaneously under a single pixel-level objective. Such dynamics may lead to unstable geometric structures, as curves can adapt to high-frequency textures instead of capturing semantically meaningful boundaries, or compete with each other for the same pixel-level residuals. Our work addresses this limitation by introducing a hierarchical optimization strategy [8,23] that separates coarse structural geometry from fine-grained details, but without external priors. Unlike prior layered methods that sequentially add paths [17] or rely on external diffusion priors [23], we directly stabilize parallel optimization of closed Bézier curves without any external model. This simple framework yields an order-of-magnitude speedup over previous hierarchy-based methods [23] with +1 dB PSNR improvements as shown in Figure 1b and Table 2.

2.2 Gaussian Splatting and Representation Learning

3D Gaussian Splatting (3DGS) [12, 33] has recently emerged as an efficient explicit representation for neural rendering and novel view synthesis. Its differentiable tile-based rasterization pipeline enables high-fidelity rendering with real-time performance. The efficiency and flexibility of this representation have inspired numerous extensions, including dynamic scene modeling [16, 24], generative 3D content creation [22,28], and geometric reconstruction using alternative primitives such as 2D Gaussians [6, 10] or tetrahedral meshes [7].

Beyond 3D scene representation, Gaussian-based primitives have also been explored for 2D image modeling. Works such as GaussianImage [31] and Image-GS [32] demonstrate that collections of 2D Gaussians can serve as compact and expressive alternatives to raster images or implicit neural representations [18,21].

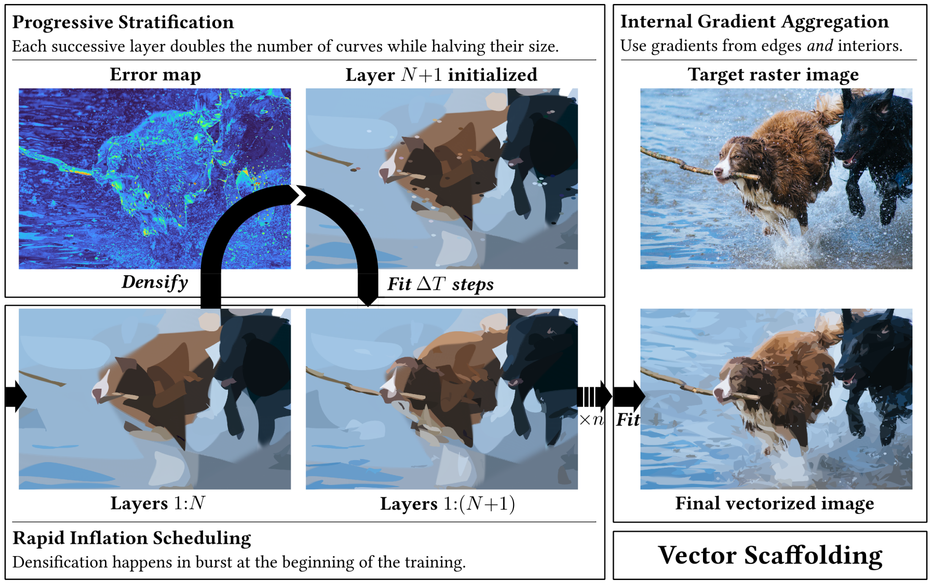


Fig. 2: Algorithm visualized. (tr) *Interior Gradient Aggregation*: Optimization is stabilized by aggregating internal area gradients alongside boundary gradients via the Reynolds transport theorem. (tl) *Progressive Stratification* aligns vector representation with the natural power law of image frequency, enabling extremely high learning rates without instability. (bl) *Rapid Inflation Scheduling*: The vector representation *inflates* in the first few hundred iterations. (br) *Vector Scaffolding Algorithm*: These components work together to achieve the final vector representation.

However, directly transferring splatting-based optimization to vector graphics introduces additional challenges due to the geometric constraints of vector primitives such as closed Bézier curves. Naively densifying primitives to minimize pixel-wise reconstruction errors can produce redundant curve sets and reduce the editability of the resulting vector graphics. In contrast, our method adopts a residual-guided adaptive densification strategy together with hierarchical scale control, enabling compact vector representations while maintaining structural coherence.

3 Method

3.1 Preliminaries

The goal of image vectorization is to construct a vector-based representation $\mathcal{B} = \{\mathcal{B}_i\}_{i=1}^{N} \in V_N$ of N vector primitives from a raster image $\mathcal{I} \in \mathbb{R}^{C\times H\times W}$. We define a differentiable renderer $g : \mathcal{B} \mapsto \hat{\mathcal{I}}$ that produces the corresponding raster image $\hat{\mathcal{I}}$ from the vector representation $\mathcal{B}$. The problem of image vectorization can then be formulated as a minimization problem:

$$\mathcal{B}^* \;=\; \arg\min_{\mathcal{B}\in V_N} \mathcal{L}_2(g(\mathcal{B}), \mathcal{I}) + \mathcal{L}_{\text{reg}}(\mathcal{B}), \tag{1}$$

where $\mathcal{L}_2$ is the pixel-wise MSE loss in the raster domain and $\mathcal{L}_{\text{reg}}$ is a collection of shape regularization terms that prevent vector primitives from forming unnatural pathological shapes. The regularization strategy is described in Section 3.5.

We employ closed Bézier curves as our internal representation and adopt the splatting-based differentiable rasterizer backend of Bézier Splatting [14]. Formally, a Bézier curve $\mathcal{B}(t)$ of degree M is defined by a Bernstein polynomial:

$$\mathcal{B}(t) \;=\; \sum_{j=0}^{M} P_j B_j^M(t), \quad t \in [0, 1], \tag{2}$$

where $B_j^M(t)$ is the j-th Bernstein polynomial of degree M and P_j is the j-th control point in the 2D image space. We treat each curve as a closed loop by linearly interpolating between the first and last control points, and assign boundary samples the same color and opacity as the filled region. That is, we additionally have a curve color $\mathbf{c}_i$ and an opacity α_i parameter for each curve $i \in \{1, 2, \dots, N\}$. To rasterize these curves, we perform α-blending of 2D Gaussians [12] sampled along the boundary and interior of the curves, following the approach of Bézier Splatting [14]. Each Gaussian is parameterized with an anisotropic covariance matrix $\Sigma = RSS^T R^T$, where the rotation matrix R is aligned with the local tangent to the curve, and the scale matrix $S = \text{diag}(\sigma_x, \sigma_y)$ is analytically derived from the distances between adjacent sampled points. The α-blending requires an additional layer depth parameter z_i for each curve for consistent occlusion. That is, at each pixel v, its color $C[v]$ is computed as:

$$C[v] \;=\; \sum_{i\in V[v]} \mathbf{c}_i \alpha_i[v] \prod_{j\in V[v], z_j < z_i} (1 - \alpha_j[v]), \tag{3}$$

where $V[v]$ is the set of curves that cover the pixel v, and $\alpha_i[v]$ denotes the rasterized opacity contribution of curve i at pixel v. This defines the differentiable rasterizer $g(\mathcal{B})$.

Unlike 3DGS [12], determining the depth parameter z_i in this 2D scenario is not straightforward. For example, Bézier Splatting [14] dynamically reassigns the depth parameter z_i during optimization by sorting curves by their bounding box sizes. With unstructured, random initialization of the primitives and maximally fair competition for per-pixel loss reduction, this leads to unstable, jittering optimization dynamics. The curves become heavily redundant and unnaturally shrink, overlap, or twist to escape local minima, resulting in a highly uneditable, superpixel-like representation similar to *polygon soup* in 3D graphics. We refer to this failure mode as *topology collapse*, where curves lose coherent structural roles and degenerate into redundant, locally optimized fragments. Our Vector Scaffolding addresses this challenge by introducing a dedicated optimization methodology for 2D image vectorization, inspired by the artist's workflow and the structure of natural images. Our method consists of three synergistic components: *Interior Gradient Aggregation* to stabilize individual base curves, *Progressive Stratification* to construct a structured hierarchy, and *Rapid Inflation Scheduling* that rapidly establishes the hierarchical structure upon stabilized learning dynamics.

3.2 Interior Gradient Aggregation

In our scenario, the Gaussians serve as a surrogate representation that carries local interaction information for each curve. This differs from the 3D case [12], where the Gaussians are independent. The actual gradients driving our optimization problem are a sum of the gradients of the Gaussians sampled from a single curve, not the individual Gaussians. Therefore, the relative weights of gradient contributions within a curve significantly affect the optimization trajectory. There are two classes of Gaussians: those sampled along the boundary and those sampled along the interior of the curve. A key cause of *topology collapse* stems from the inherent scale disparity between the interior area and the boundary of a closed geometric primitive. In 2D space, the gradient magnitude derived from a curve's interior area scales quadratically $O(r^2)$, while the gradient from its boundary scales linearly $O(r)$, where r is the radius of the curve. Given a curve $\mathcal{B}_i$, let $\mathcal{A}_i$ denote the 2D interior region enclosed by it, and $\partial\mathcal{A}_i$ denote its boundary. During the backward pass, the gradient of the photometric reconstruction loss $\mathcal{L}_2 = \int_\Omega \|g(\mathcal{B})(x) - \mathcal{I}(x)\|_2^2 dx$ with respect to the control points P_j can be conceptually decomposed using the Reynolds transport theorem [13]:

$$\nabla_{P_j}\mathcal{L}_2 \approx \underbrace{\int_{\mathcal{A}_i} \nabla_{P_j} e(x) dx}_{\text{Area Gradient}\sim O(r^2)} + \underbrace{\int_{\partial\mathcal{A}_i} e(x) \left(\frac{\partial \mathbf{x}}{\partial P_j} \cdot \mathbf{n}\right) ds,}_{\text{Boundary Gradient}\sim O(r)} \tag{4}$$

where $e(x) = \|g(\mathcal{B})(x) - \mathcal{I}(x)\|_2^2$ is the per-pixel error, and $\mathbf{n}$ is the outward normal vector. In practice, we approximate the area term as a consistent discrete quadrature over interior Gaussian samples.

Consequently, for large base curves representing macroscopic structures, the area gradient induced by fine-grained internal textures can dominate the optimization trajectory. This makes the interior gradients crucial for stable optimization. Nevertheless, previous parallel optimization [14] ignored these interior gradients, leading to insufficient supervision where curves struggle to obtain information from their enclosed regions. We address this by aggregating the positional gradients from the interior of each curve with its boundary gradients. Importantly, *Interior Gradient Aggregation* does not by itself decide which frequency content a curve should explain; rather, it restores the missing positional support from the interior of a closed region. The separation of coarse structures from fine textures is imposed by *Progressive Stratification* in the next stage, which limits where and at what scale new curves should be introduced.

3.3 Progressive Stratification

Interior Gradient Aggregation effectively aggregates the gradients from the interior of the curve, allowing curves to focus on their own scope and scale. This enables us to employ the following *Progressive Stratification* to establish a structured hierarchy. Instead of unstructured random initialization and densification, we establish a structured hierarchy in an ordered manner, starting from the coarsest representation and gradually adding more details. This is analogous to how human artists build vector artwork from simple sketches and gradually add details.

To encourage separation between coarse and fine structures, new curves $\mathcal{B}^{(k)} \setminus \mathcal{B}^{(k-1)}$ are spawned exclusively in regions with high residual errors, computed as $E(x) = \|g(\mathcal{B}^{(k-1)})(x) - \mathcal{I}(x)\|_2^2$. More importantly, instead of balancing between pruning and densification operations, we monotonically increase the number of curves geometrically, starting from the coarsest representation, *e.g.*, $N_0 = 16$ and $N_k = r_{\text{num}}^k N_0$, where r_{num} is the ratio of the number of curves per generation. Let R_k denote the initialization radius of curves spawned at generation k. We impose a monotonic scale-decaying constraint on newly spawned curves: $R_k = r_{\text{rad}} R_{k-1}$. We reinterpret the depth parameter z_i used in ordering α-blending as the temporal ordering of curves, ensuring that newer curves are always on top of older ones. Newly spawned curves are strictly assigned a lower depth value $z_{new} < z_{base}$, placing them physically on top of older generations. This explicit z-ordering removes the chaotic depth-sorting instability present in previous works [14].

We find that setting $r_{\text{num}} = 2.0$ and $r_{\text{rad}} = 0.5$ yields the best results. This inherently aligns with the power law of natural images, ensuring that newer generations are less likely to overwrite the low-frequency areas already resolved by their predecessors. This spatial constraint prevents upper-layer high-frequency errors from distorting the established scaffold. Our Vector Scaffolding stabilizes the optimization landscape: the depth z_i is structurally fixed, the scale R_k is implicitly band-limited, and the interior gradients are included. Thanks to this structural stability, we can safely boost the learning rates by $\times 50$. That is, we increase the LR of color and opacity from 0.01 to 0.5 and the LR of control points from 2×10^{-4} to 1×10^{-2}. This extremely high LR enforces colors to match the image almost immediately and control points to quickly converge to their optimal positions.

3.4 Vector Scaffolding via Rapid Inflation

The optimization starts with an extremely sparse set of curves, *e.g.*, $N_0 = 16$, to exclusively capture the global illumination and macroscopic background colors. Once the lowest-frequency base layer is anchored, we aggressively double the number of curves every ΔT iterations until the target budget N_{max} is reached. We find that our *Progressive Stratification* strategy allows us to use a very high learning rate, which makes this *inflation phase* extremely fast. Throughout the paper, we fix $\Delta T = 100$. This means we double the number of curves every 100 iterations of training, reaching $N = 1024$ curves after only 600 iterations. For the rest of the training process, we do *not* reduce the learning rate but maintain the high learning rate of 0.5 (color & opacity) and 0.01 (control points) until the end of training, where we observe monotonic improvement in reconstruction fidelity. We achieve increased reconstruction fidelity with up to +1.4 dB PSNR improvement without optimization collapse with a small number of iterations (5k), ultimately accelerating the overall vectorization process by 2.5× compared to the best baseline [14] as shown in Figure 1b.

3.5 Regularization

To ensure the final output $\mathcal{B}^*$ is an editable vector graphic rather than fuzzy Gaussians, we apply topology regularization with opacity-based pruning. Since practical vector-art workflows favor solid, opaque elements, we introduce an opacity regularization loss pushing α_i towards 1.0:

$$\mathcal{L}_\alpha \;=\; \frac{1}{N}\sum_{i=1}^{N} |1 - \sigma(\alpha_i)|, \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid activation. We also adopt the Xing loss [17] $\mathcal{L}_{\text{Xing}}$ and Bézier curve regularization [14] $\mathcal{L}_{\text{BS}}$ to avoid self-intersection and maintain structural integrity:

$$\mathcal{L}_{\text{reg}} \;=\; 0.01\mathcal{L}_\alpha + 0.02\mathcal{L}_{\text{Xing}}(\mathcal{B}) + \mathcal{L}_{\text{BS}}(\mathcal{B}). \tag{6}$$

The total objective is given in Equation (1).

Coupled with this regularization, we periodically prune any curve $\mathcal{B}_i$ where $\sigma(\alpha_i) < 0.01$. Curves that become occluded or drift from target structures naturally lose gradient contribution and opacity, enabling automated culling without complex heuristics.

4 Experiments

4.1 Experimental Setup

Datasets and Metrics. We evaluate our method on two distinct datasets. First, we use the Kodak [5] dataset to compare general performance on natural images. Second, to test reconstruction quality on high-frequency textures, we utilize the high-resolution DIV2K [1] training dataset. We use standard image quality metrics, PSNR, SSIM, and LPIPS [30], to evaluate reconstruction and perceptual fidelity, and use qualitative visualizations to assess structural coherence. We also track the wall clock time for the entire optimization process to evaluate the computational efficiency of our framework. To demonstrate structural efficiency, we strictly enforce identical curve budgets ($N \in \{256, 512, 1024\}$) across all comparisons.

Table 1: Quantitative Evaluation on the Kodak dataset [5]. We outperform the evaluated closed-curve baselines with available Kodak results across all curve budgets.

	256 Curves			512 Curves			1024 Curves		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
DiffVG [13]	24.11	0.622	0.513	25.34	0.666	0.475	26.66	0.719	0.420
Bézier Splatting [14]	24.19	0.621	0.519	25.61	0.664	0.485	26.91	0.708	0.448
Ours	**25.18**	**0.631**	**0.427**	**26.68**	**0.687**	**0.353**	**28.30**	**0.749**	**0.275**

Baselines. We benchmark our method against existing differentiable vectorization methods: DiffVG [13], LIVE [17], LIVSS [23], and Bézier Splatting [14]. Since we target practically useful vector graphics, we only consider *closed curves* for evaluation, where areas are filled by the interior of the curves rather than strokes with arbitrary thickness.

Implementation Details. We utilize the Bézier Splatting [14] rasterizer as the differentiable rasterization backend. For all experiments, we also aggregate the interior gradient from the interior of the curve. Leveraging the structurally stabilized loss landscape provided by our exponential scale decaying ($r_{\text{num}} = 2.0, r_{\text{rad}} = 0.5$), we employ an aggressive spatial learning rate of 0.01 for the geometric parameters, 50× higher than the standard configuration of Bézier Splatting [14]. The curve population inflates monotonically, doubling every $\Delta T = 100$ iterations until reaching the target budget N. That is, starting from $N_0 = 16$ curves, we double the number of curves every 100 iterations of training immediately after initialization. This means we densify 4 times for $N = 256$ curves, 5 times for $N = 512$ curves, and 6 times for $N = 1024$ curves, reaching $N = 1024$ curves at the 600-th iteration. For the optimizer, we use the Adan optimizer [25] with $\beta_1 = 0.98$, $\beta_2 = 0.92$, and $\beta_3 = 0.99$ by default. Learning rates for the scaling and rotation (Cholesky) parameters are set to $\lambda = 0.5$, which is 50× higher than Bézier Splatting's learning rate of 0.01. Learning rates for the position (control points), color, and opacity parameters are set to $\lambda_{\text{position}} = 0.01$, $\lambda_{\text{color}} = 0.5$, and $\lambda_{\text{opacity}} = 0.5$, respectively. For our Vector Scaffolding framework, we train each model for 5k iterations. All reported timings are measured on a single A100 GPU.

4.2 Main Results

Kodak Experiment. Table 1 summarizes the performance of our method on the Kodak dataset [5]. On Kodak, our hierarchical optimization method outperforms all reproduced baselines under matched closed-curve budgets. We improve PSNR by 1.0–1.4 dB over Bézier Splatting across curve budgets, with substantial gains in perceptual quality (LPIPS). Notably, the performance gap between our method and the state-of-the-art Bézier Splatting [14] exceeds that between DiffVG [13] and Bézier Splatting [14]. Since our method and Bézier Splatting

Table 2: Quantitative Evaluation on the DIV2K Dataset. We compare our Vector Scaffolding with state-of-the-art differentiable vectorization methods (closed curves only). Our method consistently achieves the best PSNR across all sparsity levels, with a margin of roughly 0.5–0.6 dB over Bézier Splatting, while achieving competitive or better perceptual quality depending on the curve budget.

	256 Curves			512 Curves			1024 Curves		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
DiffVG [13]	20.69	0.578	0.548	21.82	0.601	0.531	22.95	0.631	0.509
LIVE [17]	20.09	0.576	0.543	21.70	0.611	0.521	23.11	0.648	0.495
LIVSS [23]	17.71	**0.586**	**0.542**	18.71	**0.630**	0.530	19.83	**0.678**	0.517
Bézier Splatting [14]	20.74	0.580	0.546	22.11	0.607	0.528	23.45	0.639	0.507
Ours	**21.37**	0.564	0.548	**22.64**	0.598	**0.503**	**23.99**	0.640	**0.450**

share the same differentiable rasterization backend, the observed gains isolate the effect of optimization restructuring rather than renderer replacement. This strongly suggests that our hierarchical optimization framework is as significant as the rasterization backend, which has been the sole focus of recent literature.

DIV2K Experiment. To evaluate the robustness and generalizability of our framework, we compare our method with previous vectorization methods on the high-resolution DIV2K training dataset containing 800 images [1]. This dataset is known for its extreme high-frequency textures and complex structures, originally designed for image super-resolution benchmarks. Table 2 summarizes the results. Even under this challenging evaluation, our *Vector Scaffolding* consistently outperforms the state-of-the-art by a significant margin of roughly 0.5–0.6 dB in PSNR. Regarding perceptual quality measured by LPIPS, our method achieves the best scores at 512 and 1024 curves, while remaining comparable at 256 curves. It is noteworthy that the baseline methods tend to show higher SSIM scores at some budgets. This trade-off in SSIM can be plausibly explained as a consequence of our *geometric regularization*, which prioritizes the professional usability of the resulting vector artwork. SSIM evaluates on a local-window basis, which can favor locally detailed reconstructions even when they introduce less coherent vector structures. The baseline methods allocate more capacity to local texture and edge fluctuations, whereas our hierarchical schedule tends to preserve larger coherent fills, which is more consistent with LPIPS at medium and high budgets. This SSIM trade-off is consistent with our design goal of favoring well-stratified vector layers over aggressive local texture matching.

Accelerated Optimization. As shown in the convergence plot in Figure 1b, our method improves the speed-fidelity trade-off by enhancing *both* reconstruction quality and optimization speed. Our *Vector Scaffolding* framework ensures granularity-wise scale separation and temporal Z-ordering, which stabilizes the optimization process and reduces the required gradient updates by allowing extremely high learning rates. Consequently, we consistently accelerate the total

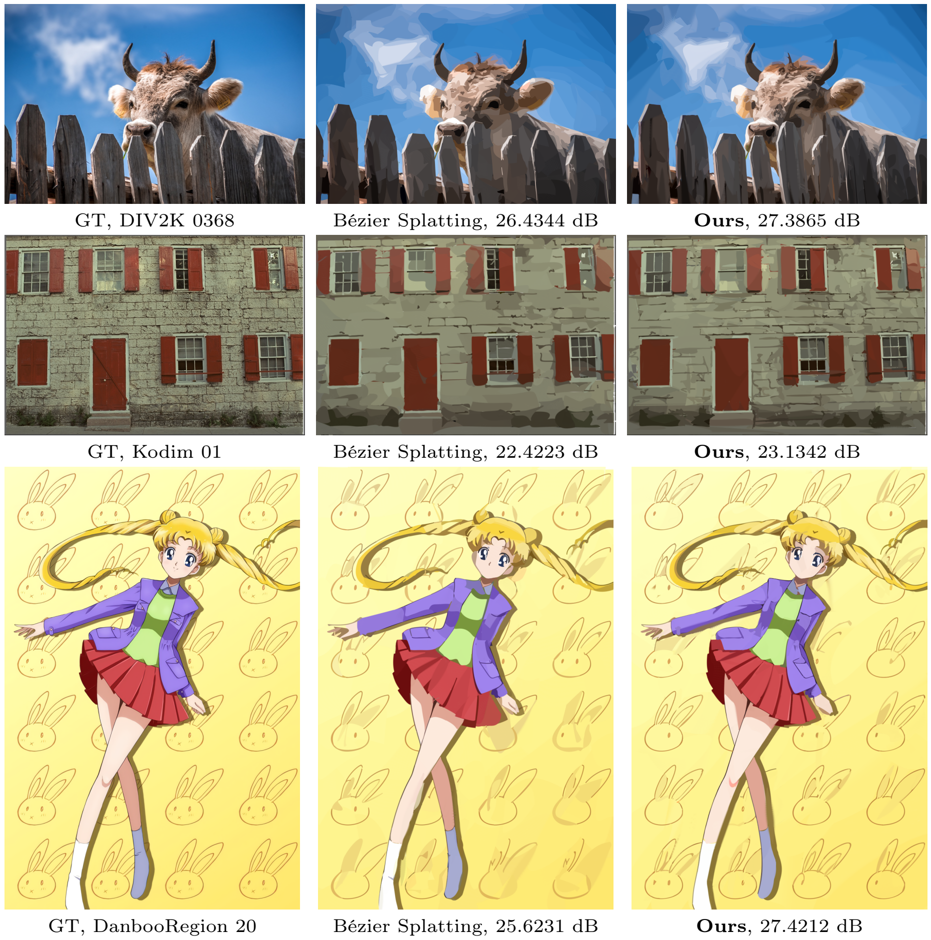


Fig. 3: Qualitative Comparison. Compared with the state-of-the-art differentiable vectorization method [14], our method preserves fine structural details and coherent object boundaries under the same curve budget ($N = 512$).

optimization time by 2.5× compared to the fastest baseline, Bézier Splatting [14], while achieving the best PSNR scores. We emphasize that this 2.5× figure is measured in *wall-clock time*; the per-step computational overhead of our method is in fact 1.25× smaller than Bézier Splatting, since Interior Gradient Aggregation does not introduce a separate calculation stage and the rapid-inflation schedule further reduces the number of active primitives during the early phase.

4.3 Structural Hierarchy and Arbitrary Level-of-Detail

Since our structured densification scheduling via *Progressive Stratification* and temporal Z-ordering enables monotonically increasing the number of curves, the

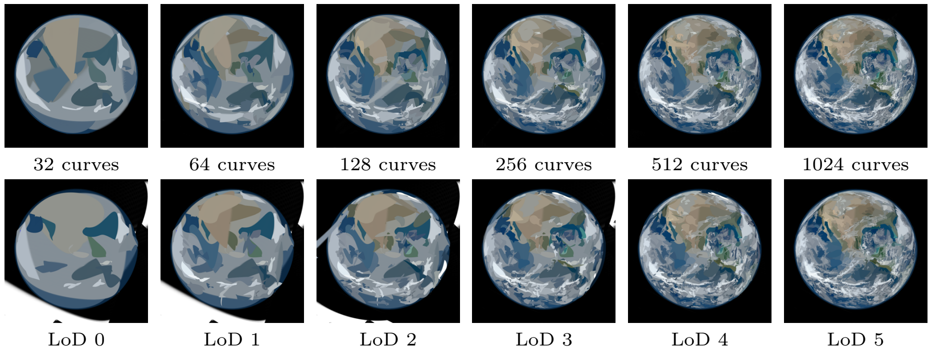


Fig. 4: LoD Control Demonstration. We fit our Vector Scaffolding to a super high-resolution image of the Earth (8000×8000) [19]. The first row shows the training dynamics at different curve counts, while the second row shows the level-of-detail (LoD) separation after fitting 1024 curves.

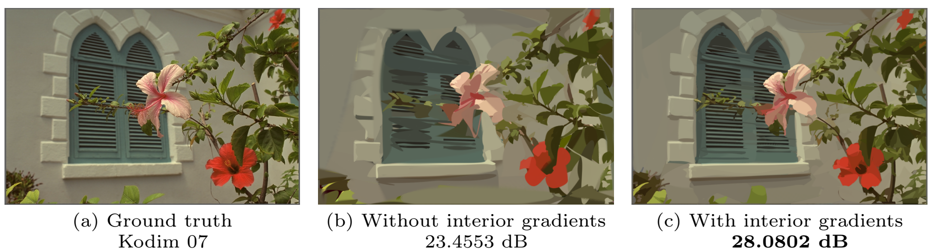


Fig. 5: Effect of Interior Gradients. (a) Ground truth. (b) Without interior gradients, the base curves lose their internal anchors, causing optimization drift and poor convergence. (c) With interior gradients, our method maintains structural integrity while capturing photometric information.

vector representation can be densified sequentially from base structures to finest details. Therefore, our Vector Scaffolding naturally supports flexible level-of-detail (LoD) rendering. This is partially demonstrated in Tables 1 and 2. We further apply our method to a 64-megapixel high-resolution image to demonstrate flexible LoD control. The results are shown in Figure 4. By fixing all optimization hyperparameters such as the learning rate, the schedule, and the densification logic while varying only the number of densification steps, we can continuously increase the number of curves to achieve better reconstruction quality. This mimics the typical artistic workflow, from simple base sketches to procedural polishing, providing designers with visually meaningful and easily editable layers, as shown in Figure 1 (left).

4.4 Ablation Studies

To validate our key ideas, we perform ablation studies isolating each component. More ablation studies are provided in Supplementary Material.

(a) Ground truth DIV2K 0383 (b) Baseline, lr × 1 24.4731 dB (c) Baseline, lr × 50 19.1911 dB (d) **Ours**, lr × 50 **24.6808 dB**

Fig. 6: Hierarchical Scaffolding vs. Flat Optimization. We ablate the *Progressive Stratification* and temporal Z-ordering by reverting to a flat optimization strategy where all N curves are initialized simultaneously and updated without spatial constraints. Flat optimization fails under our extreme learning rate. This demonstrates the efficacy of our frequency-separation for fast and stable convergence.

Efficacy of Interior Gradient. We ablate the interior gradient in Figure 5. Previous methods [14] neglect the positional gradient from the interior of the curve, which leads to topology collapse. Without interior gradients, the base curves lose their internal anchors, causing optimization drift and significantly degrading the PSNR. We show that interior gradients and boundary gradients jointly contribute to the PSNR and preserve structural integrity.

Hierarchical Scaffolding vs. Flat Optimization. We ablate *Progressive Stratification* and temporal Z-ordering by reverting to a flat optimization strategy where all N curves are initialized simultaneously and updated without spatial constraints. The results are shown in Figure 6. Under our aggressive learning rate, this flat optimization completely fails, resulting in extreme gradient oscillations and diverging losses. This suggests that our frequency-separation is an effective technique for fast and stable convergence.

5 Conclusion

In this work, we presented **Vector Scaffolding**, a hierarchical optimization framework for differentiable image vectorization. Previous methods focused on improving the rendering backend; however, we found that overall performance including reconstruction quality (PSNR), perceptual quality (LPIPS), optimization speed, and editability is limited by the topological structure of the vector representation. Our main strategy is to stabilize the optimization dynamics by matching the hierarchical structure of the optimization process to that of natural and articulated images. This structural resonance enables a high learning rate regime without causing instability, allowing us to focus directly on topology construction. Without altering the underlying renderer, our framework achieves a 2.5× speedup in wall clock time and substantial PSNR gains under matched curve budgets, demonstrating that optimization structure is a major determinant of vectorization quality.

Acknowledgements

This work was supported in part by the IITPgrants [No. RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University), No. RS-2025-02303870, No.2022-0-00156] funded by the Korea government (MSIT).

References

1. Agustsson, E., Timofte, R.: NTIRE 2017 challenge on single image super-resolution: Dataset and study. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1122–1131 (2017). https://doi.org/10.1109/CVPRW.2017.150
2. Cao, D., Wang, Z., Echevarria, J., Liu, Y.: SVGformer: Representation learning for continuous vector graphics using transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10093–10102 (2023)
3. Chen, Y., Ni, B., Liu, J., Huang, X., Chen, X.: Towards high-fidelity artistic image vectorization via texture-encapsulated shape parameterization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15877–15886 (2024). https://doi.org/10.1109/CVPR52733.2024.01503
4. Du, Z.J., Kang, L.F., Tan, J., Gingold, Y., Xu, K.: Image vectorization and editing via linear gradient layer decomposition. ACM Transactions on Graphics **42**(4), 1–13 (2023). https://doi.org/10.1145/3592128
5. Eastman Kodak Company: Kodak lossless true color image suite. Dataset (1999), https://r0k.us/graphics/kodak/, accessed: 2026-05-12
6. Guédon, A., Lepetit, V.: SuGaR: Surface-aligned gaussian splatting for efficient 3D mesh reconstruction and high-quality mesh rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5354–5363 (2024). https://doi.org/10.1109/CVPR52733.2024.00512
7. Guo, M., Wang, B., He, K., Matusik, W.: TetSphere splatting: Representing high-quality geometry with lagrangian volumetric meshes. In: International Conference on Learning Representations (ICLR) (2025)
8. Hirschorn, O., Jevnisek, A., Avidan, S.: Optimize & reduce: A top-down approach for image vectorization. Proceedings of the AAAI Conference on Artificial Intelligence **38**(3), 2148–2156 (2024). https://doi.org/10.1609/aaai.v38i3.27987
9. Ho, J., Jain, A.N., Abbeel, P.: Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems. vol. 33, pp. 6840–6851 (2020)
10. Huang, B., Yu, Z., Chen, A., Geiger, A., Gao, S.: 2D gaussian splatting for geometrically accurate radiance fields. In: ACM SIGGRAPH 2024 Conference Papers. pp. 1–11. Association for Computing Machinery (2024). https://doi.org/10.1145/3641519.3657428
11. Jain, A., Xie, A., Abbeel, P.: VectorFusion: Text-to-SVG by abstracting pixel-based diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1911–1920 (2023). https://doi.org/10.1109/CVPR52729.2023.00190
12. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4), 1–14 (2023). https://doi.org/10.1145/3592433

13. Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Transactions on Graphics **39**(6), 1–15 (2020). https://doi.org/10.1145/3414685.3417871
14. Liu, X., Zhou, C., Zhao, N., Huang, S.: Bézier splatting for fast and differentiable vector graphics rendering. In: Advances in Neural Information Processing Systems (2025)
15. Lopes, R.G., Ha, D., Eck, D., Shlens, J.: A learned representation for scalable vector graphics. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 7930–7939 (2019). https://doi.org/10.1109/ICCV.2019.00802
16. Luiten, J.T., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3D gaussians: Tracking by persistent dynamic view synthesis. In: 2024 International Conference on 3D Vision (3DV). pp. 800–809 (2024). https://doi.org/10.1109/3DV62453.2024.00044
17. Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y., Shi, H.: Towards layer-wise image vectorization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 16314–16323 (2022). https://doi.org/10.1109/CVPR52688.2022.01583
18. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics **41**(4), 1–15 (2022). https://doi.org/10.1145/3528223.3530127
19. NASA Goddard Photo and Video: Most amazing high definition image of earth – blue marble 2012. Flickr image, NASA Goddard Space Flight Center (2012), https://www.flickr.com/photos/gsfc/6760135001, public domain (NASA media usage guidelines). Accessed: 2026-05-12.
20. Reddy, P., Gharbi, M., Lukáč, M., Mitra, N.J.: Im2Vec: Synthesizing vector graphics without vector supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7342–7351 (2021). https://doi.org/10.1109/CVPR46437.2021.00726
21. Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: Advances in Neural Information Processing Systems. vol. 33, pp. 7462–7473 (2020)
22. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: DreamGaussian: Generative gaussian splatting for efficient 3D content creation. In: International Conference on Learning Representations (ICLR) (2024)
23. Wang, Z., Huang, J., Sun, Z., Gong, Y., Cohen-Or, D., Lu, M.: Layered image vectorization via semantic simplification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7728–7738 (2025)
24. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4D gaussian splatting for real-time dynamic scene rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20310–20320 (2024). https://doi.org/10.1109/CVPR52733.2024.01920
25. Xie, X., Zhou, P., Li, H., Lin, Z., Yan, S.: Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(12), 9508–9520 (2024). https://doi.org/10.1109/TPAMI.2024.3423382
26. Xing, X., Wang, C., Zhou, H., Zhang, J., Yu, Q., Xu, D.: DiffSketcher: Text guided vector sketch synthesis through latent diffusion models. In: Advances in Neural Information Processing Systems. vol. 36, pp. 15869–15889 (2023)

27. Xing, X., Zhou, H., Wang, C., Zhang, J., Xu, D., Yu, Q.: SVGDreamer: Text guided SVG generation with diffusion model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4546–4555 (2024). https://doi.org/10.1109/CVPR52733.2024.00435
28. Yi, T., Fang, J., Wang, J., Wu, G., Xie, L., Zhang, X., Liu, W., Tian, Q., Wang, X.: GaussianDreamer: Fast generation from text to 3D gaussians by bridging 2D and 3D diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6796–6807 (2024). https://doi.org/10.1109/CVPR52733.2024.00649
29. Zhang, P., Zhao, N., Liao, J.: Text-to-vector generation with neural path representation. ACM Transactions on Graphics **43**(4), 1–13 (2024). https://doi.org/10.1145/3658204
30. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 586–595 (2018). https://doi.org/10.1109/CVPR.2018.00068
31. Zhang, X., Ge, X., Xu, T., He, D., Wang, Y., Qin, H., Lu, G., Geng, J., Zhang, J.: GaussianImage: 1000 FPS image representation and compression by 2D gaussian splatting. In: Computer Vision – ECCV 2024. Lecture Notes in Computer Science, vol. 15067, pp. 327–345. Springer (2024). https://doi.org/10.1007/978-3-031-72673-6_18
32. Zhang, Y., Li, B., Kuznetsov, A., Jindal, A., Diolatzis, S., Chen, K., Sochenov, A., Kaplanyan, A., Sun, Q.: Image-GS: Content-adaptive image representation via 2D gaussians. In: ACM SIGGRAPH 2025 Conference Papers. pp. 1–11. Association for Computing Machinery (2025). https://doi.org/10.1145/3721238.3730596
33. Zwicker, M., Pfister, H., van Baar, J., Gross, M.: EWA volume splatting. In: Proceedings Visualization, 2001. VIS '01. pp. 29–36. IEEE Computer Society (2001). https://doi.org/10.5555/601671.601674