

CubicSplat: Differentiable Vector Graphics via Error-Bounded Forward Relaxation

Chenglong Liu¹, Xin Zhang¹, Yimeng Zhu¹, Liyang He¹, Yixiao Ma¹,
Yu Su^{2,3}, Zhenya Huang¹, and Qi Liu¹

¹ State Key Laboratory of Cognitive Intelligence, University of Science and Technology of China, Hefei, 230088, China

{sepcnt,zx2020,zzyymm,heliyang,mayx}@mail.ustc.edu.cn
{huangzhy,qiliuql}@ustc.edu.cn

² Hefei Normal University, Hefei, 230061, China
yusu@hfnu.edu.cn

³ Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua, 321004, Zhejiang, China

Abstract. Vector graphics are prized for their resolution independence, compact storage, and direct editability, making differentiable optimization of their parametric primitives an attractive goal. Yet classical rasterization is discontinuous with respect to geometry, and existing remedies that smooth the forward pass demand increasingly elaborate heuristics as scene complexity grows. We trace this fragility to a **gradient seesaw**: design choices that improve forward geometric exactness can systematically degrade the induced gradient signal, and vice versa. To navigate this tension we introduce **CubicSplat**, a differentiable vector rasterizer that replaces Bézier closest-point solvers with uniform polyline surrogates whose geometric error is bounded at $\mathcal{O}(S^{-2})$. The resulting static computation graph yields well-conditioned gradients by construction, while a compositing-derived visibility mechanism prunes degenerate primitives without auxiliary regularization. On DIV2K and Kodak benchmarks CubicSplat achieves state-of-the-art reconstruction quality with over **2 dB** PSNR gain in the closed-fill setting, while training up to **4×** faster than prior methods. The code is available at <https://github.com/CubicSplat/repo>

Keywords: Differentiable rendering · Vector graphics · Scene representation · Gradient conditioning

1 Introduction

Vector graphics (VGs) encode images as resolution-independent, compact, and directly editable parametric primitives. Fitting these parameters by gradient descent is conceptually elegant but hindered by discontinuous rasterization, which

 Corresponding author. Contact: qiliuql@ustc.edu.cn

yields gradients that are either zero almost everywhere or explosive near boundaries. A natural remedy is to redesign the forward pass so that pixels respond smoothly to geometric change.

In 3D mesh rendering, SoftRas [22] trades a softened forward image for smooth gradients, while nvdiffrast [18] preserves a crisp forward pass and recovers boundary gradients via deferred antialiasing. DiffVG [20] brings the same idea to 2D vector graphics, replacing hard inside/outside decisions with area-averaged coverage. Follow-up methods such as LIVE [26], LIVSS [34], SGLIVE [45], and Bézier Splatting [23] have achieved impressive results, yet a persistent frustration remains: as scenes grow in complexity, these methods require opacity heuristics, auxiliary regularization losses, and ad-hoc annealing schedules to keep optimization from collapsing. The sheer volume of such scaffolding suggests that something more fundamental is at play.

A simple experiment exposes the missing piece (Sec. 4.4). Increasing forward fidelity (finer sampling, adaptive subdivision, tighter Bézier approximation) monotonically reduces geometric error, yet the quality of the learned representation plateaus almost immediately while runtime grows and optimization can become less stable. This plateau is hard to reconcile with approximation error as the bottleneck, but follows naturally from the dual role of the forward pass: it must be accurate enough to evaluate the objective, yet more critically, it must induce gradients well-conditioned enough to navigate the objective landscape.

We identify this tension as the **gradient seesaw**: design choices that improve forward exactness can systematically degrade the quality of the induced gradient signal, while an overly coarse forward approximation can itself become the binding source of error. Three concrete mechanisms illustrate the effect: **(i) Exact forward, ill-conditioned backward.** Differentiating through iterative closest-point or root-finding [20] procedures for exact Bézier distances introduces ill-conditioning at degenerate configurations (Appendix A.2). The forward model is geometrically faithful, yet the gradients it induces become unreliable precisely where optimization needs them most. **(ii) Smooth gradients, bloated forward.** Point-sample based methods [23] that place Gaussian splats along curves yield well-behaved gradients, yet the discrete summation can deviate structurally from the true geometry around high-curvature regions and does not commute with spatial rescaling (Appendix A.3). When the learned vector representation is rendered at resolutions higher than those used during training, the fixed sampling budget cannot keep pace, and the reconstructed image progressively diverges from the intended geometry. **(iii) Adaptive refinement, discontinuous graph.** Adaptive subdivision reduces forward geometric error on demand, yet each discrete split event changes the computation graph, producing non-smooth, heteroscedastic gradient variance across primitives that undermines consistent optimization.

The upshot is that fidelity operates in a regime of diminishing, and eventually negative, returns: a more exact forward model can be a worse gradient oracle, yet a forward model that is too inexact optimizes the wrong objective.

Motivated by this analysis, we propose **Error-bounded Forward Relaxation**: we intentionally relax the forward rasterization to obtain a stable, well-conditioned gradient oracle while bounding the induced forward bias. We instantiate this principle in **CubicSplat**, a tile-parallel differentiable vector graphics renderer that achieves throughput comparable to Bézier Splatting without coupling geometric fidelity or multi-scale reconstruction quality to a per-primitive sampling budget. The core idea is to approximate each Bézier curve with a uniform polyline surrogate so that all distance queries reduce to closed-form point-to-segment computations and no iterative solver enters the backward pass. This eliminates the dominant source of gradient ill-conditioning while introducing only a bounded, controllable geometric bias. CubicSplat further supports closed shapes through winding-number fill and handles occlusion via standard Porter–Duff compositing, providing a unified treatment of open strokes and filled regions without auxiliary sampling or ad-hoc regularization. As a practical byproduct, the compositing pass naturally yields per-primitive visibility statistics that we exploit to maintain efficiency at large primitive counts.

Our experiments validate these claims. On a 200-image DIV2K subset with $N=1024$ primitives, CubicSplat achieves 25.78 dB (closed) / 25.93 dB (open) versus Bézier Splatting’s 23.45 / 25.45 dB, while reducing wall-clock training time by $4\times$ (closed) and $1.8\times$ (open). The method is also more parameter-efficient: with only 256 curves CubicSplat already matches the PSNR that DiffVG requires 1024 curves to reach. A single forward–backward step on a 2040×1344 image with 2048 curves is $1.4\times$ faster than Bézier Splatting, $120\times$ faster than DiffVG. Beyond these core results, rendering remains resolution-faithful up to 24K and primitive counts scale to 32K on a single consumer GPU. Ablations further confirm sampling-density invariance, reinforcing that the polyline surrogate acts as a better-conditioned optimization oracle rather than a crude approximation.

Our contributions are as follows:

- We identify the **forward-fidelity vs. gradient-quality tension** intrinsic to differentiable vector rendering and provide empirical evidence through a broad sampling-density invariance regime.
- We propose **Error-bounded Forward Relaxation**, bringing the classical polyline approximation into a principled oracle-design framework: a family $R_S = R \circ \Pi_S$ of surrogate renderers with $\mathcal{O}(S^{-2})$ geometric bias, paired with compositing-derived visibility pruning that replaces explicit geometric regularization with renderer-intrinsic capacity allocation.
- We demonstrate that **CubicSplat** achieves state-of-the-art quality–efficiency tradeoffs, substantially outperforming strong baselines on both open and closed primitives while requiring a fraction of the training time.

2 Related Work

2.1 Differentiable Rasterization

Early differentiable renderers recover gradients through boundary integrals or Monte Carlo edge sampling [14, 24]. SoftRas [22] introduces probabilistic raster-

ization that yields smooth gradients at the cost of a softened forward image, whereas `nvdiffrast` [18] preserves a crisp forward pass and recovers boundary gradients via deferred antialiasing. Approximate differentiable renderers [17] deliberately trade forward-image fidelity for smooth gradients, and Wang et al. [35] explicitly analyze the bias-variance tradeoff governed by a smoothing parameter in SDF rendering. For splatting-style rasterizers, unbiased gradient estimation under stochastic sampling has also been studied [28].

2.2 Differentiable Vector Graphics

DiffVG [20] introduced a fully differentiable rasterizer for vector graphics with both an analytical prefiltering mode and a Monte Carlo antialiasing mode. The analytical path differentiates through iterative cubic Bézier closest-point solvers, whose ill-conditioning we analyze in Appendix A.2; the Monte Carlo mode avoids this issue but incurs substantial per-pixel sampling cost, making it prohibitive for high-resolution scenes. LIVE [26], LIVSS [34], and SGLIVE [45] reduce inter-curve gradient interference by incrementally adding paths. On the representation side, Im2Vec [30] learns differentiable path representations via closed-form Bézier integrals, and CLIPasso [33] optimizes stroke sequences under semantic losses.

Bézier Splatting [23] accelerates differentiable vector rendering by distributing 2D Gaussian primitives along Bézier curves. Because image formation relies on discrete point samples [43, 44], the method inherits phase- and density-dependent aliasing that requires denser sampling to suppress, and its discretized summation does not commute with spatial rescaling, coupling optimization outcomes to sampling hyperparameters (Appendix A.3). Our method sidesteps these issues with an explicit, error-bounded polyline surrogate that delivers stable gradients without tying approximation quality to a per-primitive sampling budget.

2.3 Gaussian Splatting and Gradient Quality

3D Gaussian Splatting [16] and its variants [8, 9, 12, 15, 25] achieve real-time novel-view synthesis through differentiable Gaussian rasterization. Recent studies expose several gradient pathologies in this family of methods: approximate opacity compositing versus volumetric rendering [3], intra-primitive gradient collision [39], opacity bias from densification rules [32], and forward/backward depth-order mismatch [29]. These findings reinforce the broader theme that forward approximations can have outsized effects on gradient quality.

3 Method

We view differentiable vector graphics learning as gradient-oracle design. Rather than pushing the forward renderer toward maximal geometric exactness, we construct a family of relaxed forward passes that yield stable, well-conditioned gradients while keeping the forward bias bounded and interpretable.

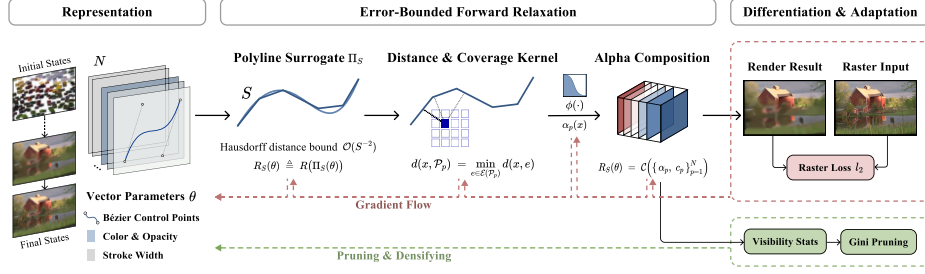


Fig. 1: Overview of the **CubicSplat** pipeline. 1) Each primitive is encoded as a set of learnable Bézier parameters. 2) The forward pass applies the error-bounded relaxation: a polyline surrogate Π_S converts Bézier geometry into closed-form distance queries, and a smooth coverage kernel ϕ maps the signed residual to per-pixel opacity, which is assembled into the final image via alpha compositing. 3) The raster loss drives gradient updates back to the vector parameters, while compositing-derived visibility statistics feed into Gini pruning and densification.

3.1 The Framework of CubicSplat

Let θ collect the continuous parameters of N layered vector primitives: control points, thickness r_p , color c_p , and opacity. The front-to-back layer order is fixed throughout optimization. Given a target image I^* , we solve

$$\min_{\theta} \mathcal{L}(R_S(\theta), I^*), \quad (1)$$

where R_S is a relaxed differentiable rasterizer parameterized by a sampling density S , and $\mathcal{L}$ is a pixel-space loss.

The renderer R_S determines the gradient oracle $\nabla_{\theta} \mathcal{L}$; in vector graphics, many common failures (unstable convergence, phantom gradients under occlusion, sensitivity to sampling rules) are better explained by oracle conditioning than by forward image fidelity alone. We therefore design R_S to optimize oracle quality subject to a bounded forward bias, rather than to maximize geometric exactness.

Concretely, R_S decomposes into three stages that map parameters to pixels:

$$\alpha_p(x) = \underbrace{\phi\left(\frac{\delta_p(x, \Pi_S(\mathcal{B}_p))}{\tau_p}\right)}_{\text{coverage kernel (Sec. 3.3)}}, \quad R_S(\theta) = \underbrace{\mathcal{C}\left(\{\alpha_p, c_p\}_{p=1}^N\right)}_{\text{compositing (Sec. 3.4)}}, \quad (2)$$

where $\delta_p(x)$ is a **topology-dependent signed residual** (Sec. 3.3) that encodes how far pixel x lies inside or outside primitive p , and:

1. $\Pi_S(\mathcal{B}_p)$ is a **polyline surrogate** (Sec. 3.2) that approximates each Bézier primitive with a uniform S -segment polyline, reducing all distance queries to closed-form point-to-segment computations and eliminating iterative solvers from the backward pass;

2. $\phi(\cdot)$ is a smooth, monotone **coverage kernel** (Sec. 3.3) that maps the signed residual to soft per-pixel opacity via an anti-aliasing band of width τ_p ;
3. $\mathcal{C}$ is a front-to-back **alpha-compositing** pass (Sec. 3.4) whose intermediate transmittance values yield per-primitive visibility statistics, enabling inequality-triggered pruning via renderer-derived signals rather than explicit geometric regularization.

The sampling density S enters only through the surrogate Π_S : increasing S monotonically reduces geometric approximation error at $\mathcal{O}(S^{-2})$, but our ablations reveal a broad solution-equivariance regime beyond a modest threshold S_0 , confirming that S functions as a relaxation knob for oracle conditioning rather than an approximation budget (Sec. 4.4).

The remainder of this section details each stage. Only the training oracle is relaxed; at evaluation time the learned parameters can be rendered with an arbitrarily high S or an exact Bézier renderer.

3.2 Error-Bounded Forward Relaxation

Consider a Bézier segment $\mathcal{B}(t)$, $t \in [0, 1]$. We define a surrogate operator Π_S that maps each Bézier segment to a uniform S -segment polyline:

$$\mathcal{P}_S = \Pi_S(\mathcal{B}) = \{v_0, \dots, v_S\}, \quad v_i = \mathcal{B}\left(\frac{i}{S}\right), \quad (3)$$

and similarly for an entire primitive consisting of one or more Bézier segments. Any fixed uniform rule fits the same framework; we adopt a fixed rule to preserve a static computation graph.

This induces a family of relaxed renderers:

$$R_S(\theta) \triangleq R(\Pi_S(\theta)), \quad (4)$$

where $\Pi_S(\theta)$ applies the polyline surrogate to all Bézier geometry encoded in θ .

Increasing S monotonically improves geometric fidelity: Π_S converges to the original Bézier geometry as $S \rightarrow \infty$. We bound the geometric discrepancy between $\mathcal{B}$ and $\Pi_S(\mathcal{B})$ [4, 7, 31] (e.g., a Hausdorff-distance bound decaying as $\mathcal{O}(S^{-2})$ under mild smoothness conditions), which implies that the induced coverage bias is also controlled. This establishes R_S as an error-controlled relaxation of R ; a formal treatment is given in Appendix A.4.

While Π_S becomes geometrically exact as S increases, our ablations reveal a broad solution-equivariance regime: there exists a threshold S_0 such that for all $S \geq S_0$, optimizing $\min_{\theta} \mathcal{L}(R_S(\theta), I^*)$ yields learned representations θ_S that are geometrically equivalent and nearly identical in reconstruction quality, even though the forward surrogate continues to grow more accurate.

This behavior is inconsistent with an approximation-error-dominated narrative. If optimization were primarily limited by forward approximation error, increasing S would systematically alter the learned solution. Instead, beyond S_0 , S chiefly governs oracle conditioning and runtime. We therefore interpret S as

a forward-relaxation knob that trades forward fidelity for a better-conditioned gradient oracle. Only the training oracle is relaxed: the learned vector parameters can always be rendered with a higher S or an exact Bézier renderer at evaluation time (Sec. 4.5).

3.3 Differentiable Coverage from Signed Residual

We define coverage through a smooth kernel applied to a signed residual that measures how far a pixel lies inside or outside a primitive.

Let $\mathcal{P}_p$ denote the polyline surrogate for primitive p . The Euclidean distance from a pixel center x to the polyline is

$$d(x, \mathcal{P}_p) = \min_{e \in \mathcal{E}(\mathcal{P}_p)} d(x, e), \quad (5)$$

where $\mathcal{E}(\mathcal{P}_p)$ is the set of line segments and $d(x, e)$ is the closed-form point-to-segment distance. This avoids iterative closest-point solvers and yields a stable computation graph.

We construct a signed residual $\delta_p(x)$ that is positive inside the primitive and negative outside, encoding both geometry and topology in a single continuous quantity:

$$\delta_p(x, \Pi_S(\mathcal{B}_p)) = \begin{cases} r_p - d(x, \mathcal{P}_p), & \text{open stroke,} \\ s_p(x) \cdot d(x, \mathcal{P}_p), & \text{closed fill,} \end{cases} \quad (6)$$

where $r_p \geq 0$ is the learnable stroke half-width, and $s_p(x) \in \{-1, +1\}$ is the sign derived from a nonzero winding-number test on the polyline boundary ($s_p = +1$ if x is inside, $s_p = -1$ otherwise). For open strokes, a pixel is inside when it falls within distance r_p of the curve; for closed fills, inside/outside is determined entirely by the winding rule consistent with the nonzero fill convention.

The coverage kernel ϕ in Eq. (2) maps the normalized signed residual δ_p/τ_p to soft opacity through a smooth, monotone function $\phi : \mathbb{R} \rightarrow [0, 1]$. The anti-aliasing band $\tau_p > 0$ controls the transition width: ϕ saturates to 1 well inside the primitive and decays to 0 well outside, with the steepest response at the boundary ($\delta_p = 0$) where optimization most needs informative gradients. The specific kernel form (half-sided Gaussian for strokes, logistic sigmoid for fills) and its motivation are detailed in Appendix A.5.

The winding-number sign $s_p(x)$ is a discrete quantity; we treat it as constant with respect to θ and route all gradients through the continuous distance $d(x, \mathcal{P}_p)$. This is justified because s_p changes only when the polyline boundary crosses the pixel, a codimension-1 event in parameter space (Appendix A.4). At the boundary itself, the kernel ϕ already provides the relevant gradients for that transition. This design is consistent with the observed S -equivariance in closed-mode optimization: once the polyline boundary reaches sufficient fidelity, the winding-number fill is already correct for nearly all pixels, and learning is driven primarily by boundary motion under the prefiltered band.

3.4 Compositing and Visibility-Driven Adaptation

We render primitives in front-to-back order via standard alpha compositing with initial transmittance $T_0(x) = 1$ and per-primitive color c_i :

$$C(x) = \sum_{i=1}^N T_{i-1}(x) \alpha_i(x) c_i, \quad T_i(x) = T_{i-1}(x) (1 - \alpha_i(x)). \quad (7)$$

A key design principle is to avoid explicit geometric constraints on the parametric curves (e.g. convexity or non-self-intersection penalties), which would introduce nontrivial auxiliary losses and complicate the optimization landscape. Instead, we allow arbitrary curve configurations to emerge freely during training and detect degeneracy where it matters by inspecting inexpensive, renderer-derived signals. Rasterization already exposes per-pixel transmittance $T_{i-1}(x)$ and per-primitive contribution $T_{i-1}(x) \alpha_i(x)$ as intermediate quantities. We aggregate these into a per-primitive importance score:

$$w_p \propto \sum_x T_{p-1}(x) \alpha_p(x), \quad (8)$$

optionally combined with a visibility rate and a geometric coverage estimate. All terms are renderer-derived and available at negligible overhead.

Rather than regularizing geometry, we monitor how concentrated importance is across primitives. When a small subset dominates, as quantified by a dispersion metric such as the Gini coefficient over $\{w_p\}$, we prune primitives whose w_p remains persistently small, directly removing degenerate or fully occluded elements from the representation. Freed capacity can optionally be reallocated by reinitializing new primitives near regions of high residual error, following a strategy similar to LIVE [26].

4 Experiments

We evaluate CubicSplat on image vectorization, focusing on (i) reconstruction quality for a fixed primitive budget and (ii) efficiency in both single forward-backward pass and end-to-end training time.

4.1 Datasets and Implementation Details

We adopt the large-scale evaluation protocol introduced by Bézier Splatting [23], which moved beyond the few low-resolution test images common in earlier work to diverse, high-resolution benchmarks. Specifically, we evaluate on a 200-image subset of DIV2K [1] and the full 24-image Kodak dataset [6], all at original resolution. We report PSNR, SSIM, and LPIPS [42], and compare against DiffVG [20], LIVE [26], LIVSS [34], and Bézier Splatting [23]. Following the standard DiffVG convention, we parameterize each closed primitive as two connected cubic Bézier

Table 1: Single training-step time averaged over a 2040×1344 image with 2048 curves on an RTX 4090.

	Open curves			Closed curves		
	DiffVG	Bézier Splatting	Ours	DiffVG	Bézier Splatting	Ours
Forward	141.3ms	4.5ms	2.70ms	85.2ms	14.1ms	10.68ms
Backward	701.3ms	4.7ms	4.07ms	448.3ms	24.58ms	8.96ms
Total step	842.6ms	9.2ms	6.77ms	533.5ms	38.68ms	19.64ms



Fig. 2: Qualitative results across varying budgets, topology modes, and datasets [1, 6, 27]. At low budgets (closed 128, open/closed 256), CubicSplat preserves clean boundaries and smooth color gradients, producing coherent stylization rather than noisy artifacts. At higher budgets (open 1024, closed 2048), fine details such as the half-timbered façade emerge faithfully, confirming that the gradient oracle scales from compact stylization to high-fidelity reconstruction.

segments and each open primitive as three connected cubic Bézier segments, using 10 control points in both cases. We optimize control points, color, width, and opacity with Adan [36] under an ℓ_2 loss, using learning rates of 0.1 for color and opacity, 10^{-3} for control points, and 10^{-4} for width. We train for 12,000 steps in open mode and 10,000 steps in closed mode without early stopping; all baselines use their respective default configurations. The default polyline sampling density is $S=24$ for open curves and $S=12$ for closed curves; the quality–speed trade-off across S is analyzed in Sec. 4.4. Primitives are pruned and immediately densified every 500 steps when the Gini coefficient exceeds 0.35, with pruning disabled for the final 2,000 steps. At evaluation time, we render with $S^*=24$ or the training density, whichever is larger, ensuring that the forward surrogate is at least as accurate as during optimization.

4.2 Single Training-step Speed

Table 1 reports single training-step times on a 2040×1344 image with 2048 curves at default sampling densities. On open curves, CubicSplat completes a forward–backward pass in 6.77 ms, $1.4\times$ faster than Bézier Splatting and over $120\times$ faster than DiffVG. On closed curves the gap over Bézier Splatting widens to roughly $2\times$, while DiffVG remains over an order of magnitude slower. The speed advantage stems from representation efficiency: Bézier Splatting discretizes each curve

Table 2: Quantitative evaluation on the DIV2K dataset [1] (200 images). Best results in **bold**, second-best underlined. Training time is the average per image on an RTX 4090. CubicSplat achieves the highest PSNR and SSIM across all budgets and both modes while training in under 2.1 min per image; in closed mode at $N=1024$ it surpasses Bézier Splatting by over 2 dB PSNR at roughly $4\times$ lower cost.

	Method	256 curves				512 curves				1024 curves			
		SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	Time $\downarrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	Time $\downarrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	Time $\downarrow$
Open	DiffVG [20]	0.552	19.83	0.563	18.9min	0.587	21.47	0.537	22.0min	0.616	22.62	0.517	30.6min
	Bézier Splatting [23]	<u>0.600</u>	<u>22.17</u>	0.540	<u>3.4min</u>	<u>0.646</u>	<u>23.79</u>	0.498	<u>3.3min</u>	<u>0.699</u>	<u>25.45</u>	0.448	<u>3.2min</u>
	CubicSplat (Ours, $S=24$)	0.624	23.07	0.589	1.8min	0.662	24.43	0.548	1.8min	0.708	25.93	<u>0.502</u>	1.8min
Closed	DiffVG [20]	0.578	20.69	0.548	16.4min	0.601	21.82	0.531	18.5min	0.631	22.95	0.509	25.1min
	LIVE [26]	0.576	20.09	<u>0.543</u>	2.6h	0.611	21.70	0.521	4.2h	0.648	23.11	0.495	5.1h
	LIVSS [34]	<u>0.586</u>	17.71	0.542	39.2min	<u>0.630</u>	18.71	0.530	54.3min	<u>0.678</u>	19.83	0.517	1.4h
	Bézier Splatting [23]	0.580	<u>20.74</u>	0.546	<u>7.8min</u>	0.607	<u>22.11</u>	<u>0.528</u>	<u>8.3min</u>	0.639	<u>23.45</u>	<u>0.507</u>	<u>8.6min</u>
	CubicSplat (Ours, $S=12$)	0.629	23.00	0.580	1.8min	0.664	24.31	0.547	1.9min	0.705	25.78	0.508	2.1min

Table 3: A quantitative evaluation on the Kodak dataset [6] with 256 to 1024 curves on RTX 4090. CubicSplat leads in PSNR and SSIM across all settings and achieves the best LPIPS in closed mode.

	Method	256			512			1024		
		SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$
Open	DiffVG	0.601	23.43	0.535	0.645	24.70	0.495	0.699	26.04	0.439
	Bézier Splatting	0.679	26.18	0.457	0.743	27.90	0.383	0.797	29.24	0.310
	CubicSplat(Ours, $S=24$)	0.681	26.75	0.498	0.740	28.28	0.432	0.803	29.94	0.355
Closed	DiffVG	0.622	24.11	0.513	0.666	25.34	0.475	0.719	26.66	0.420
	Bézier Splatting	0.621	24.19	0.519	0.664	25.61	0.485	0.708	26.91	0.448
	CubicSplat(Ours, $S=12$)	0.680	26.64	0.507	0.730	28.06	0.454	0.786	29.66	0.391



Fig. 3: Qualitative comparison with state-of-the-art differentiable VG rasterization methods, including DiffVG [20], LIVE [26], LIVSS [34], and Bézier Splatting [23]. The lake scene illustrates gradient and reflection reconstruction, while the dog example highlights preservation of fine details such as eye color and facial structure.

into densely sampled Gaussian primitives that can reach millions in closed-fill settings, whereas CubicSplat requires only a modest number of polyline segments per curve with closed-form distance queries, avoiding both dense splat accumulation and iterative solvers.

4.3 Quantitative Results

Table 2 and Table 3 summarize reconstruction quality and training cost for various primitive counts under both modes. **CubicSplat establishes new state-of-the-art** PSNR and SSIM while substantially reducing end-to-end training

time. The gains are consistent in the open setting and become decisive for closed shapes, where accurate filling and stable gradients are critical.

CubicSplat is also more parameter-efficient: visibility-driven pruning removes redundant and occluded curves, concentrating capacity on primitives that actually contribute to the image. As a result, CubicSplat with 256 curves already matches the PSNR of DiffVG with 1024 curves in both modes (Tab. 2).

The improvements are most pronounced in the closed mode. At $N=1024$, CubicSplat delivers more than 2 dB PSNR improvement over Bézier Splatting while cutting per-image training time by roughly $4\times$, demonstrating that the quality gains accompany rather than trade against efficiency. In the open mode, CubicSplat still improves PSNR and SSIM at every budget and reduces training time markedly, yielding a strictly better quality–time trade-off.

We observe that Bézier Splatting can achieve lower LPIPS for open strokes despite worse PSNR and SSIM. We view this as a difference in failure mode, not fidelity. CubicSplat’s error-bounded surrogate yields clean, piecewise-smooth coverage with sharp, scale-consistent boundaries (Fig. 4); at low budgets this produces faithful but stylized results that under-represent photographic micro-texture. Sample-based splatting instead introduces stochastic high-frequency fluctuations that can incidentally mimic fine texture (Appendix A.3), which LPIPS, itself trained on natural-image statistics, may favor despite lower pixel accuracy. As all methods optimize the same ℓ_2 loss, the gap reflects a metric preference: LPIPS can reward texture-like variation over the clean, scale-consistent geometry targeted by low-budget vectorization.

4.4 Ablations and Analysis

Table 4 isolates four design choices on the uniformly sampled DIV2K subset under both topology modes and curve budgets $N \in \{256, 512, 1024\}$. Taken together, the ablations support our central thesis: CubicSplat is an error-controlled forward relaxation that improves the optimization oracle, and pushing forward geometric exactness beyond a non-degeneracy threshold yields diminishing returns in quality while imposing substantial performance costs.

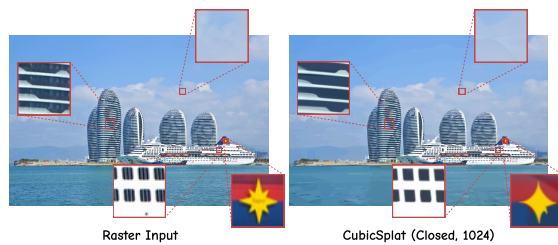


Fig. 4: Close-up detail comparison. Raster input (left) and CubicSplat reconstruction with 1024 closed fills (right). Insets highlight preserved sharp architectural edges, coherent layer structure, and clean color boundaries using only parametric vector primitives, yielding stable fine details under magnification.

When the curve budget or sampling density increases, additional work can be partially absorbed by otherwise idle execution resources, producing sublinear scaling in observed end-to-end time. Reporting training throughput in tasks-per-minute (TPM) makes this effect explicit and enables fair comparison across settings where hardware utilization differs.

Sampling density S validates forward relaxation. Increasing S monotonically reduces geometric surrogate error and, in the limit, recovers an **exact** Bézier renderer; yet the learned solutions exhibit a broad equivariance regime once the surrogate is non-degenerate. In open mode, gains saturate beyond $S=24$ with negligible throughput change. In closed mode, saturation occurs even earlier because winding-number membership already provides exact interior labeling; $S=12$ attains essentially the same quality as $S=24$ at substantially higher throughput. These trends confirm that S functions as a relaxation knob rather than an approximation budget, consistent with our error bound (Appendix A.4), which guarantees decreasing forward bias but does not imply that optimization quality must improve once the oracle is well conditioned.

Visibility-driven Gini pruning is essential at large N . Removing Gini pruning degrades PSNR increasingly with larger curve budgets in both topology modes. This aligns with the oracle perspective: as N grows, occlusion depth increases and more primitives become weakly visible or redundant, amplifying gradient interference and wasting compute. By explicitly tracking contribution and visibility and pruning low-importance primitives, CubicSplat maintains an effective representation size and preserves gradient quality, with the benefit growing as N increases.

Regularization reduces expressivity and is unnecessary under oracle stability. Adding a convexity-oriented regularizer consistently decreases reconstruction quality across all budgets and both modes. This indicates that the optimal vector representation for natural images is often non-convex and geometrically complex; restricting the hypothesis class harms expressivity with-

Table 4: Ablation on the uniformly sampled DIV2K subset [1] (200 images). We report SSIM/PSNR/LPIPS and training throughput in Tasks Per Minute of a single GPU (TPM, higher is better), measured with 2 workers per GPU. Best results in **bold**, second-best underlined.

	Setting	256 curves				512 curves				1024 curves			
		SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	TPM $\uparrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	TPM $\uparrow$	SSIM $\uparrow$	PSNR $\uparrow$	LPIPS $\downarrow$	TPM $\uparrow$
Open	$S = 24$	0.624	23.07	0.589	0.861	0.662	24.43	0.548	0.883	0.708	<u>25.93</u>	0.502	0.873
	w. Regularization	0.614	22.60	0.597	<u>0.894</u>	0.650	23.89	0.561	<u>0.938</u>	0.693	25.28	0.522	<u>0.917</u>
	w.o Gini-Pruning	0.611	22.38	0.596	0.971	0.645	23.56	0.554	0.982	0.686	24.83	0.508	0.952
	$S = 32$	0.624	<u>23.06</u>	0.589	0.859	0.662	<u>24.42</u>	0.548	0.868	0.708	25.94	0.502	0.830
	$S = 12$	0.622	22.93	0.591	0.814	0.658	24.27	0.550	0.794	0.704	25.75	0.504	0.832
	$S = 8$	0.615	22.43	0.595	0.753	0.649	23.64	0.556	0.775	0.692	24.97	0.511	0.832
	de Casteljau	0.623	23.01	0.590	0.645	0.660	24.37	0.549	0.631	0.706	25.87	0.503	0.633
Closed	$S = 24$	0.630	23.01	0.580	0.572	0.664	24.33	0.547	0.481	0.706	25.80	0.508	0.413
	w. Regularization	0.620	22.63	0.588	0.650	0.653	23.95	0.558	0.578	0.694	25.38	0.521	0.503
	w.o Gini-Pruning	0.616	22.21	0.589	0.546	0.643	23.22	0.561	0.456	0.672	24.21	0.531	0.390
	$S = 32$	0.630	23.01	0.580	0.461	0.664	24.33	0.547	0.387	0.706	25.80	0.508	0.317
	$S = 12$	0.629	22.99	0.580	<u>0.879</u>	0.664	24.32	0.547	<u>0.786</u>	0.705	25.78	0.509	<u>0.654</u>
	$S = 8$	0.628	22.95	0.581	0.975	0.662	24.25	0.548	0.914	0.704	25.70	0.510	0.826
	de Casteljau	0.629	22.97	0.581	0.517	0.663	24.29	0.548	0.531	0.704	25.73	0.510	0.481

out a compensating stability benefit. In CubicSplat, stability derives from the relaxed oracle and visibility-aware adaptation: pathological or redundant curves are automatically identified by low visibility and pruned, making an explicit regularizer both unnecessary and counterproductive.

Adaptive subdivision underperforms due to graph discontinuities.

De Casteljaeu subdivision reduces forward geometric error but introduces discrete split events that alter the computation graph. The ablation shows that this strategy trails fixed- S sampling in both quality and throughput, consistent with our analysis that subdivision boundaries induce gradient discontinuities and heteroscedastic updates (Appendix B). Fixed- S relaxation preserves a stable graph and yields more reliable gradients, even when its forward approximation error is formally larger.

4.5 Scaling Behavior

The error-controlled relaxation analyzed in Sec. 4.4 has direct consequences for scalability: because a low sampling density S is not an engineering compromise but a mathematically bounded surrogate with negligible quality loss, CubicSplat can allocate its compute budget almost entirely to geometric expressiveness (more primitives, higher resolution) rather than to tighter curve approximation.

Quality is resolution-invariant up to extreme magnification. We train on 5 DIV2K [1] images using 1024 curves in both modes and evaluate forward rendering up to $144\times$ more pixels on a single RTX 4090, computing PSNR after downsampling to original resolution (Appendix D). Render time scales as $t \propto s^2$ with resolution scale s in Fig. 5(a), while PSNR stays within 0.5 dB of the 2K baseline across all resolutions, sampling densities, and both modes. Figure 5(b) confirms that quality is governed by spline geometry rather than surrogate discretization: increasing S from 12 to 48 in closed mode changes PSNR by less than 0.03 dB at $\sim 4\times$ the cost, consistent with the $\mathcal{O}(S^{-2})$ bound. At native 2K, $S=8$ reaches near-saturated quality at 2.9 ms (25.11 dB, closed), leaving headroom

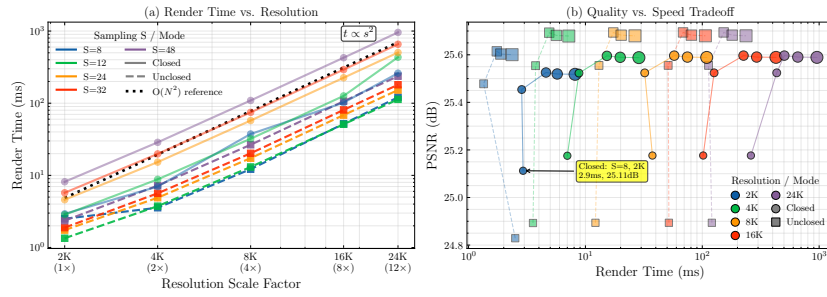


Fig. 5: Resolution scalability of CubicSplat. (a) Render time versus resolution scale at different sampling densities S . (b) PSNR versus render time for all tested (resolution, S) configurations. At native 2K, $S=8$ achieves near-optimal quality in 2.9 ms; increasing S to 48 yields <0.03 dB gain at $4\times$ the cost. Original data refers to Appendix D.

for real-time use. This robustness contrasts with point-sampled splatting, where discrete sampling introduces phase-sensitive artifacts under magnification.

Optimization remains well conditioned at high primitive counts. We vary the number of closed curves N from 2048 to 32768 on a detailed illustration (Fig. 6). PSNR improves consistently from 24.45 dB to 38.09 dB, indicating that the gradient oracle does not degrade as scene complexity grows. Training time scales sublinearly (82.5 s to 220.8 s) thanks to the tile-parallel architecture and visibility-based pruning, while video memory grows from 632 MB to 2118 MB. The combination of low per-primitive cost from the polyline surrogate and effective pruning from compositing-derived visibility statistics allows CubicSplat to operate comfortably at 32K primitives on a single consumer GPU.

5 Conclusion

We introduce CubicSplat, a scalable differentiable vector rasterizer that addresses the gradient seesaw through error-bounded forward relaxation. By replacing Bézier closest-point root solving with closed-form polyline distance queries, CubicSplat yields a stable, static computation graph while maintaining a tight $\mathcal{O}(S^{-2})$ geometric error bound. Experiments on large-scale benchmarks confirmed that this controlled relaxation does not sacrifice reconstruction quality: the better-conditioned gradient oracle consistently produces superior optimization outcomes at a fraction of the computational cost.

These efficiency characteristics suggest two natural extensions. The tile-parallel architecture and lightweight per-primitive cost bring **real-time differentiable vector rendering within reach**, enabling interactive design tools that optimize on the fly at display rates [5]. The low per-frame cost also **opens the door to video vectorization**, where temporal coherence demands both fast rendering and stable gradients under continuous motion. More broadly, the gradient seesaw perspective may offer a useful lens for other differentiable rendering settings where forward exactness is pursued at the expense of optimization quality.



Fig. 6: Scaling with closed-fill count N on a detailed illustration from the DanbooRegion dataset [40]. Insets rendered at $8\times$ magnification show **scale-consistent** fine structures under extreme primitive counts. PSNR improves steadily with sublinear training time and manageable memory overhead.

Combining CubicSplat with perceptual losses, semantic constraints, and learned priors [2,10,11,13,13,19,21,37,38,41] is a promising direction toward controllable, real-time vector-content creation.

Acknowledgements

This work was supported by grants from the National Natural Science Foundation of China (62525606, U25B2072) and Open Research Fund of Zhejiang Key Laboratory of Intelligent Education Technology and Application (No. 2025ZN-JYKF006).

References

1. Agustsson, E., Timofte, R.: Ntire 2017 challenge on single image super-resolution: Dataset and study. In: 2017 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: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10093–10102 (2023). <https://doi.org/10.1109/CVPR52729.2023.00973>
3. Celarek, A., Kopanas, G., Drettakis, G., Wimmer, M., Kerbl, B.: Does 3d gaussian splatting need accurate volumetric rendering? In: Computer Graphics Forum. vol. 44, p. e70032. Wiley Online Library (2025)
4. Clarke, F.H.: Optimization and nonsmooth analysis. SIAM (1990)
5. Das, A., Yang, Y., Hospedales, T., Xiang, T., Song, Y.Z.: Béziersketch: A generative model for scalable vector sketches. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. pp. 632–647. Springer International Publishing, Cham (2020)
6. Eastman Kodak Company: Kodak lossless true color image suite (1999), <http://r0k.us/graphics/kodak/>, dataset of 24-bit lossless images released by Eastman Kodak Company:contentReference[oaicite:34]index=34
7. Farin, G.E.: Curves and surfaces for CAGD: a practical guide. Morgan Kaufmann (2002)
8. Guo, M., Wang, B., He, K., Matusik, W.: Tetsphere splatting: Representing high-quality geometry with lagrangian volumetric meshes **2025**, 30290–30314 (2025), https://proceedings.iclr.cc/paper_files/paper/2025/file/4ae7d78ebbe48f772e31c5c3fcc04c43-Paper-Conference.pdf
9. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5354–5363 (2024). <https://doi.org/10.1109/CVPR52733.2024.00512>
10. Hirschorn, O., Jevnisek, A., Avidan, S.: Optimize & reduce: a top-down approach for image vectorization. AAAI’24/IAAI’24/EAAI’24, AAAI Press (2024). <https://doi.org/10.1609/aaai.v38i3.27987>
11. Hu, T., Yi, R., Qian, B., Zhang, J., Rosin, P.L., Lai, Y.K.: Supersvg: Superpixel-based scalable vector graphics synthesis. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 24892–24901 (2024). <https://doi.org/10.1109/CVPR52733.2024.02351>

12. Huang, B., Yu, Z., Chen, A., Geiger, A., Gao, S.: 2d gaussian splatting for geometrically accurate radiance fields. In: ACM SIGGRAPH 2024 Conference Papers. SIGGRAPH '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3641519.3657428>
13. Jain, A., Xie, A., Abbeel, P.: Vectorfusion: Text-to-svg by abstracting pixel-based diffusion models. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1911–1920 (2023). <https://doi.org/10.1109/CVPR52729.2023.00190>
14. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3907–3916 (2018). <https://doi.org/10.1109/CVPR.2018.00411>
15. Katsumata, K., Vo, D.M., Nakayama, H.: A compact dynamic 3d gaussian representation for real-time dynamic view synthesis. In: Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXXVI. p. 394–412. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-73016-0_23
16. Kerbl, B., Kopanas, G., Leimkuehler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* **42**(4) (Jul 2023). <https://doi.org/10.1145/3592433>
17. Keselman, L., Hebert, M.: Approximate differentiable rendering with algebraic surfaces. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) *Computer Vision – ECCV 2022*. pp. 596–614. Springer Nature Switzerland, Cham (2022)
18. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph.* **39**(6) (Nov 2020). <https://doi.org/10.1145/3414685.3417861>
19. Law, H., Kang, S.H.: Image vectorization with depth: convexified shape layers with depth ordering. *SIAM Journal on Imaging Sciences* **18**(2), 963–1001 (2025)
20. Li, T.M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph.* **39**(6) (Nov 2020). <https://doi.org/10.1145/3414685.3417871>
21. Li, W., Zhou, R., Zhou, J., Song, Y., Herter, J., Qin, M., Huang, G., Pfister, H.: 4d langspat: 4d language gaussian splatting via multimodal large language models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2025)
22. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 7707–7716 (2019). <https://doi.org/10.1109/ICCV.2019.00780>
23. Liu, X., Zhou, C., Zhao, N., Huang, S.: Bezier splatting for fast and differentiable vector graphics rendering. *arXiv preprint arXiv:2503.16424* (2025)
24. Loper, M.M., Black, M.J.: Opendr: An approximate differentiable renderer. In: *European Conference on Computer Vision*. pp. 154–169. Springer (2014)
25. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20654–20664 (2024). <https://doi.org/10.1109/CVPR52733.2024.01952>
26. 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*. pp. 16314–16323 (2022)

27. Fluent emoji, <https://github.com/microsoft/fluentui-emoji> Accessed: 2026-02-20
28. Müller, J.U., Weinmann, M., Klein, R.: Unbiased gradient estimation for differentiable surface splatting via poisson sampling. In: Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII. p. 281–299. Springer-Verlag, Berlin, Heidelberg (2022), https://doi.org/10.1007/978-3-031-19827-4_17
29. Radl, L., Steiner, M., Parger, M., Weinrauch, A., Kerbl, B., Steinberger, M.: Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering. *ACM Trans. Graph.* **43**(4) (Jul 2024). <https://doi.org/10.1145/3658187>
30. Reddy, P., Gharbi, M., Lukáč, M., Mitra, N.J.: Im2vec: Synthesizing vector graphics without vector supervision. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7338–7347 (2021). <https://doi.org/10.1109/CVPR46437.2021.00726>
31. Rockafellar, R.T., Wets, R.J.: Variational analysis. Springer (1998)
32. Rota Bulò, S., Porzi, L., Kotschieder, P.: Revising densification in gaussian splatting pp. 347–362 (2024)
33. Vinker, Y., Pajouheshgar, E., Bo, J.Y., Bachmann, R.C., Bermano, A.H., Cohen-Or, D., Zamir, A., Shamir, A.: Clipasso: semantically-aware object sketching. vol. 41. Association for Computing Machinery, New York, NY, USA (Jul 2022). <https://doi.org/10.1145/3528223.3530068>
34. Wang, Z., Huang, J., Sun, Z., Gong, Y., Cohen-Or, D., Lu, M.: Layered image vectorization via semantic simplification pp. 7728–7738 (2025). <https://doi.org/10.1109/CVPR52734.2025.00724>
35. Wang, Z., Deng, X., Zhang, Z., Jakob, W., Marschner, S.: A simple approach to differentiable rendering of sdfs (2024). <https://doi.org/10.1145/3680528.3687573>
36. 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>
37. Xing, X., Wang, C., Zhou, H., Zhang, J., Yu, Q., Xu, D.: Diffsketcher: text guided vector sketch synthesis through latent diffusion models. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. NIPS ’23, Curran Associates Inc., Red Hook, NY, USA (2023)
38. 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. pp. 4546–4555 (2024)
39. Ye, Z., Li, W., Liu, S., Qiao, P., Dou, Y.: Absgs: Recovering fine details in 3d gaussian splatting (2024). <https://doi.org/10.1145/3664647.3681361>
40. Zhang, L., Ji, Y., Liu, C.: Danbooregion: An illustration region dataset (2020)
41. Zhang, P., Zhao, N., Liao, J.: Text-to-vector generation with neural path representation. *ACM Trans. Graph.* **43**(4) (Jul 2024). <https://doi.org/10.1145/3658204>
42. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 586–595 (2018). <https://doi.org/10.1109/CVPR.2018.00068>
43. 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 p. 327–345 (2024). https://doi.org/10.1007/978-3-031-72673-6_18

44. 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 (2025). <https://doi.org/10.1145/3721238.3730596>
45. Zhou, H., Zhang, H., Wang, B.: Segmentation-guided layer-wise image vectorization with gradient fills. In: Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part X. p. 165–180. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-72684-2_10