DOI: 10.1111/cgf.70375

EUROGRAPHICS 2026 / B. Masia and J. Thies
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 45 (2026), Number 2

Layer3D: A 3D Layered Representation for Multiview Vector Graphics

Zhongyue Guan[1,2], Yixin Hu[2], and Zeyu Wang[1,3†]

[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]Tencent [3]The Hong Kong University of Science and Technology

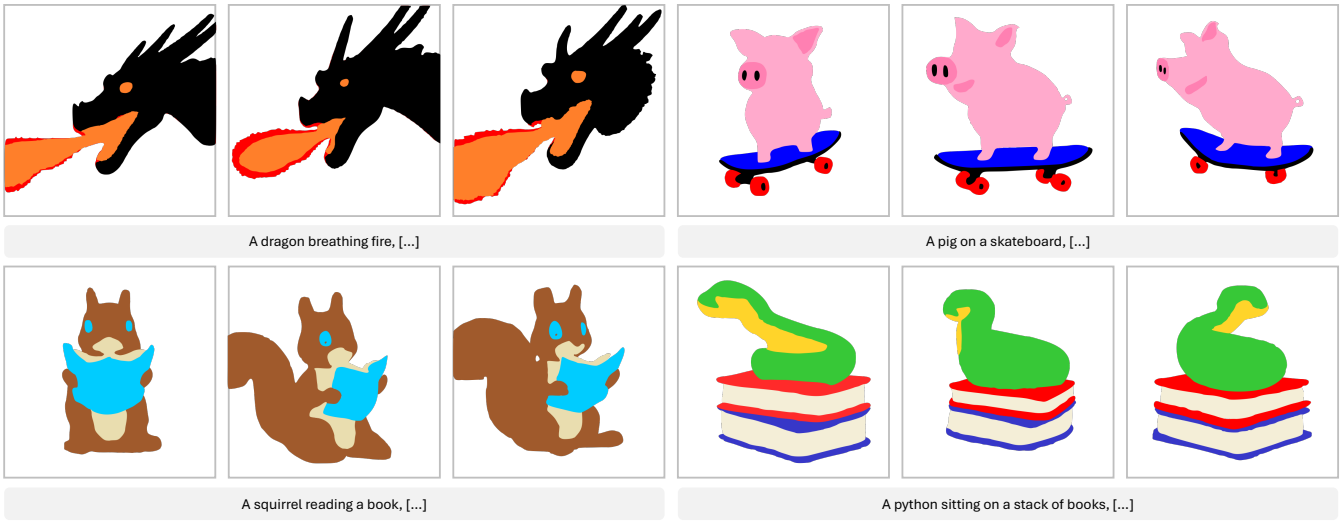


Figure 1: *Generated results by Layer3D. Given a text prompt, our method extracts multiview vector graphics from 3D neural primitives.*

Abstract

We present Layer3D, a novel 3D neural representation that models objects as collections of decomposable neural implicit primitives. These primitives enable the generation of layered images with consistent correspondences across viewpoints, establishing a flexible framework for multiview vector graphics decomposition. Integrated into a text-to-3D pipeline via Score Distillation Sampling (SDS), Layer3D learns to generate primitives with diverse shape topologies while preserving structural coherence. To ensure front-to-back ordering required for 2D flat graphics, our method incorporates front-to-back rendering and shape regularization constraints. Experimental results demonstrate that Layer3D consistently produces meaningful, topology-diverse layers across multiple views, thereby facilitating intuitive and effective layer-based vector editing.

CCS Concepts

• ***Computing methodologies*** → ***Computer graphics; Non-photorealistic rendering;*** *Shape analysis; Image processing;*

1. Introduction

Over the past few decades, vector graphics have become a cornerstone of digital design, providing a flexible and scalable means of representing visual content. Unlike raster images, which rely on discrete pixels, vector graphics employ geometric primitives such as Bézier curves and polygons. Designers can manipulate these primitives through control points, enabling precise editing and efficient reuse across varying resolutions. To reduce the burden of manually constructing primitives from scratch, researchers have developed a range of automated generation techniques based on diverse input modalities. Early work explored converting meshes

† Corresponding author.

into vector representations [EWHS08, EPD09], while subsequent approaches advanced image vectorization [MZX*22, RGLM21, FLB17, DKT*23, LLGRK20] and, more recently, text-to-vector generation [JXA22, ZZL24, TLF*24, PAR*25, XZW*24, FSW22]. These developments have lowered the barrier to entry for novices and provided professionals with convenient starting templates.

Despite these advances, generating vector graphics that remain consistent under viewpoint changes remains a significant challenge. Even minor shifts in viewing direction require designers to adjust control points, primitives, and layer ordering to maintain coherence across multiple views. We identify two core difficulties in this process: (1) multiview shape optimization and (2) multiview layer ordering. Several prior works partially address these challenges. Eisemann et al. [EWHS08] propose a rendering framework that converts 3D meshes into stylized 2D vector art; however, the quality of the resulting output depends substantially on the fidelity of the input mesh. 2.5D Cartoon Models [RID10] introduce a novel data structure for multiview vector graphics, but their system requires extensive manual anchoring of orthogonal views, increasing the learning barrier for new users. Dream3DVG [LXL*25] generates 3D vector graphics from text prompts, yet suffers from persistent layer fragmentation issues similar to [JXA22, XZW*24, FSW22], which complicate subsequent editing. Adobe Turntable [Ado24] demonstrates multiview vector generation capabilities, though its technical details remain undisclosed.

To overcome these limitations, we introduce **Layer3D**, a 3D layered neural implicit representation. Layer3D decomposes objects into meaningful, primitive-based layers within a 3D context, enabling consistent 2D vector graphics across multiple viewing directions. Building on the idea of 2D neural implicit vector layers (NIVeL) [TLF*24], we represent each 3D subject as a set of neural implicit primitives, characterized by per-primitive volume density and color, in contrast to the per-pixel coloring used in Neural Radiance Fields (NeRF) [MST*20]. To enforce front-to-back ordering in a 3D context, we incorporate a depth-aware strategy based on NeuralSort [GWZE19], and introduce shape regularization terms to constrain ambiguous z-order configurations. From different viewing directions, Layer3D produces multiview layer images that are easily vectorized using standard curve extraction techniques. Integrated into a text-to-3D pipeline based on Score Distillation Sampling (SDS) [PJBM22], our approach consistently generates diverse, topology-preserving layers that are practical for downstream vector editing.

In summary, our contributions are threefold:

- We introduce a 3D neural representation that models objects as decomposable, primitive-based layers.
- We design a generative pipeline that supervises decomposable primitive generation from text prompts using front-to-back rendering and shape regularization.
- We empirically demonstrate that our approach yields consistent multiview vector graphics, with intermediate layers offering tangible benefits for vector editing workflows.

2. Related Work

As our primary focus is on multiview vector graphics, we review prior work across three relevant research directions. We first review 2D vector graphics synthesis (Section 2.1), and then discuss non-photorealistic rendering approaches that leverage 3D or multiview inputs to produce view-specific or novel-view vector graphics (Section 2.2). We also summarize recent advances in general 3D generation and more specific studies on generating 3D sketches with Bézier curves, which enable multiview rendering (Section 2.3).

2.1. Vector Graphics Synthesis

Various representations of vector graphics have been proposed for 2D illustration and animation [Wor18, BG89, ASP07, OBB*13, DRvdP14, DRvdP15]. The most commonly used approach, particularly prevalent in digital art and web development, represents images as stacked layers of primitives defined by specific properties and composition methods. This is exemplified by the Scalable Vector Graphics (SVG) format, which supports a diverse range of primitive types. This format facilitates the creation of high-quality, resolution-independent graphics that can seamlessly scale across varying display sizes.

Image vectorization. Beyond the creation of tools and user interfaces for generating vector graphics, researchers have explored methods to derive vector graphics from raster images through vectorization techniques. Previous image vectorization methods often approach the task as a clustering and curve extraction problem [Web25, Cor25, Sel25]. This line of work is effective at dividing pixel clusters into non-overlapping shapes based on color segmentation. However, these methods overlook the layered structure inherent to the SVG format, leading to SVGs composed of fragmented elements that are challenging to edit.

The differentiable rasterizer for vector graphics (DiffVG) proposed by Li et al. [LLGRK20], enables a learning-based image vectorization method with raster-level supervision. Xu et al. proposed LIVE [MZX*22] to progressively generate vector graphics in a layer-wise fashion. Nevertheless, the issue of fragmented elements persists. Other works have specifically addressed the layer decomposition problem [FLB17, DKT*23], but they heavily rely on segmentation masks as additional input to constrain the solution space. Recently, StarVector [RPA*24] leveraged the code generation capabilities of Multimodal Large Language Models (MLLMs), framing image vectorization as an inverse rendering and code generation task to overcome the primitive constraints. In contrast, our method addresses the decomposition problem by storing distinct primitives at the raster level and individually vectorizing each rendered layer in the final step, thereby reducing the complexity of the vectorization process. We demonstrate that the raster layers generated by our approach can be effectively vectorized using common image vectorization techniques.

Text-to-vector generation. DiffVG [LLGRK20] also enables direct optimization of the control points of each primitive under the supervision of models such as CLIP [RKH*21] or diffusion models [RBL*21], thereby unlocking new possibilities for text-to-vector generation [FSW22, JXA22, XZW*24]. However, these

methods also share the aforementioned limitation: the generated results are difficult to edit. The primitives tend to resemble colorful brush strokes that collectively compose the final outcome rather than meaningful layers designed with a creator's intent.

To produce vector graphics with layer-wise high quality, several neural implicit representations for the vector layers have been proposed. Zhang et al. [ZZL24] introduce Neural Path Representation, a dual-branch Variational Autoencoder (VAE) trained on a vector dataset. NIVeL [TLF*24] and NeuralSVG [PAR*25] both propose neural implicit vector layers, utilizing multilayer perceptrons (MLPs) to represent geometric shapes across different layers. NIVeL employs a pixel-based representation that facilitates the extraction of curves, accommodating shapes with arbitrary topology and genus. Apart from vector graphics generation, editing poses another challenge. Zhang et al. [ZZL23] address this by enabling the modification o f v ector c ontent a ppearance o r s cenario u sing text prompts. However, their method does not support changes in viewing direction. Drawing inspiration from NIVeL, we extend its pixel-based representation into the domain of 3D generation, enabling the creation of multiview vector graphics.

2.2. Non-Photorealistic Rendering in Vector Graphics

A number of non-photorealistic rendering techniques have been developed to export vector graphics from 3D models under specific viewpoints. Several object-space algorithms, for example, generate line drawings as polylines or line segments directly from 3D models across arbitrary camera configurations [DFRS03, JDA07, OBS04, LBHH23]. These methods focus primarily on extracting geometric feature lines to enhance shape depiction, but they generally neglect filled regions and the layer ordering of the elements.

To address filled r egions, p rior w ork h as p ursued t wo main representations: planar maps and stacking layers. Planar map approaches [BG89, ASP07] place all primitives on a single layer, treating regions as faces bounded by vector contours. Building on this paradigm, Eisemann et al. [EWHS08] converted 3D meshes into stylized 2D vector art with smooth contour boundaries, while subsequent work introduced closed-loop extraction techniques to support consistent multi-frame renderings [KH11].

In contrast, stacking-layer approaches decompose scenes into multiple ordered layers, which aligns more closely with our interest. Stroila et al. [SEH08] generated layered closed curves that capture silhouettes, shadows, and highlights from implicit surfaces. Eisemann et al. [EPD09] proposed a visibility-based decomposition of 3D meshes for flat vector rendering, although their method does not ensure layer consistency across viewpoints. Rivers et al. [RID10] introduced 2.5D cartoon models in which vector planes interpolate between user-specified 3D positions and depth orders, enabling 3D rotation effects under varying camera perspectives.

Building on the stacking-layer paradigm, we extend this representation into a fully three-dimensional neural framework. Our method decomposes the 3D representation into a collection of primitives that can be rendered in linear order across different viewpoints, with each layer subsequently vectorized to produce the final vector graphics output.

(a) Composition (b) Part-based (c) Stacking-based

Figure 2: *Part-based layers vs. stacking-based layers. Given the same composition (a), it can be decomposed into part-based layers (b), where components are arranged with minimal overlap, or stacking-based layers (c), where components are represented through overlaid regions.*

2.3. 3D Generation

Our work also draws inspiration from recent advances in 3D generation. While prior work attempted to generate and decompose 3D shapes by modeling objects as hierarchies of semantic relations [PKGF21, MGY*19], the rapid progress of diffusion models [RBL*21, Sta23, SWY*23] and novel 3D representations [MST*20, KKLD23, ZTNW23] has significantly advanced this field. Early optimization-based methods relied on 2D priors as multiview supervision. DreamFields [JMB*22], for example, first introduced the use of pre-trained CLIP models to guide NeRF optimization. Building on this, DreamFusion [PJBM22] incorporated 2D diffusion models as multiview supervision through the proposed Score Distillation Sampling (SDS) loss. This formulation opened a new paradigm for 3D generation by transferring knowledge from 2D diffusion priors, inspiring numerous follow-up works across different generation tasks. In line with this direction, our method also employs SDS loss to optimize our neural representation.

While these approaches generally treat the 3D representation as a unified whole, recent research has begun to explore component-level segmentation in 3D generation [LLL*24, CSL*24, CWS*25, MLY*25, LLP*25]. Such methods primarily pursue part-based decomposition, representing objects as assemblies of distinct components. However, this paradigm does not align with the stacking-layer characteristics of vector graphics that our work seeks to preserve (Figure 2).

Another line of research has attempted to reconstruct objects using 3D Bézier curves. Although promising, direct curve-based reconstruction without explicit surfaces often suffers from occlusion artifacts [ZWZ*24, LFT*25, WZLY25], and simple closed curves struggle to represent volumetric forms such as spheres and ellipsoids [CLPK24]. Dream3DVG [LXL*25] addressed some of these limitations through visibility-aware rendering; however, the resulting outputs still contain fragmented elements, a challenge also observed in earlier 2D vector graphics generation methods [XZW*24, JXA22]. Our method adopts a view-dependent layer ordering strategy, and further ensures coherent stacking-layer outputs that are better suited for downstream editing and manipulation.

3. Method

The overall pipeline is illustrated in Figure 3. We organize our method into three components: representation (Section 3.1), ren-

dering (Section 3.2), and optimization (Section 3.3), and describe each in detail below.

3.1. Layered Representation

The objective is to generate a stack of 3D implicit primitives, $\{\mathbf{P}_1,...,\mathbf{P}_N\}$, where the rendering of all primitives from a specific viewpoint can represent the target object. For each viewing direction $\hat{\mathbf{d}}$, each 3D primitive $\mathbf{P}_i$ is rendered as a distinct layer $l_i = \mathcal{R}(\mathbf{P}_i, \hat{\mathbf{d}})$. All layers are then composited according to their order to produce the final target image.

Similar to NeRF [MST*20], we represent the shape of 3D primitives in different layers using a 5D continuous implicit function $\mathcal{F}_s$ with learnable parameters ψ:

$$\mathcal{F}_s(\mathbf{p},\mathbf{d};\psi) : \mathcal{R}^5 \rightarrow [0,1]^N \tag{1}$$

Here, the input 3D point $\mathbf{p} = (x,y,z) \in [0,1]^3$ resides in the unit cube, and $\mathbf{d} = (\theta,\phi)$ is a 2D viewing direction. The function outputs the volume density at a specific spatial location from the given viewing direction, classifying whether the point is part of the shape (denoted value as 1) or absent from it (denoted value as 0) within each of the layers.

Positional Encoding. The network maps each input $\mathbf{i} = (\mathbf{p},\mathbf{d})$ into a higher-dimensional space using positional encoding similar to NeRF [MST*20], defined as follows:

$$\gamma(\mathbf{i}) = [\sin(O\pi\mathbf{i}), \cos(O\pi\mathbf{i})], \text{ where } O = [2^0, 2^1, \ldots, 2^{F-1}]^\top \tag{2}$$

In this equation, F is a hyperparameter that determines the number of octaves in the representation.

Color representation. We store the filled color for each primitive. Specifically, we represent the colors of the N primitives as $c = \{c_1, c_2, ..., c_N\}$, where each c_i corresponds to an RGB value, with $c_i \in [0,1]^3$. To predict the colors, we introduce another implicit function $\mathcal{F}_c$ with learnable parameters ε to optimize the RGB colors for the N layers.

3.2. Front-to-Back Rendering

Unlike 2D vector graphics generation, however, the relative ordering between 3D primitives cannot be predetermined, since occlusion relationships vary with the viewing direction (Figure 4). To resolve this ambiguity, we employ a differentiable soft-sorting operator, NeuralSort [GWZE19], which relaxes the discrete permutation induced by depth ordering into a continuous approximation that remains differentiable.

Given M camera rays and N primitives, our renderer produces a single, view-specific front-to-back order of the rendered layers, while remaining differentiable end-to-end. Each primitive $\mathbf{P}_i$ yields, for every ray $m \in \{1,\ldots,M\}$, an effective per-ray density $A_{i,m} \in [0,1]$ and an expected depth $\bar{D}_{i,m} \in \mathbb{R}_+$ computed from volumetric samples. We pack these into matrices $A \in \mathbb{R}^{M\times N}$ and $D \in \mathbb{R}^{M\times N}$, whose rows correspond to rays and columns to primitives. To obtain a discrete layer order per view, we aggregate per-ray depths into a single score per primitive per view using a visibility

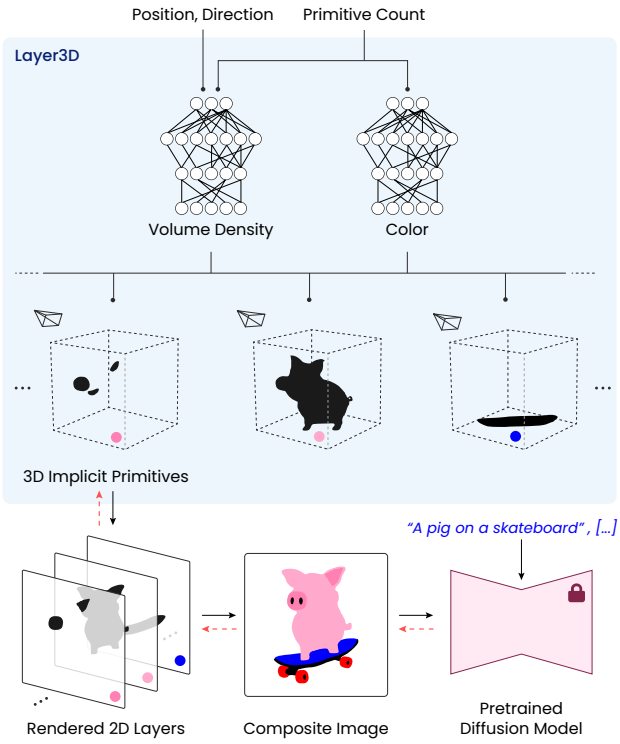


Figure 3: *Overall pipeline. Layer3D represents a subject as a stack of implicit neural primitives, parameterized by two MLPs: one predicts volume densities, and the other assigns per-primitive colors. Given the camera pose, viewing direction, and an upper bound on the primitive count, the primitives are rendered independently into 2D layers and then composited in a differentiable, view-dependent front-to-back order. Each layer can be vectorized using standard techniques. The parameters are optimized via score distillation sampling, guided by a text-conditioned pretrained diffusion model.*

weighting:

$$\bar{D}_i^\star = \frac{\sum_{m=1}^M A_{i,m}\bar{D}_{i,m}}{\sum_{m=1}^M A_{i,m} + \varepsilon}, \qquad s_i = -\bar{D}_i^\star, \tag{3}$$

so primitives that are both close and visible receive larger scores s_i. Let $s = [s_1,\ldots,s_N]^\top$.

We compute a soft permutation $P_\tau(s) \in \mathbb{R}^{N\times N}$ via NeuralSort [GWZE19], a temperature-controlled (τ) differentiable relaxation of sorting. We also form a hard permutation $P_{\text{hard}} \in \{0,1\}^{N\times N}$ by argsort(s) (row k selects the primitive at rank k). To keep forward rendering discrete yet preserve gradients, we use a straight-through estimator:

$$\bar{P} = P_{\text{hard}} + \big(P_\tau(s) - \text{stopgrad}(P_\tau(s))\big). \tag{4}$$

Numerically, $\bar{P}$ equals P_{hard} in the forward pass, while the backward pass takes gradients through $P_\tau(s)$.

For each ray m, we reorder its per-primitive densities with $\bar{P}$:

$$\tilde{\alpha}_{m,:} = \big(\bar{P}A_{m,:}^\top\big)^\top \in [0,1]^N, \tag{5}$$

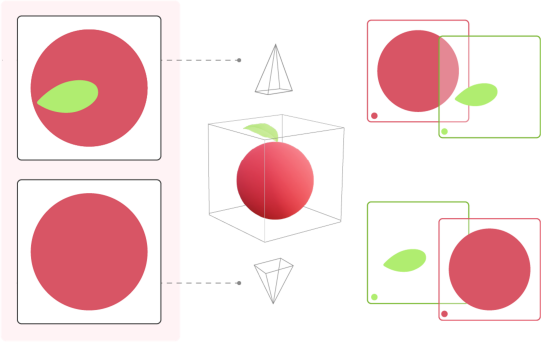
so $\tilde{\alpha}_{n,k}$ is the density of the primitive placed at global rank k. We denote the per-ray RGBs as $C \in \mathbb{R}^{M\times N\times 3}$, where each primitive $\mathbf{P}_i$ contributes a color for every ray m. To avoid any pixel-level mixing, we reorder the colors with the hard permutation consistently across rays:

$$\tilde{C}_{m,:} = P_{\text{hard}}\, C_{m,:}, \qquad \tilde{C}_{m,k} \in \mathbb{R}^3. \tag{6}$$

We composite all the layers in the ordered ranks using standard transmittance accumulation. For each ray m,

$$\begin{aligned} T_{m,1} &= 1, \quad T_{m,k+1} = T_{m,k}\left(1-\tilde{\alpha}_{m,k}\right), \\ \hat{\mathbf{C}}_m &= \sum_{k=1}^{N} T_{m,k}\,\tilde{\alpha}_{m,k}\,\tilde{C}_{m,k} \end{aligned} \tag{7}$$

The straight-through NeuralSort in (4) keeps the forward pass strictly discrete while preserving gradients through the soft relaxation, enabling stable, end-to-end learning of view-dependent occlusions and per-primitive colors. As $\tau \to 0$, P_τ approaches a hard permutation and the learned order becomes crisp.



(a) Composition (b) Layer3D (c) Ordered Layers

Figure 4: *View-dependent layer composition. To reproduce the correct composition (a), neural primitives (b) must be rendered in a front-to-back order determined by the viewing direction (c).*

3.3. Optimization

To optimize the representation, we incorporate the Layer3D model into a text-to-3D pipeline, as illustrated in Figure 3. For each text prompt, we train the layered representation starting with a specific initialization. Building on DreamFusion [PJBM22], each iteration of the optimization process involves the following steps: (1) randomly sampling a camera, (2) rendering an image of the Layer3D model from the sampled camera position using an albedo setting, (3) computing the gradients of the loss function with respect to the Layer3D parameters, and (4) updating the parameters using an optimizer.

Primitive initialization. To facilitate the early stages of optimization, we initialize each primitive by placing a density "blob" centered around the origin. This initialization encourages the scene content to be concentrated around the center of the 3D coordinate space, rather than directly near the sampled camera positions. To parameterize the added density $\tau^i_{\text{init}}(\mu)$ for the primitive $\mathbf{P}_i$, we employ a Gaussian Probability Density Function (PDF) as follows:

$$\tau^i_{\text{init}}(\mu) = \hat{\lambda}_i \cdot \exp\left(-\frac{\|\mu\|^2}{2\hat{\sigma}_i^2}\right) \tag{8}$$

where $\hat{\lambda}_i, \hat{\sigma}_i$ are hyperparameters for scale and width of the PDF depending on primitive $\mathbf{P}_i$.

Camera configurations. At each iteration, a camera position is randomly sampled in spherical coordinates, with elevation angle $\phi_{\text{cam}} \in [-10^\circ, 90^\circ]$, azimuth angle $\theta_{\text{cam}} \in [-180^\circ, 180^\circ]$, and distance from the origin in $[1, 1.2]$. Given the camera pose, we render each 3D neural primitive and compute the final composition image at a resolution of 64×64.

Loss. We optimize the parameters of each neural implicit primitive using SDS, ensuring the rendered composition aligns with a high probability according to a text-conditioned diffusion model. The loss penalizes the KL-divergence between a unimodal Gaussian distribution centered at a learned sample $g(\psi, c)$, and the data distribution $p_\phi(z; y, t)$ captured by the pre-trained diffusion model conditioned on text embeddings y, averaged over several sampled time steps t:

$$\mathcal{L}_{\text{sds}}(\psi, \mathbf{c}) = \mathbb{E}_t\left[\frac{\sigma_t}{\alpha_t} w(t) \mathrm{KL}\left(q(\mathbf{z}_t \mid g(\psi, \mathbf{c}); y, t) \,\|\, p_\phi(\mathbf{z}_t; y, t)\right)\right] \tag{9}$$

where t is a timestep, $w(t)$ is a weighting function, and α_t, σ_t are coefficients of the diffusion model.

We also introduce an entropy loss to penalize uncertainty in the model's mixing coefficients. Following NeRF's volume rendering rule that maps density to opacity [MST*20], we derive a per-layer mixing coefficient k_l from the sampled volume density of layer l. We prefer the mixing coefficients to be biased towards 0 or 1 to avoid translucent shapes. This preference is enforced by the entropy loss applied to the per-layer mixing coefficients k_l, which involves only the MLP parameters ψ:

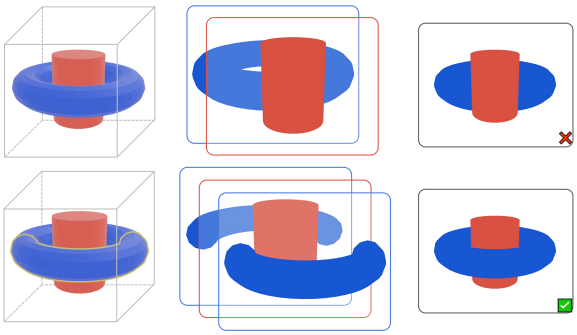
$$\mathcal{L}_{\text{entr}}(\psi) = -\sum_l k_l \log(k_l) - \sum_l (1-k_l)\log(1-k_l) \tag{10}$$

We further introduce a shape regularization loss to encourage each primitive to intersect any viewing ray in one contiguous depth interval, so that it can be rendered as a single stacking layer (Figure 5). For each ray $m \in \{1, \ldots, M\}$, we sample S depth values $\{t_s\}_{s=1}^S$ along the ray. For a given primitive $\mathbf{P}_i$, the per-sample density is given by

$$h_{i,m}(t_s) = T_{i,m,s}\,\alpha_{i,m,s}. \tag{11}$$

where $\alpha_{i,m,s}$ is the per-sample opacity and $T_{i,m,s}$ is the accumulated transmittance. Ideally, $h_i(t)$ should be unimodal along the ray, corresponding to a single contiguous intersection interval. If $h_i(t)$ exhibits multiple peaks, the primitive becomes self-occluded, which conflicts with the assumption of front-to-back rendering.

To regularize this, we penalize local minima in $h_i(t)$ that are



(a) 3D Primitives (b) Ordered Layers (c) Composition

Figure 5: *Handling self-occlusion. To ensure correct front-to-back rendering and composition (c), the 3D primitives (a) are decomposed into non-self-occluded layers (b, lower row) that can be sequentially rendered in linear order. In contrast, primitives with self-occlusion (upper row) cannot be correctly rendered using a front-to-back strategy.*

flanked by higher neighboring values:

$$\mathcal{L}_{\text{shape}}(\psi) = \sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{s=2}^{S-1} \text{ReLU}\big(h_{i,m}(t_{s-1}) - h_{i,m}(t_s)\big) \cdot \text{ReLU}\big(h_{i,m}(t_{s+1}) - h_{i,m}(t_s)\big)\big) \tag{12}$$

where $\text{ReLU}(\cdot) = \max(\cdot, 0)$. This loss becomes positive when t_m is a local minimum surrounded by higher opacity, thus discouraging multi-interval activations of the same primitive along a ray.

The final optimization objective combines the SDS, entropy, and shape regularization losses as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{sds}}(\psi, \mathbf{c}) + \lambda_{\text{e}}\,\mathcal{L}_{\text{entr}}(\psi) + \lambda_{\text{s}}\,\mathcal{L}_{\text{shape}}(\psi) \tag{13}$$

where λ_{e} and λ_{s} are weighting parameters.

4. Implementation Details

We employ DeepFloyd [Sta23] as the 2D diffusion prior in our framework, which accelerates convergence by bypassing back-propagation through the image encoder and performing denoising diffusion directly in pixel space. To guide the model toward a flat vector style, we append the phrase "2D vector art" to each prompt. For multiview supervision, we adopt a weighted combination of text embeddings, appending descriptors such as "front view," "side view," and "back view" according to the azimuth angle.

All experiments are conducted on a single 32GB V100 GPU. We set $N = 5$, $\lambda_{\text{e}} = 10^{-2}$ and $\lambda_{\text{s}} = 10^{-4}$ as default parameters. Optimization was performed for 100 iterations per representation, requiring an average of 65 minutes per prompt. To generate the final SVG outputs, we fit cubic Bézier curves using Inkscape [Ink] for each layer l_i rendered from specific viewpoints. The curves extracted from all layers $\{l_1, \ldots, l_N\}$ are then combined in rendering order to produce the final vectorized result.

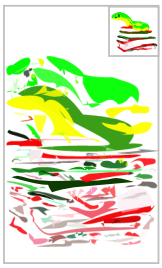
5. Experiments

We conduct ablation studies under different configurations, as presented in Section 5.2, and report the results together with comparisons against baseline methods in Section 5.1.

5.1. Results and Comparisons

We present multiview results in Figures 1 and 12, and illustrate the corresponding layer decomposition outcomes in Figure 11.

Baselines. For baseline comparisons, we first optimize the original NeRF under the same supervision to obtain multiview images, which are then vectorized using Inkscape [Ink] and LIVE [MZX*22]. In addition, we vectorize the composite images generated by Layer3D as an alternative baseline to demonstrate that incorporating a layered representation facilitates more reliable curve extraction. For a fair comparison, we set the maximum number of paths in LIVE to five, matching the upper bound of our primitive count. We do not employ higher path counts in LIVE, as illustrated on the right, since this results in an excessive number of fragmented and irregular primitives that hinder meaningful decomposition.



5.1.1. Qualitative Comparisons

Figure 11 presents a qualitative comparison of the results generated by our method and baseline approaches. Our method is parameter-efficient, producing results that are faithful to the prompt and effectively capture the input concepts. While vectorizing with Inkscape can preserve the original raster image's appearance, it often produces highly fragmented layers, making further editing difficult. NeRF-based approaches are primarily designed to reconstruct photorealistic appearance with fine-grained color and texture variation; this emphasis on high-frequency detail makes subsequent vectorization challenging and often leads to fragmented layers. Moreover, vectorizing each view independently does not enforce cross-view layer correspondence. In contrast, LIVE struggles to accurately represent the geometry of objects when using a low parameter count, a limitation that becomes evident when comparing the vectorized composite image from Layer3D with LIVE. Additionally, due to our 3D primitive separation, the multiview vector graphics extracted from Layer3D maintain shape correspondence across different views, a property not guaranteed by separate vectorization of images from different views. Thus, our 3D layered representation provides distinct advantages in multiview vector graphics generation, enabling the separation of clean shapes with diverse topologies and view correspondence, all while maintaining alignment with the input concept.

Layer-wise quality. Given the difficulty of quantitatively evaluating vector quality, we conduct a user study to assess the layer-wise quality based on subjective judgment. The study consists of two questions: (1) Please select the decomposed layers that exhibit the best quality, and (2) Please select the multiview results that demonstrate consistent layer-wise correspondence.

We selected eight text prompts and three viewing directions to

generate multiview vector graphics using two baseline methods and our proposed approach, producing a total of $8 \times 3 \times 3 = 72$ vector graphics for user evaluation. A set of 9 vector graphics with their corresponding layer decomposition was presented for each text prompt followed by two questions. Users were asked to directly vote for their preferred result among the three methods randomly ordered for each question. 20 participants were involved in the assessment and the voting results are presented in Figure 6.

We observe that the results generated by our method were preferred by participants for both layer-wise quality and multiview correspondence.

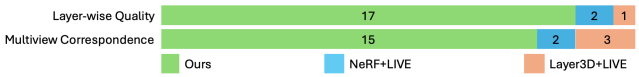


Figure 6: *User study results: our method receives the most votes on both criteria, outperforming the two baselines.*

5.1.2. Quantitative Comparisons

Beyond qualitative evaluations, we assess our results from the following perspectives, aligned with our goal of generating multiview vector graphics with high-quality intermediate layers and consistent layer-wise correspondence:

Visual quality. To assess the visual quality of the generated vector graphics, we compute the Fréchet Inception Distance (FID) between the rendered images of the generated set and those of the ground truth SVGs. For this purpose, we have compiled a dataset consisting of 50 high-quality vector graphics sourced from Openclipart [Ope], covering a diverse range of categories such as characters and animals, which serves as our ground truth.

Text alignment. The multiview vector graphics should exhibit alignment with the input text prompt. To evaluate this, we configure a turntable setting with four camera positions and export multiview vector graphics for each representation. The alignment between the generated SVGs and the input text prompt is quantified by calculating the CLIP cosine similarity [RKH*21] between the text prompt and the rendered SVGs.

Layer-wise text alignment. The multiview results for each specific layer should maintain semantic consistency. To evaluate this, we compute the CLIP cosine similarity between each layer image from a specific viewpoint and the input text prompt. The final score for each representation is then obtained by averaging the CLIP similarities across all layers and viewpoints.

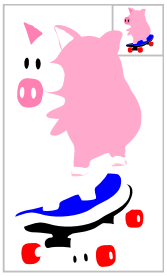
Table 1: *Quantitative comparison against baselines using FID, CLIP, and layer-wise CLIP.*

	#Layers	FID ↓	CLIP ↑	LayerCLIP ↑
NeRF	-	368.64	24.16	-
Layer3D	-	363.62	32.35	-
NeRF+LIVE	5	323.48	27.70	24.53
Layer3D+LIVE	5	324.21	29.10	24.53
Ours	≤ 5	**315.07**	**32.40**	**29.27**

The quantitative comparison results are presented in Table 1. We used 50 prompts and generated 2 samples per prompt. We first calculate the FID and CLIP cosine similarity for raster results generated by both NeRF and Layer3D. Using a rasterized high-quality vector graphics dataset as the ground truth for FID calculations, Layer3D achieves higher visual quality and text alignment scores compared to NeRF at the raster level.

For vector-level comparisons, we also compute the FID and CLIP scores by rasterizing the composite vector graphics. Compared to direct vectorization from composite images, our method — which performs primitive-wise decomposition during optimization — obtains the highest CLIP score while maintaining comparable visual quality. This highlights our ability to semantically replicate the desired content and vector style, consistent with the qualitative results shown in Figure 11. Furthermore, we assess the layer-wise quality by calculating the CLIP score between the given text prompt and each generated layer for both the baselines and our method. Our approach achieves the highest layer-wise CLIP score among the three methods.

Compare with Adobe Turntable. Since Adobe Turntable does not support text prompts, we use the second-view pig example from Figures 1 and 11 as input and rotate it to generate a result aligned with the first view. As shown on the right, Adobe Turntable fails to preserve front-to-back layer composition under viewpoint changes: after rotation, the layers no longer overlap correctly. In contrast, Layer3D maintains consistent front-to-back rendering by enforcing a view-dependent layer ordering, as illustrated in Figure 11.



5.2. Ablation Study

To assess the contribution of each individual loss term, we conducted experiments with different combinations of losses. For a clearer assessment of layer performance under varying supervisory signals, we initialized the predicted colors from $\mathcal{F}_c$ by optimizing them with respect to a generated image conditioned on the target text prompt, and subsequently froze these colors for comparison across different loss configurations.

(a) $\mathcal{L}_{SDS}$ (b) $+\lambda_e \mathcal{L}_{\text{entr}}$ (c) $+\lambda_s \mathcal{L}_{\text{shape}}$ (d) Ours

Figure 7: *Results obtained with different combinations of loss terms. Note: (b) $\mathcal{L}_{SDS} + \lambda_e \mathcal{L}_{entr}$, (c) $\mathcal{L}_{SDS} + \lambda_s \mathcal{L}_{shape}$, (d) Ours: $\mathcal{L}_{SDS} + \lambda_e \mathcal{L}_{entr} + \lambda_s \mathcal{L}_{shape}$. The results are obtained given the text prompt "A match stick on fire, 2D vector art."*

As illustrated in Figure 7, employing $\mathcal{L}_{\text{SDS}}$ alone produces the target subject but with noticeably blurred and indistinct boundaries,

which is not suitable for further vectorization. We observe that the inclusion of $\mathcal{L}_{\text{entr}}$ encourages the formation of sharper and more coherent boundaries, while $\mathcal{L}_{\text{shape}}$ promotes cohesive structures. When combined, our proposed loss formulation leverages the complementary strengths of these components and achieves the highest quality, yielding outputs that are both structurally accurate and visually coherent.

We further investigate the effect of the number of layers. Since N acts as an upper bound, we incrementally increase N for the examples until the decomposition no longer yields additional layers. As shown in Figure 8, using too few layers can oversimplify color variations, leading to overly abstract, silhouette-like results.

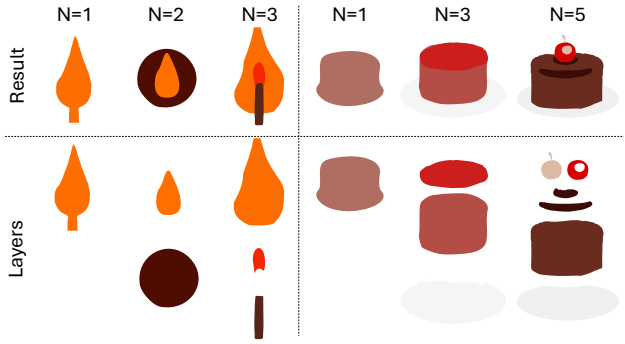


Figure 8: *Results obtained with different numbers of layers N. For the two examples, the prompts are "A match stick on fire, 2D vector art" and "A cake with chocolate frosting and cherry, 2D vector art," respectively.*

We also experimented with pre-assigned colors by assigning fixed colors to each primitive and observing the resulting shape optimization process. In this setting, we find that primitives tend to duplicate shapes with different opacities that blend to approximate the target appearance at the raster level. Such behavior prioritizes color decomposition over the intended shape decomposition, deviating from our primary objective. Moreover, it often leads to less detailed generation, as shown in the right column of Figure 9 in comparison to the python example in Figure 1.

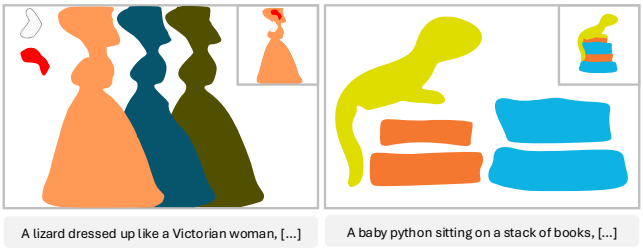


Figure 9: *Results generated with pre-assigned colors. The left column shows an example where duplicate shapes were generated, while the right column shows an example where the generated output contains reduced detail.*

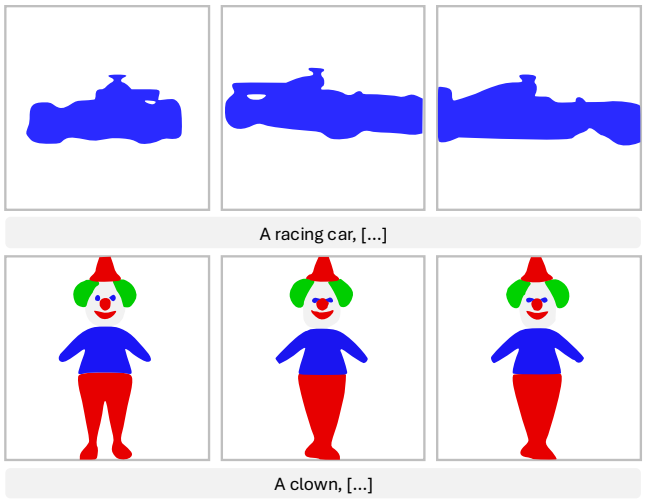


Figure 10: *Failure cases. The upper row illustrates a failure case where only a single layer is generated for the racing car. The lower row depicts another failure case in which the diffusion model fails to provide accurate geometric supervision across different perspectives, resulting in the canonical pose facing the camera despite camera rotation.*

6. Limitations

Our current framework adopts a vanilla text-to-3D pipeline without integrating multiview diffusion, which often leads to failure cases where the recovered geometry becomes inconsistent across different viewpoints. The method is also subject to typical artifacts of SDS-based optimization; a promising direction for future work is to integrate our approach with recent 3D diffusion models. Moreover, the method lacks explicit layer-wise control, particularly in scenarios where primitives share similar colors, which can cause multiple layers to collapse into a single representation, as illustrated in Fig. 10. Future work may extend the approach by developing mechanisms for disentangled layer control.

7. Conclusion

This paper has presented Layer3D, a 3D layered neural implicit representation aimed at addressing the challenge of decomposing 3D subjects into primitive-based vector layers. It enables the rendering of multiview layer images, facilitating curve extraction while accommodating shapes with diverse topologies. Integrated into a text-to-3D pipeline based on SDS, Layer3D demonstrates the ability to generate meaningful intermediate layers, which are particularly beneficial for vector graphics editing. We validated through experiments the effectiveness of Layer3D in generating high-quality, editable vector outputs and maintaining consistent vector graphics across views.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Project No. 62502410).

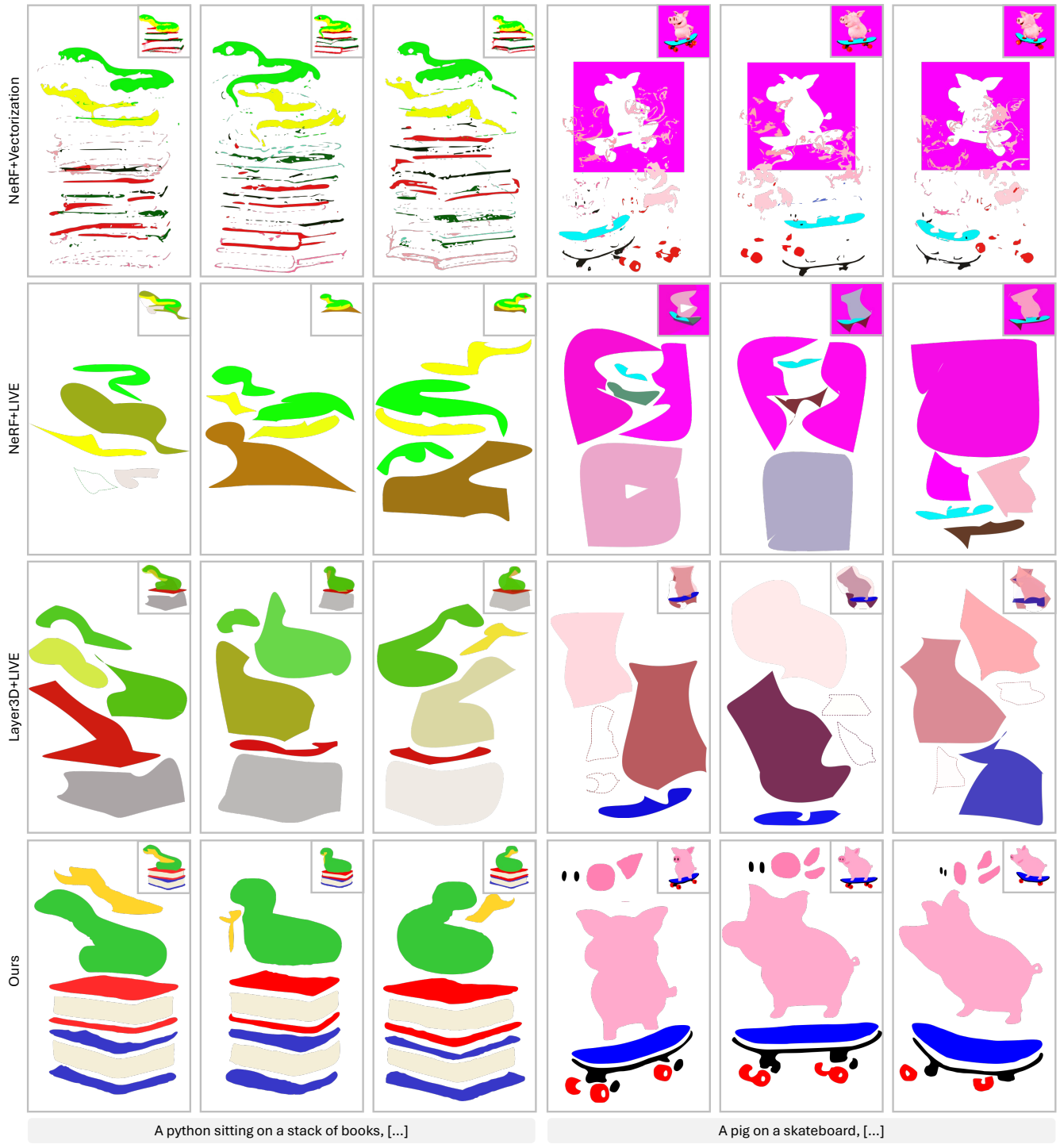


Figure 11: *Layer-wise decomposition results comparing baseline methods with our approach. By leveraging 3D neural primitives, our method establishes consistent layer-wise 3D correspondences across varying viewing angles, whereas the baselines either produce undesirable fragmented results or yield layers that are inconsistent across views.*

Figure 12: *More multiview results generated by Layer3D.*

References

[Ado24] ADOBE: Project Turntable. https://www.adobe.com/max/2024/sessions/project-turntable-gs3-9.html, 2024. [Accessed 16-09-2025]. 2

[ASP07] ASENTE P., SCHUSTER M., PETTIT T.: Dynamic Planar Map Illustration. *ACM Trans. Graph. 26*, 3 (July 2007), 30–es. URL: https://doi.org/10.1145/1276377.1276415, doi:10.1145/1276377.1276415. 2, 3

[BG89] BAUDELAIRE P., GANGNET M.: Planar Maps: An Interaction Paradigm for Graphic Design. *SIGCHI Bull. 20*, SI (Mar. 1989), 313–318. URL: https://doi.org/10.1145/67450.67511, doi:10.1145/67450.67511. 2, 3

[CLPK24] CHOI C., LEE J., PARK J., KIM Y. M.: 3Doodle: Compact Abstraction of Objects with 3D Strokes. *ACM Trans. Graph. 43*, 4 (July 2024). URL: https://doi.org/10.1145/3658156, doi:10.1145/3658156. 3

[Cor25] CORTEX V.: Vtracer. https://github.com/visioncortex/vtracer, 2025. Accessed: 2025-01-12. 2

[CSL*24] CHEN M., SHAPOVALOV R., LAINA I., MONNIER T., WANG J., NOVOTNY D., VEDALDI A.: PartGen: Part-level 3D Generation and Reconstruction with Multi-View Diffusion Models. *arXiv preprint arXiv:2412.18608* (2024). 3

[CWS*25] CHEN M., WANG J., SHAPOVALOV R., MONNIER T., JUNG H., WANG D., RANJAN R., LAINA I., VEDALDI A.: AutoPartGen: Autogressive 3D Part Generation and Discovery. *arXiv preprint arXiv:2507.13346* (2025). 3

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive Contours for Conveying Shape. *ACM Trans. Graph. 22*, 3 (July 2003), 848–855. URL: https://doi.org/10.1145/882262.882354, doi:10.1145/882262.882354. 3

[DKT*23] DU Z.-J., KANG L.-F., TAN J., GINGOLD Y., XU K.: Image Vectorization and Editing via Linear Gradient Layer Decomposition. *ACM Trans. Graph. 42*, 4 (July 2023). URL: https://doi.org/10.1145/3592128, doi:10.1145/3592128. 2

[DRvdP14] DALSTEIN B., RONFARD R., VAN DE PANNE M.: Vector Graphics Complexes. *ACM Trans. Graph. 33*, 4 (July 2014). URL: https://doi.org/10.1145/2601097.2601169, doi:10.1145/2601097.2601169. 2

[DRvdP15] DALSTEIN B., RONFARD R., VAN DE PANNE M.: Vector Graphics Animation with Time-varying Topology. *ACM Trans. Graph. 34*, 4 (July 2015). URL: https://doi.org/10.1145/2766913, doi:10.1145/2766913. 2

[EPD09] EISEMANN E., PARIS S., DURAND F.: A Visibility Algorithm for Converting 3D Meshes into Editable 2D Vector Graphics. *ACM Trans. Graph. 28*, 3 (July 2009). URL: https://doi.org/10.1145/1531326.1531389, doi:10.1145/1531326.1531389. 2, 3

[EWHS08] EISEMANN E., WINNEMÖLLER H., HART J. C., SALESIN D.: Stylized Vector Art from 3D Models with Region Support. In *Proceedings of the Nineteenth Eurographics Conference on Rendering* (Goslar, DEU, 2008), EGSR '08, Eurographics Association, p. 1199–1207. URL: https://doi.org/10.1111/j.1467-8659.2008.01258.x, doi:10.1111/j.1467-8659.2008.01258.x. 2, 3

[FLB17] FAVREAU J.-D., LAFARGE F., BOUSSEAU A.: Photo2clipart: Image Abstraction and Vectorization Using Layered Linear Gradients. *ACM Trans. Graph. 36*, 6 (Nov. 2017). URL: https://doi.org/10.1145/3130800.3130888, doi:10.1145/3130800.3130888. 2

[FSW22] FRANS K., SOROS L., WITKOWSKI O.: CLIPDraw: Exploring Text-to-Drawing Synthesis through Language-Image Encoders. In *Advances in Neural Information Processing Systems* (2022), Koyejo S., Mohamed S., Agarwal A., Belgrave D., Cho K., Oh A., (Eds.), vol. 35, Curran Associates, Inc., pp. 5207–5218. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/21f76686538a5f06dc431efea5f475f5-Paper-Conference.pdf. 2

[GWZE19] GROVER A., WANG E., ZWEIG A., ERMON S.: Stochastic Optimization of Sorting Networks via Continuous Relaxations. In *International Conference on Learning Representations* (2019). URL: https://openreview.net/forum?id=H1eSS3CcKX. 2, 4

[Ink] INKSCAPE: Inkscape. https://inkscape.org/. Accessed: 2026-02-10. 6

[JDA07] JUDD T., DURAND F., ADELSON E.: Apparent Ridges for Line Drawing. *ACM Trans. Graph. 26*, 3 (jul 2007), 19–es. URL: https://doi.org/10.1145/1276377.1276401, doi:10.1145/1276377.1276401. 3

[JMB*22] JAIN A., MILDENHALL B., BARRON J. T., ABBEEL P., POOLE B.: Zero-Shot Text-Guided Object Generation with Dream Fields. 3

[JXA22] JAIN A., XIE A., ABBEEL P.: VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models. *arXiv* (2022). 2, 3

[KH11] KARSCH K., HART J. C.: Snaxels on a Plane. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2011), NPAR '11, Association for Computing Machinery, p. 35–42. URL: https://doi.org/10.1145/2024676.2024683, doi:10.1145/2024676.2024683. 3

[KKLD23] KERBL B., KOPANAS G., LEIMKUEHLER T., DRETTAKIS G.: 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph. 42*, 4 (July 2023). URL: https://doi.org/10.1145/3592433, doi:10.1145/3592433. 3

[LBHH23] LIU C., BÉNARD P., HERTZMANN A., HOSHYARI S.: Con-Tesse: Accurate Occluding Contours for Subdivision Surfaces. *ACM Trans. Graph. 42*, 1 (Jan. 2023). URL: https://doi.org/10.1145/3544778, doi:10.1145/3544778. 3

[LFT*25] LIU R., FU D., TAN N., LANG I., HANOCKA R.: WIR3D: Visually-Informed and Geometry-Aware 3D Shape Abstraction, 2025. URL: https://arxiv.org/abs/2505.04813, arXiv:2505.04813. 3

[LLGRK20] 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). URL: https://doi.org/10.1145/3414685.3417871, doi:10.1145/3414685.3417871. 2

[LLL*24] LIU A., LIN C., LIU Y., LONG X., DOU Z., GUO H.-X., LUO P., WANG W.: Part123: Part-aware 3D Reconstruction from a Single-view Image. In *ACM SIGGRAPH 2024 Conference Papers* (New York, NY, USA, 2024), SIGGRAPH '24, Association for Computing Machinery. URL: https://doi.org/10.1145/3641519.3657482, doi:10.1145/3641519.3657482. 3

[LLP*25] LIN Y., LIN C., PAN P., YAN H., FENG Y., MU Y., FRAGKIADAKI K.: PartCrafter: Structured 3D Mesh Generation via Compositional Latent Diffusion Transformers, 2025. URL: https://arxiv.org/abs/2506.05573, arXiv:2506.05573. 3

[LXL*25] LI Y., XIAO J., LU Z., WANG Y., JIANG H.: Empowering Vector Graphics with Consistently Arbitrary Viewing and View-dependent Visibility, 2025. URL: https://arxiv.org/abs/2505.21377, arXiv:2505.21377. 2, 3

[MGY*19] MO K., GUERRERO P., YI L., SU H., WONKA P., MITRA N. J., GUIBAS L. J.: StructureNet: Hierarchical Graph Networks for 3D Shape Generation. *ACM Trans. Graph. 38*, 6 (Nov. 2019). URL: https://doi.org/10.1145/3355089.3356527, doi:10.1145/3355089.3356527. 3

[MLY*25] MA C., LI Y., YAN X., XU J., YANG Y., WANG C., ZHAO Z., GUO Y., CHEN Z., GUO C.: P3-SAM: Native 3D Part Segmentation,

2025. URL: https://arxiv.org/abs/2509.06784, arXiv:2509.06784. 3

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV* (2020). 2, 3, 4, 5

[MZX*22] 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 conference on computer vision and pattern recognition* (2022). 2, 6

[OBB*13] ORZAN A., BOUSSEAU A., BARLA P., WINNEMÖLLER H., THOLLOT J., SALESIN D.: Diffusion Curves: A Vector Representation for Smooth-shaded Images. *Commun. ACM 56*, 7 (July 2013), 101–108. URL: https://doi.org/10.1145/2483852.2483873, doi:10.1145/2483852.2483873. 2

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-Valley Lines on Meshes via Implicit Surface Fitting. *ACM Trans. Graph. 23*, 3 (aug 2004), 609–612. URL: https://doi.org/10.1145/1015706.1015768, doi:10.1145/1015706.1015768. 3

[Ope] OPENCLIPART: Openclipart. https://openclipart.org/. Accessed: 2026-02-10. 7

[PAR*25] POLACZEK S., ALALUF Y., RICHARDSON E., VINKER Y., COHEN-OR D.: NeuralSVG: An Implicit Representation for Text-to-Vector Generation, 2025. URL: https://arxiv.org/abs/2501.03992, arXiv:2501.03992. 2, 3

[PJBM22] POOLE B., JAIN A., BARRON J. T., MILDENHALL B.: DreamFusion: Text-to-3D using 2D Diffusion. *arXiv* (2022). 2, 3, 5

[PKGF21] PASCHALIDOU D., KATHAROPOULOS A., GEIGER A., FIDLER S.: Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks . In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Los Alamitos, CA, USA, June 2021), IEEE Computer Society, pp. 3203–3214. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00322, doi:10.1109/CVPR46437.2021.00322. 3

[RBL*21] ROMBACH R., BLATTMANN A., LORENZ D., ESSER P., OMMER B.: High-Resolution Image Synthesis with Latent Diffusion Models, 2021. arXiv:2112.10752. 2, 3

[RGLM21] REDDY P., GHARBI M., LUKAC M., MITRA N. J.: Im2Vec: Synthesizing Vector Graphics without Vector Supervision, 2021. arXiv:2102.02798. 2

[RID10] RIVERS A., IGARASHI T., DURAND F.: 2.5D Cartoon Models. *ACM Trans. Graph. 29*, 4 (July 2010). URL: https://doi.org/10.1145/1778765.1778796, doi:10.1145/1778765.1778796. 2, 3

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., KRUEGER G., SUTSKEVER I.: Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning* (18–24 Jul 2021), Meila M., Zhang T., (Eds.), vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 8748–8763. URL: https://proceedings.mlr.press/v139/radford21a.html. 2, 7

[RPA*24] RODRIGUEZ J. A., PURI A., AGARWAL S., LARADJI I. H., RODRIGUEZ P., RAJESWAR S., VAZQUEZ D., PAL C., PEDERSOLI M.: StarVector: Generating Scalable Vector Graphics Code from Images and Text, 2024. URL: https://arxiv.org/abs/2312.11556, arXiv:2312.11556. 2

[SEH08] STROILA M., EISEMANN E., HART J. C.: Clip Art Rendering of Smooth Isosurfaces. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 135–145. doi:10.1109/TVCG.2007.1058. 3

[Sel25] SELINGER P.: Potrace. https://github.com/tatarize/potrace, 2025. Accessed: 2025-01-12. 2

[Sta23] STABILITYAI: DeepFloyd IF. https://github.com/deep-floyd/IF, 2023. Accessed: 2025-01-17. 3, 6

[SWY*23] SHI Y., WANG P., YE J., MAI L., LI K., YANG X.: MVDream: Multi-view Diffusion for 3D Generation. *arXiv:2308.16512* (2023). 3

[TLF*24] THAMIZHARASAN V., LIU D., FISHER M., ZHAO N., KALOGERAKIS E., LUKÁC M.: NIVeL: Neural Implicit Vector Layers for Text-to-Vector Generation. In *CVPR* (2024), pp. 4589–4597. URL: https://doi.org/10.1109/CVPR52733.2024.00439. 2, 3

[Web25] WEBER M.: Autotrace. https://github.com/autotrace/autotrace, 2025. Accessed: 2025-01-12. 2

[Wor18] WORLD WIDE WEB CONSORTIUM (W3C): Scalable Vector Graphics (SVG) 2, 2018. Accessed: 2023-07-09. URL: https://www.w3.org/TR/SVG2/. 2

[WZLY25] WANG C., ZHOU H., LUO L., YU Q.: ViewCraft3D: High-Fidelity and View-Consistent 3D Vector Graphics Synthesis, 2025. URL: https://arxiv.org/abs/2505.19492, arXiv:2505.19492. 3

[XZW*24] 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)* (June 2024), pp. 4546–4555. 2, 3

[ZTNW23] ZHANG B., TANG J., NIESSNER M., WONKA P.: 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. *ACM Trans. Graph. 42*, 4 (July 2023). URL: https://doi.org/10.1145/3592442, doi:10.1145/3592442. 3

[ZWZ*24] ZHANG Y., WANG L., ZOU C., WU T., MA R.: Diff3DS: Generating View-Consistent 3D Sketch via Differentiable Curve Rendering. *arXiv preprint arXiv:2405.15305* (2024). 3

[ZZL23] ZHANG P., ZHAO N., LIAO J.: Text-Guided Vector Graphics Customization. In *SIGGRAPH Asia 2023 Conference Papers* (New York, NY, USA, 2023), SA '23, Association for Computing Machinery. URL: https://doi.org/10.1145/3610548.3618232, doi:10.1145/3610548.3618232. 3

[ZZL24] ZHANG P., ZHAO N., LIAO J.: Text-to-Vector Generation with Neural Path Representation. *ACM Trans. Graph. 43*, 4 (July 2024). URL: https://doi.org/10.1145/3658204, doi:10.1145/3658204. 2, 3