Illustrator's Depth: Monocular Layer Index Prediction for Image Decomposition

Nissim Maruani*
Inria, UCA

Peiying Zhang
CityUHK

Siddhartha Chaudhuri
Adobe Research

Matthew Fisher
Adobe Research

Nanxuan Zhao
Adobe Research

Vladimir G. Kim
Adobe Research

Pierre Alliez
Inria, UCA

Mathieu Desbrun
Inria/X, IP Paris

Wang Yifan
Adobe Research

Abstract

We introduce Illustrator's Depth, a novel definition of depth that addresses a key challenge in digital content creation: decomposing flat images into editable, ordered layers. Inspired by an artist's compositional process, illustrator's depth infers a layer index for each pixel, forming an interpretable image decomposition through a discrete, globally consistent ordering of elements optimized for editability. We also propose and train a neural network using a curated dataset of layered vector graphics to predict layering directly from raster inputs. Our layer index inference unlocks a range of powerful downstream applications. In particular, it significantly outperforms state-of-the-art baselines for image vectorization while also enabling high-fidelity text-to-vector-graphics generation, automatic 3D relief generation from 2D images, and intuitive depth-aware editing. By reframing depth from a physical quantity to a creative abstraction, illustrator's depth prediction offers a new foundation for editable image decomposition.

1. Introduction

The organization of a digital artwork into a stack of layers is a fundamental concept in creative software. This paradigm, common to both vector-based and raster graphics tools, is central to the creative process as it allows for the independent manipulation and editing of individual compositional elements. This layering is also inherently related to the physical depth of objects within a scene, in that closer elements obscure those that are farther away.

While recent neural architectures can efficiently and accurately predict monocular depth from images [2, 54] or compute panoptic segmentations [15, 33], they are unable to decompose input illustrations or images into useful, ordered layers for three main reasons. First, illustrative layers differ fundamentally from physical depths: important
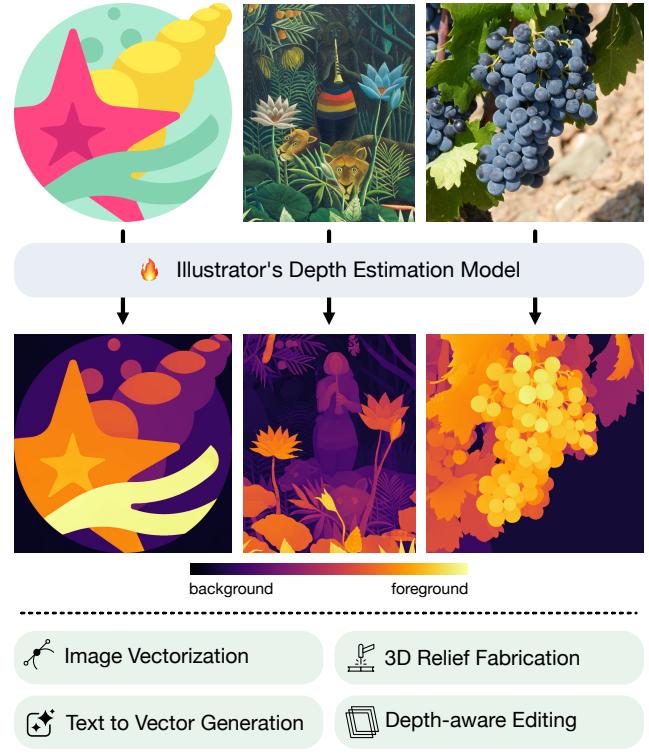

Figure 1. **Overview.** Given an input image, our model predicts *Illustrator's Depth*, a learned ordering of compositional layers that reflects how an artist might have structured the image layout. This representation, applicable broadly to illustrations (left), paintings (middle), or even some realistic images (right), enables multiple downstream applications such as vectorization, intuitive editing, text-to-vector generation, and 3D relief fabrication.

visual elements such as shadows may be placed *above* the objects on which they are cast, and non-orthogonal flat surfaces with overlapping physical *depth gradients* may nevertheless be mapped to discrete, sortable layers (see dominoes in Fig. 2). Second, because illustrations typically appear on flat media (i.e., book pages, posters, or paintings), monocu-

---

*Work performed during an internship at Adobe Research

1

lar depth estimation models are explicitly trained to *ignore* them (see t-shirt in Fig. 2). Third, illustrative layering also differs from plain panoptic segmentation: the grouping and structuring of segmented regions of an input are key to the editability of the layer decomposition. An illustrator's notion of layer depth is thus a subtle mix between segmentation and depth ordering to facilitate both design and editing.

Although rarely acknowledged or articulated as such, layer inference is a core challenge in vectorization that impacts numerous downstream applications by offering an intuitive layer decomposition enhancing editing capabilities. Existing state-of-the-art methods, however, remain limited in scope, either handling only simple inputs [44, 50], or relying on brittle heuristics [11, 18, 23, 29, 61, 62] which do not consistently yield useful results. In the raster domain, several approaches have explored transparent layer extraction or generation [20, 28, 49, 53, 56], yet these operate exclusively at the object level. To the best of our knowledge, no existing technique can achieve fine-grained, detailed image layer decomposition.

We introduce Illustrator's Depth, a new concept designed to address these challenges by providing a novel way to represent the structural layering of vector graphics. Specifically, we define the illustrator's depth of an image as the inverse mapping from each pixel to its corresponding layer index in its digital mockup, effectively capturing the spatial and compositional ordering of the artwork. We infer illustrators' depth from arbitrary images automatically by leveraging a Depth Pro based neural network [2] trained on a large, curated SVG dataset. Our model operates in a feed-forward manner to predict pixel-level layer indices, enabling a wide range of applications such as image editing and depth-aware vector graphics manipulation.

More specifically, we present a number of contributions:
- We introduce the notion of *Illustrator's Depth* and train a network to predict it, enabling fast layer decomposition;
- We show that incorporating our model into standard vectorization pipelines yields consistently layered SVGs with state-of-the-art visual fidelity;
- We propose a novel method for evaluating layer quality in vector graphics by rasterizing the predicted illustrator's depth and assessing its consistency with the ground truth;
- We demonstrate that coupling our pipeline with Text2Img models substantially enhances the generation of high-quality, editable vector illustrations from text;
- Finally, we showcase other applications of illustrator's depth in layer-based segmentation, depth-aware object insertion, tactile graphics creation, and artwork analysis.

2. Related Work

Monocular depth estimation (MDE). Classical learning approaches for depth estimation from images [6, 8, 12, 36, 41, 60] have evolved into strong backbones trained on diverse data [1, 2, 31, 54]. They yield finely detailed and *continuous* (relative or metric) depth maps that serve as robust physical priors. Yet, they remain blind to content without true volume, like printed posters or patterns on clothing. In contrast, our objective is to produce a new kind of depth *prioritizing user editability over metric prediction.*

Layered depth for view synthesis. Layered Depth Images store multiple depth samples per ray to model occlusions [5, 43], while Multiplane Images approximate scenes by many fronto-parallel planes for novel-view rendering [24, 63]. These abstractions excel at detecting disocclusions and synthesizing new views, but they produce *multi-sample* or *multi-plane depths*, not a single discrete index per pixel that a designer can restack. Furthermore, they focus on physical depth like MDE, unlike our illustrator's depth which focuses on layer index prediction.

Amodal / instance / panoptic segmentation. Moving from geometry to semantics, segmentation families group regions by categories, but do not encode geometric ordering. Standard instance and panoptic methods provide high-quality visible masks [3, 4, 10, 14–16, 32] without global per-pixel depth ordering. Amodal instance and amodal-panoptic formulations extend masks to occluded regions (for countable "thing" categories, typically), while "stuff" categories remain modal; representative datasets and mod-

(a) Input Image   (b) Illustrator's Depth (Ours)   (c) Physical Depth (Depth Pro [2])
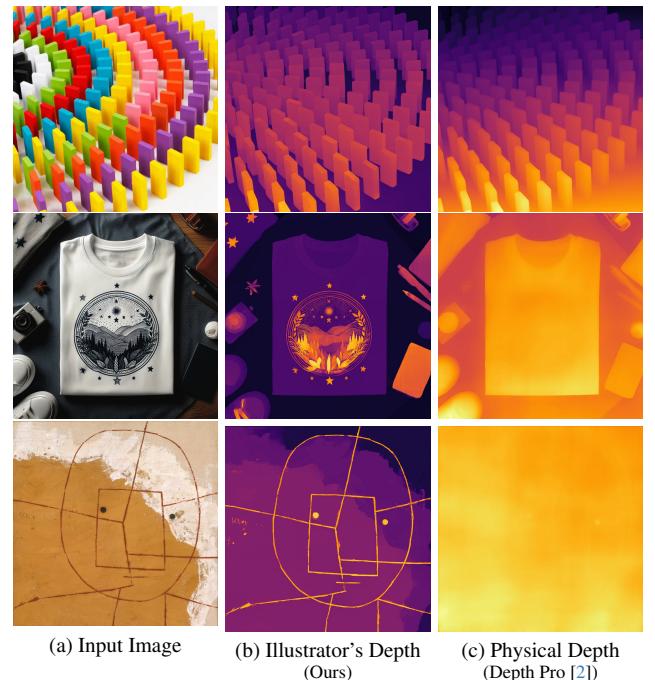
Figure 2. **Physical *vs.* Illustrator's Depth.** Unlike monocular depth estimation, illustrator's depth (middle, in false colors) produces piecewise-flat regions corresponding to layers and preserves compositional ordering even for printed or flat elements (e.g., shadows, drawings, or textures) that lack real-world depth (right).
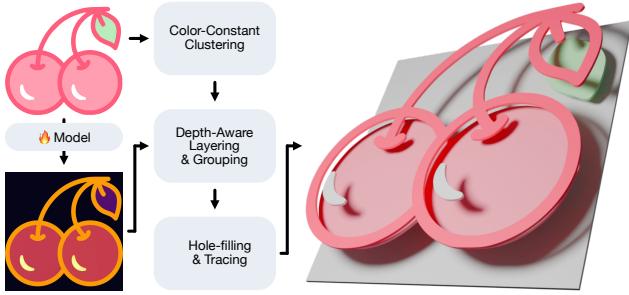
Figure 3. **Depth-aware Image Vectorization.** Our predicted illustrator's depth map (bottom left) can be integrated in traditional vectorization pipelines to produce well-layered SVG images (right, in 3D for clarity). On this example, our model allows the grouping of two disconnected white clusters to form a single background layer, while accurately separating the white highlights.

els include [25, 30, 51, 65]. Occlusion-aware and amodal transformers refine completion and boundary reasoning [5, 13, 19, 46], yet supervision and metrics remain instance-centric or pairwise. None imposes a single, transitive ordering across all pixels, which is the target of our globally-consistent ordinal layer map.

Generative decompositions for editing. Inspired by traditional approaches [7, 37, 45], editing-focused decompositions produce per-subject RGBA layers to facilitate local edits. Examples include real-time human matting [21], generative pipelines that output editable layers for subjects and effects [20, 53], and atlas-based video methods that unwrap scenes into a few textures with an alpha channel for temporal consistency [18, 22]. These layers are effective for targeted edits but are *independent* and not constrained to a global, per-pixel depth order. Instead, we seek a single "illustrator's depth" map that provides an coherent ordering of *all* pixels in order to facilitate further editing.

Layering in vectorization. An obvious application of our layer index estimation is vectorization: given our per-pixel ordinal map, standard raster-to-vector pipelines can group paths by layer and export edit-ready stacks. Existing systems based on heuristics or optimization [11, 18, 23, 29, 62] often fail to infer a clean, useful layering. Learning-based approaches [22, 34, 38, 39, 55] can, in principle, learn layer order from examples, but their training often compounds all the steps of the vectorization process (including Bézier control points), resulting in frequent reconstruction failures on complex inputs. Very recent works explore explicit layer predictions for better editing [44, 50], but remain limited in the amount of paths and details they generate. Instead, our layer index prediction provides a supervised signal for *ordering itself*, allowing traditional vectorizers to assemble SVGs in a manner most useful for further editing.

3. Method

We now introduce our notion of illustrator's depth in Sec. 3.1, before describing our dataset curation in Sec. 3.2, and finally presenting our neural network implementation and training in Sec. 3.3. Evaluation tests and ablation studies will be presented and discussed at length in Sec. 4.

3.1. Illustrator's Depth

From an input *illustration I*, represented as a raster $H \times W$ RGB image, we refer to its *illustrator's depth* as the mapping from each image pixel of the input to a layer index $i \in \{1...N\}$. Conceptually, this map represents how an artist might have structured the image as a composition of N separate layers (see Fig. 3), each corresponding to a different element or object drawn at a particular depth. Thus, illustrator's depth provides a per-pixel layer assignment that captures an interpretable notion of structural depth implicit in the artist's compositional workflow, which can then be directly leveraged for editing purposes. This paper proposes predicting this mapping from an image using a neural network and a curated training set of layered compositions, yielding an illustrator's depth image $D_\theta(I) \in \mathbb{R}^{H \times W}$ where depth is treated as a continuous value rather than a discrete layer index as it still captures relative ordering while allowing straightforward binning into discrete values if necessary.

3.2. Curating a Training Dataset

Training our network to predict Illustrator's Depth requires a large-scale dataset of images paired with their ground-truth layer structure. Scalable Vector Graphics (SVG) files are an ideal source for this data, as they are inherently composed of layered vector paths that define the stacking order of a composition. We leverage this property by developing a three-stage data preparation pipeline: first, we source a suitable dataset of layered SVGs; second, we curate it to reduce ambiguity; and finally, we rasterize the vector files into corresponding image and depth map pairs for training.

Data sourcing. While SVGs provide a structural foundation, the quality of their layering is crucial. Many SVG

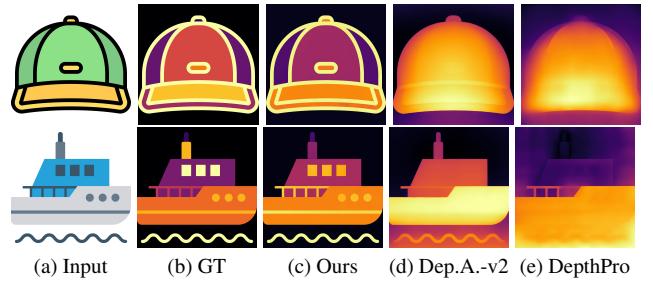(a) Input (b) GT (c) Ours (d) Dep.A.-v2 (e) DepthPro

Figure 4. **Predicted illustrator's depth evaluation.** Conventional monocular depth models (DepthAnything-v2 [54] (d), Depth-Pro [2] (e)) predict physical depth; in contrast, our model (c) accurately infers layer indices suitable for illustration decomposition.

3

datasets, while visually correct when rendered, contain disorganized or programmatically generated layers that do not reflect an artist's intent. Yet, effective learning depends on a dataset with intuitively and consistently structured compositions. After reviewing existing options, we selected the MMSVG-Illustration dataset [55], which features SVGs where elements are layered in a consistent and meaningful way, with layers systematically organized from the lowest index for the background to the highest index for the foreground, and outline strokes always placed above their corresponding color fills for instance.

Data curation. Even a high-quality dataset like MMSVG contains inherent ambiguities that can hinder learning. Artistic layering is often subjective; for instance, multiple distinct objects might logically share the same depth level, and different artists may have different layering habits. This variability can create a noisy training signal. To normalize these variations and create a more consistent ground truth, we perform two curation steps. First, we merge consecutive layers that share the same RGB color to simplify the structure. Second, we identify and exclude ambiguous cases where non-consecutive layers of the same color overlap in the final rendered image, as this significantly improves training stability.

Ground-truth rasterization. Once the SVG dataset is curated, the final step is to generate the rasterized image-depth pairs for training. For each curated SVG file, we generate its corresponding RGB input image I and ground-truth illustrator's depth map $D(I)$ of size $H \times W$ through a custom rasterization process. First, we create a temporary version of the SVG where each layer's original color is replaced by a unique color representing its layer index i in base 256: the index is thus encoded across the RGB channels via

$$\left(i \bmod 256, \lfloor i/256 \rfloor \bmod 256, \lfloor i/256^2 \rfloor \bmod 256 \right).$$

We then rasterize this modified SVG; the resulting "false color" image is converted back into a per-pixel integer depth map using the formula $D(I) = R + 256 \cdot G + 256^2 \cdot B$. This encoding strategy allows us to efficiently represent a large number of layers with virtually no additional data loading overhead. All the resulting pairs $\{I_k, D(I_k)\}_k$ of images and their illustrator's depths form our training dataset.

3.3. Neural Network & Training

Model. Predicting illustrator's depth requires reasoning about object boundaries, occlusion, and grouping. While distinct from physical depth estimation, this task benefits immensely from the powerful priors learned by state-of-the-art monocular depth estimation (MDE) models. In particular, we find that Depth Pro [2], built on Dino-v2 [26] and equipped with a multi-scale encoder, provides a robust feature extractor that allows our model to generalize well from our training set of simple vector graphics to complex, artistic images, as we will demonstrate in Sec. 4. We initialize

our model with Depth Pro's pre-trained weights, leveraging its learned understanding of geometry and occlusion as a crucial prior for our task to enable broad generalization.

Scale-invariant loss function. In natural images, distant objects correspond to large physical depth values, which are inherently more challenging to estimate accurately. Therefore, most MDE models [2, 31, 54] learn *inverse* depth values $1/d$, prioritizing the accuracy of foreground objects over distant ones. In contrast, illustrations are composed in a structured, layer-wise manner from background to foreground, where depth values typically range from 1 to N. In this setting, estimating the illustrator's depth is not inherently harder for background layers than for foreground ones. Instead of learning in disparity space, we thus train our model to predict discrete ground-truth layer indices $(1, ..., N)$ directly, assigning *equal* importance to all image layers (please see ablation studies in Sec. 4.1). Our primary objective, however, is to recover the correct *relative* ordering of these layers rather than their absolute index values. To focus the training on this relative structure, and remain robust to the potentially large range of N, we adopt a scale-invariant normalization scheme similar to MiDaS [31]. For any depth map D, we compute its median m and mean absolute deviation s, and normalize each depth value d as $\hat{d} := (d - m)/s$. We then train the network using a Mean Absolute Error (MAE) loss on these normalized maps, *i.e.*, using the loss:

$$L_{\text{MAE}}(D(I), D_\theta(I)) = \overline{|\hat{D}(I) - \hat{D}_\theta(I)|}. \quad (1)$$

Training. The network is trained on our SVG dataset using standard training practices, including data augmentation (color jitter, random inversion, random blur) and a cosine learning rate schedule. Additionally, we follow [2] by emplying two distinct learning rates for the encoder (DINO-v2 [26]) and the CNN-based decoder. Details are provided in Sec. 4, the Supplementary Material, and the code.

Post-processing. As our network outputs pixel-wise illustrator's depth estimates, *optional* post-processing can be applied to derive discrete layer indices. Depending on the target application, two common strategies are advisable: (1) direct segmentation of depth values using binning or thresholding, and (2) clustering in RGB space followed by assigning each cluster its median depth. We typically adopt the first strategy for raster image processing (Sec. 4.4), whereas the second is better suited for vectorization tasks (Secs. 4.2 and 4.3) where inputs typically exhibit color-consistent regions. In the latter case, clusters with similar colors and depths can be further merged to simplify the resulting SVG paths (see Fig. 5). Notably, even without post-processing, our predicted illustrator's depth maps are visually coherent and structurally clean, see Figs. 1, 2, 4 and 8.
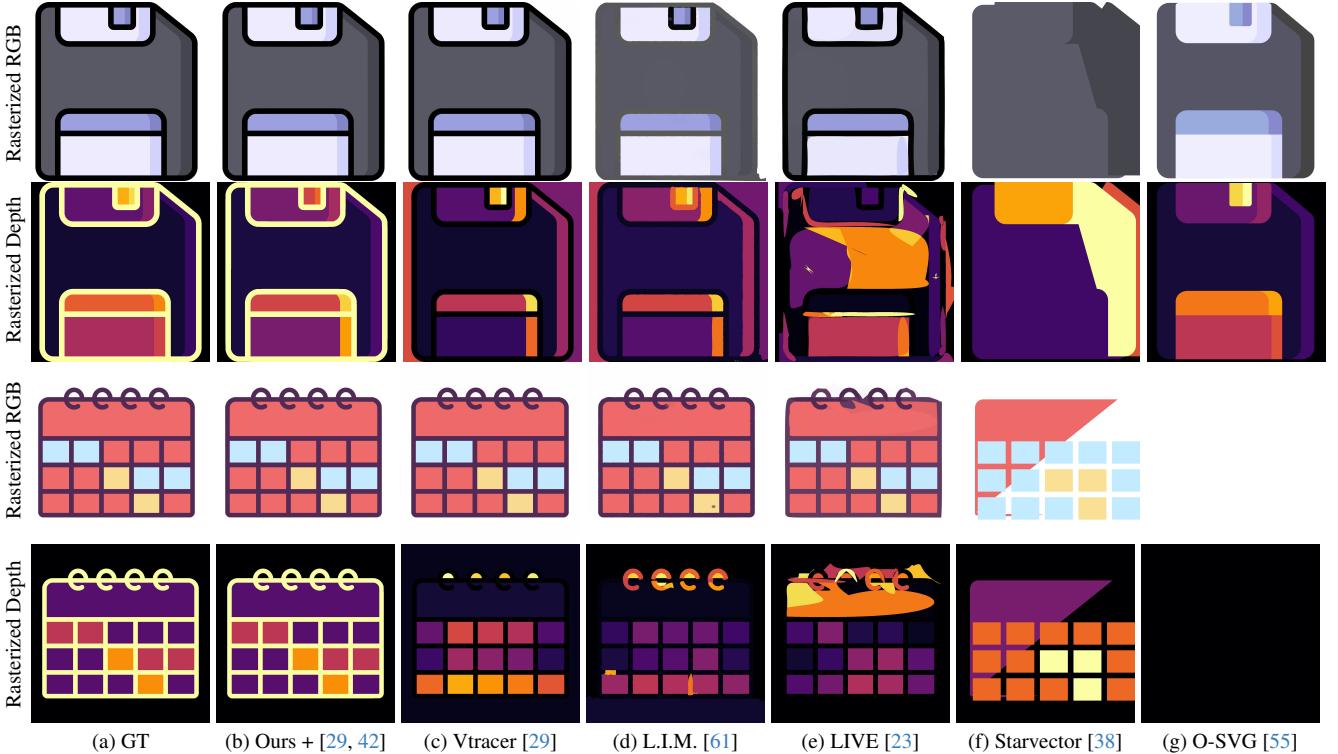
4

Figure 5. **Image vectorization with illustrator's depth.** Paired with standard vectorization pipelines, our method produces editable, depth-ordered SVGs that closely preserve the structure of the input image. Compared to heuristic. optimization-driven, or learning-based baselines, our approach systematically yields much cleaner layering and higher visual fidelity.

Left labels (top to bottom): Rasterized RGB, Rasterized Depth, Rasterized RGB, Rasterized Depth

Column labels: (a) GT   (b) Ours + [29, 42]   (c) Vtracer [29]   (d) L.I.M. [61]   (e) LIVE [23]   (f) Starvector [38]   (g) O-SVG [55]

4. Experiments and Applications

In this section, we outline our training setup in Sec. 4.1 and benchmark our model against state-of-the-art monocular depth estimators. Then we demonstrate a variety of applications of illustrator's depth. First, we embed our trained model into a vectorization pipeline (Sec. 4.2), which outperforms state-of-the-art methods, and show how it enables a creative, fully editable workflow when paired with generative image models (Sec. 4.3). We then showcase diverse raster-based editing tools enhanced by our predicted illustrator's depths (Sec. 4.4), including relief generation for tactile graphics and layer-wise decomposition.

4.1. Predicting Illustrator's Depth

Training. As detailed in Sec. 3, our model is trained on the MMSVG-Illustration dataset [55]. Following data cleaning

Table 1. **Evaluation of illustrator's depth on MMSVG.** While models trained for physical depth perform poorly, our method achieves near-perfect layer ordering and low layering error.

	Order ↑	MAE ↓	MSE ↓
Depth Pro [2]	0.636	1.44	4.76
Depth Anything-v2 [54]	0.791	1.16	3.58
Ours	**0.987**	**0.12**	**0.26**

Table 2. **Impact of key components on illustrator's depth.** Removing depth prior, data cleaning, or training in disparity space degrades layer-order consistency and accuracy, at times significantly, confirming the contribution of each component to the overall performance of our layer index predictions.

Depth prior initialization	Data cleaning	Direct index training	Order ↑	MAE ↓	MSE ↓
	✓	✓	0.903	0.51	1.17
✓		✓	0.905	0.53	1.21
✓	✓		0.980	0.50	1.88
✓	✓	✓	**0.981**	**0.16**	**0.29**

and rasterization to a resolution of 1536×1536, the dataset comprises approximately 100K consistently layered SVG images, with 80% allocated for training and 20% reserved for evaluation. In line with [61], we randomly select 100 images for quantitative analysis — see Supplementary Material for results on the SVGX-Core dataset. Training is done for 40 epochs on 8 Nvidia® A100 GPUs, with a cosine learning rate schedule, a max learning rate of $5 \cdot 10^{-6}$, and a batch size of 8.

Baselines. We compare our approach with two state-of-the-art monocular depth estimation (MDE) methods, Depth Pro [2] and Depth Anything-v2 [54].

Metrics. We evaluate performance by rendering illustrator's depth maps from ground-truth SVGs as described in

Table 3. **Evaluation of the vectorization.** Here, we test different vectorization methods — grouped by layering strategies — on the validation set of the MMSVG dataset. Our approach achieves the best combination of layering accuracy, path compactness, and reconstruction fidelity, systematically outperforming heuristic, optimization-based, and data-driven baselines.

Method	Layering Prior	Layering Quality				Visual Fidelity		
		Order ↑	MAE ↓	MSE ↓	Path Number ↓	MSE ($\times 10^{-2}$) ↓	SSIM ↑	LPIPS ↓
Vtracer [29]	Heuristics	0.689	2.58	15.67	3.65	0.023	0.994	0.022
Less Is More [61]	Heuristics	0.746	2.43	21.10	5.54	0.663	0.961	0.043
LIVE [23]	Optimization-based	0.838	4.88	96.91	8.62	0.297	0.946	0.053
Starvector [38]	Data-driven	0.918	1.52	9.75	0.53	9.123	0.858	0.302
OmniSVG [55]	Data-driven	0.925	1.31	8.08	0.54	9.997	0.830	0.317
Ours + [29, 42]	Data-driven	**0.987**	**0.46**	**2.09**	**0.16**	**0.018**	**0.997**	**0.005**

Sec. 3.2. Since each method produces depth estimates in its own scale, we first normalize all predicted depth maps using the procedure described in Sec. 3.3 prior to computing Mean Squared Error (MSE) and Mean Absolute Error (MAE). While both MSE and MAE assess pixel-wise depth accuracy, many of our target applications require a *globally consistent layer ordering* rather than *precise* depth values. Therefore, following Zhang et al. [60], we further evaluate *depth ordering consistency* by randomly sampling pixel pairs from the ground truth and predictions, and checking whether their relative depth order is preserved (see Supplementary Material for details). The resulting *depth ordering consistency* metric (abbreviated as Order in Tabs. 1-3) measures the percentage of correctly ordered pixel pairs, providing a complementary measure of global depth consistency.

Results. While related, physical depth and illustrator's depth do capture fundamentally different concepts (Fig. 2). Standard MDE models, trained to predict real-world geometry, struggle to recover correct layer ordering in illustrations (Fig. 4); our model, purposely trained to infer layer indices, achieves markedly better results, outperforming all baselines by a wide margin (Tab. 1). Inference takes less than one second on current GPUs as reported in [2].

Ablation studies. We conduct a series of ablation studies to validate our design choices discussed in Sec. 3. As detailed in Tab. 2, both Depth Pro initialization (leveraging a physical depth prior from weights learned on millions of images) and data cleaning (removing inconsistencies and ambiguities in ground-truth layers) boost the depth ordering consistency quite sharply. Although training directly with layer indices $(1, ..., N)$ instead of disparity space $(1/d)$ yields comparable global ordering scores, it facilitates a more balanced optimization between foreground and background layers: this results in better depth transitions and a clear advantage across *all* evaluation metrics; see the Supplementary Material for additional qualitative evaluations.

4.2. Vectorization

Image vectorization, which consists in converting raster images to vector graphics, is a particularly straightforward ap-

plication of illustrator's depth.

Pipeline. Our model integrates seamlessly into existing vectorization pipelines such as VTracer [29], where we replace area-based sorting heuristics with our predicted illustrator's depth. We first compute color clusters, sort them using our layer index prediction, inpaint layers to fill holes and bridge gaps (with, e.g., Scikit-Image [47]), before vectorizing each layer with potrace [42]. The whole process, including our illustrator's depth prediction, only takes seconds.

Baselines. We benchmark our pipeline against key state-of-the-art approaches, based on simple area heuristics (VTracer [29]) or more advanced cluster-sorting strategies (Less Is More [61]), optimization methods (LIVE [23]), and LLM-based tools (StarVector [38], OmniSVG [55]).

Metrics. Vectorization demands both compactness and accuracy for best editability. We thus measure *layering quality* using the depth ordering consistency (Order), mean squared error (MSE), and mean absolute error (MAE), as well as path count errors $|N-\tilde{N}|/N$ to compare the number of paths in ground-truth (N) vs. reconstructed ($\tilde{N}$) SVGs. We then evaluate *visual fidelity* by measuring the rasterized output compared the input using MSE in RGB space, Structural Similarity Index Measure (SSIM) [64], and Learned Perceptual Image Patch Similarity (LPIPS) [58].

Results. Although most vectorization methods produce outputs that look quite close to the input raster images, visualizing their layer indices in false colors reveals substantial differences in layering quality (Fig. 5). Methods relying on heuristics such as VTracer [29] and Less is More [61] frequently misorder layers; for instance, spiral binding holes in the calendar in Fig. 5 are incorrectly positioned on top despite belonging to the background. Optimization-based LIVE [23] introduces spurious layers and shapes, while LLM-based approaches [38, 55] often fail (sometimes, spectacularly) to achieve full reconstruction. In contrast, our pipeline is able to faithfully reconstruct the input while producing layer indices close to the ground truth. Additionally, quantitative results from Tab. 3 confirm these observations: our method matches VTracer's reconstruc-

tion fidelity while outperforming all SOTA competitors in layer-index accuracy. Interestingly, our layering evaluation reveals a clear divide between methods excelling at reconstruction but weak in layering (VTracer, Less is More) and those with opposite strengths (Starvector, OmniSVG). Our approach thus combines the power of traditional vectorizers with the quality of data-driven layer index prediction, enabling state-of-the-art performance on both fronts.

4.3. Text-to-Vector-Graphics Generation

The creation of high-quality vector graphics remains a challenging problem. Direct generation techniques, such as those employing Score Distillation Sampling (SDS) [27, 57] or Large Language Models (LLMs) [38, 39, 55], have not yet matched the visual fidelity achieved by state-of-the-art text-to-image generative models. Here again, our illustrator's depth neural prediction can dramatically help in obtaining high-quality editable illustrations.

Pipeline. Leveraging recent advances in high-quality image generation [9, 17], we first generate vector-style raster images (prompts are detailed in the Supplementary Material). These raster images are subsequently transformed into structured, editable, and layered SVG using our specialized vectorization pipeline described in Sec. 4.2.

Results. Fig. 6 presents examples generated via Flux [17] and postprocessed with illustrator's depth. The resulting SVG illustrations exhibit high visual complexity and coherent layer organization, facilitating the intuitive grouping and editing of individual elements (see supplementary video). Our vectorization can be similarly integrated to Nano Banana [9] to offer a more advanced, multi-stage generative workflow as illustrated in Fig. 7. We also show comparisons with Neural Path Representation [57], NeuralSVG [27], and LayerTracer [44], in the Supplementary Material.
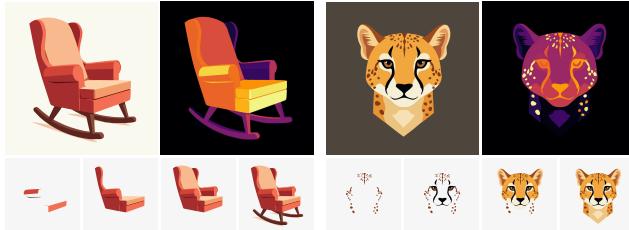

Figure 6. **Vector graphics generation.** By augmenting text-to-image diffusion models like Flux [17] with illustrator's depth, generated images can be automatically transformed into editable vector graphics. Layers (bottom, displayed from front to back) facilitate intuitive manipulation of individual elements.

4.4. Beyond Vector Graphics

Despite being trained exclusively on depth data generated from simple SVG images, our model demonstrates a remarkable ability to generalize beyond this narrow scope. It

Figure 7. **Illustrator's depth in generative workflows.** Starting from a cellphone photo and a rug texture (left), a pipeline based on Nano Banana [9] and illustrator's depth synthesizes a vector-graphics illustration and converts it into a layered SVG, supporting depth-aware editing such as recoloring and object insertion.

successfully infers illustrator's depth across highly diverse inputs, from complex illustrations and artistic renderings, to even natural images, due to our use of pretrained priors [2, 26] learned from millions of images. This section showcases two practical applications leveraging this strong generalization. Additional qualitative results and discussions of failure cases are provided in Figs. 1 & 2, and in the Supplementary Material.

    (a) Input Image   (b) Illustrator's Depth   (c) Relief, 3D rendering

Figure 8. **Automatic relief generation from single images.** With no manual intervention, illustrator's depth (middle) can be converted into 3D surfaces by interpreting predicted depth as elevation. The resulting meshes, shown on the right, demonstrate how images can be transformed into tactile or printable reliefs.

4.4.1. Automatic Relief Generation From a Single Image

Task. Relief is a sculptural method where elements remain attached to a solid background to give the impression that the sculpture has been raised above the background. Basrelief, a shallow form of this technique, is widely applied, from coinage to architectural ornament [59]. Current methods for generating 3D reliefs from 2D images are fundamentally limited by their reliance on user-defined depth ordering [35]. We eliminate this user interaction entirely by leveraging the fully automated output of our model.

Pipeline. Given an input image, our system first generates a pixel-wise illustrator's depth map $d_\theta(i, j)$. This depth is then directly used to build a triangulated surface by transforming each pixel into a vertex with 3D coordinates $(i, j, d_\theta(i, j))$, and triangulating adjacent vertices.

Results. The resulting mesh easily integrates into any 3D application as illustrated in Fig. 8. Crucially, our illustrator's depth transforms flat paintings into 3D objects without any manual annotation, offering an alternative, intuitive, and tangible interaction with works of art.

4.4.2. Depth-Based Editing

Task. Raster image editing relies on the composition of multiple layers. Existing segmentation tools, however sophisticated they have become, often fall short because they perform based on ambiguous requests: if a user clicks a face, do they mean the face, the whole character, or the entire foreground? Our work helps resolve this ambiguity, as enriching input images with our predicted illustrator's depth dramatically facilitates layer separation.

Pipeline. Illustrator's depth is easily leveraged to inform segmentation: based on a user-defined threshold value t adjustable in realtime via a slider, an image can be split into two layers, one (foreground) defined as illustrator's depths
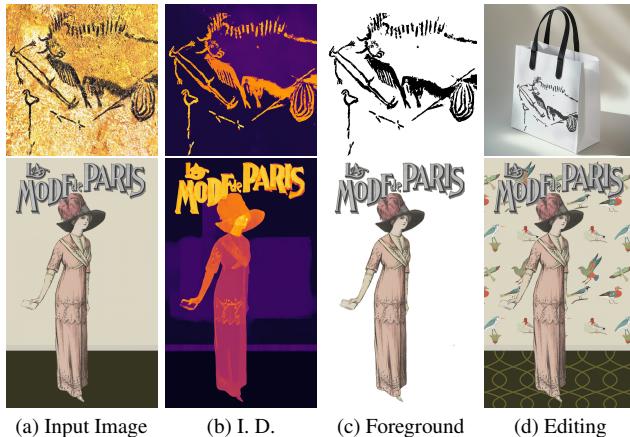

(a) Input Image (b) I. D. (c) Foreground (d) Editing

Figure 9. **Depth-Aware Image Editing.** From an input image (a), illustrator's depth (b) enables selective separation of key image regions (c) for seamless compositing or depth-aware insertion (d).
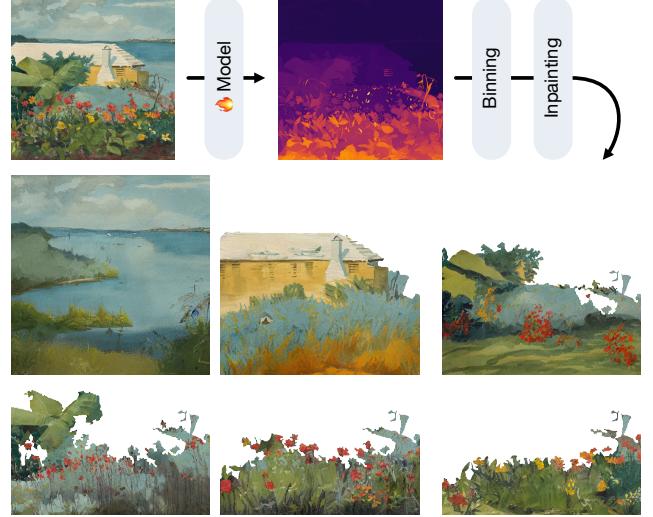

Figure 10. **Illustrator's depth with an inpainting model.** Given our illustrator's depth (top), we can bin the values into several layers and inpaint each occluded regions with Stable Diffusion [40]. The resulting set of overlapping layers can be directly used for 3D parallax effects as demonstrated in our supplemental video.

satisfying $D[i, j] > t$ and one (background) for all others. More generally, any binning strategy into N layers, found through a quick analysis of the entire map D or derived manually, provides a decomposition into layers by ranges of illustrator's depths, which can be directly uploaded in raster graphic editors to allow for direct editing.

Results. Illustrator's depth within the context of raster image editing provides a robust mechanism for selective element isolation as demonstrated in Fig. 9. Paired with any inpainting model such as [40], our method can produce N overlapping layers to allow for parallax effects for instance, see Fig. 10. Additional examples can be found in the accompanying Supplementary Material and video.

5. Conclusion and Future Work

We introduced Illustrator's Depth, a novel concept that augments image pixels with additional layer indices, enabling straightforward decomposition into an edit-ready stack. Trained on a curated dataset of SVG files, our network can infer illustrator's depth across a wide range of inputs, ranging from simple icons to complex raster graphics. We demonstrated that our method achieves SOTA performance in image vectorization and facilitates a number of downstream tasks beyond vector graphics, such as text-to-vector generation, interactive editing, and relief generation.

Although our current model is trained specifically for this task with a curated dataset of SVGs, the rapid advancement of vision models toward one-shot and zero-shot generalization [48] suggests a near-future where illustrator's depth could be inferred directly from natural prompts, with-

out explicit training. Beyond its current technical form, we believe that the underlying concept of illustrator's depth will remain relevant across a variety of creative domains: by shifting the notion of *depth* from a physical metric to a layer-based ready-to-edit abstraction, our work introduces a new paradigm for intelligent creative tools to better assist the artistic process. Illustrator's depth transforms image decomposition from a mere technical challenge into a creative and assistive foundation for the next generations of computational art and design systems.

6. Acknowledgments

References

[1] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4009–4018, 2021. 2

[2] Alexey Bochkovskiy, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan Richter, and Vladlen Koltun. Depth Pro: Sharp monocular metric depth in less than a second. In *The Thirteenth International Conference on Learning Representations*, 2025. 1, 2, 3, 4, 5, 6, 7

[3] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12475–12485, 2020. 2

[4] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022. 2

[5] Helisa Dhamo, Keisuke Tateno, Iro Laina, Nassir Navab, and Federico Tombari. Peeking behind objects: Layered depth prediction from a single image. *Pattern Recognition Letters*, 125:333–340, 2019. 2, 3

[6] David Eigen, Christian Puhrsch, and Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014. 2

[7] Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. Photo2clipart: Image abstraction and vectorization using layered linear gradients. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017. 3

[8] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018. 2

[9] Google. Gemini 2.5 Flash Image, 2025. 7, 2, 3, 4

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[11] Or Hirschorn, Amir Jevnisek, and Shai Avidan. Optimize & reduce: a top-down approach for image vectorization. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence*, pages 2148–2156. AAAI Press, 2024. 2, 3

[12] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007. Publisher: Springer. 2

[13] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4019–4028, 2021. 3

[14] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6399–6408, 2019. 2

[15] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019. 1

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023. 2

[17] Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux. 1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025. 7, 2, 3

[18] Ho Law and Sung Ha Kang. Image Vectorization with Depth: convexified shape layers with depth ordering. *SIAM Journal on Imaging Sciences*, 18(2):963–1001, 2025. 2, 3

[19] Hyunmin Lee and Jaesik Park. Instance-wise occlusion and depth orders in natural scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21210–21221, 2022. 3

[20] Yao-Chih Lee, Erika Lu, Sarah Rumbley, Michal Geyer, Jia-Bin Huang, Tali Dekel, and Forrester Cole. Generative omnimatte: Learning to decompose video into layers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12522–12532, 2025. 2, 3

[21] Shanchuan Lin, Andrey Ryabtsev, Soumyadip Sengupta, Brian L Curless, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Real-time high-resolution background matting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8762–8771, 2021. 3

[22] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7930–7939, 2019. 3

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

[24] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (ToG)*, 38(4):1–14, 2019. 2

[25] Rohit Mohan and Abhinav Valada. Amodal panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21023–21032, 2022. 3

[26] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael

Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, 2024. arXiv:2304.07193 [cs]. 4, 7

[27] Sagi Polaczek, Yuval Alaluf, Elad Richardson, Yael Vinker, and Daniel Cohen-Or. NeuralSVG: An Implicit Representation for Text-to-Vector Generation, 2025. arXiv:2501.03992 [cs]. 7, 3

[28] Yifan Pu, Yiming Zhao, Zhicong Tang, Ruihong Yin, Haoxing Ye, Yuhui Yuan, Dong Chen, Jianmin Bao, Sirui Zhang, and Yanbin Wang. Art: Anonymous region transformer for variable multi-layer transparent image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7952–7962, 2025. 2

[29] Sanford Pun and Chris Tsang. VTracer. *https://github.com/visioncortex/vtracer*, 2025. 2, 3, 5, 6

[30] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2019. 3

[31] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. Publisher: IEEE. 2, 4

[32] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025. 2

[33] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, and Laura Gustafson. SAM 2: Segment anything in images and videos. In *The Thirteenth International Conference on Learning Representations*, 2025. 1

[34] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J. Mitra. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351, 2021. 3

[35] Andreas Reichinger, Stefan Maierhofer, and Werner Purgathofer. High-quality tactile paintings. *Journal on Computing and Cultural Heritage*, 4(2):1–13, 2011. 8

[36] Babak Rezaeirowshan, Coloma Ballester, and Gloria Haro. Monocular Depth Ordering using Perceptual Occlusion Cues:. In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, pages 431–441, Rome, Italy, 2016. SCITEPRESS - Science and Technology Publications. 2

[37] Christian Richardt, Jorge Lopez-Moreno, Adrien Bousseau, Maneesh Agrawala, and George Drettakis. Vectorising bitmaps into semi-transparent gradient layers. In *Computer Graphics Forum*, pages 11–19. Wiley Online Library, 2014. 3

[38] Juan A. Rodriguez, Abhay Puri, Shubham Agarwal, Issam H. Laradji, Pau Rodriguez, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. StarVector: Generating scalable vector graphics code from images and text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16175–16186, 2025. 3, 5, 6, 7

[39] Juan A Rodriguez, Haotian Zhang, Abhay Puri, Aarash Feizi, Rishav Pramanik, Pascal Wichmann, Arnab Mondal, Mohammad Reza Samsami, Rabiul Awal, Perouz Taslakian, et al. Rendering-aware reinforcement learning for vector graphics generation. *arXiv preprint arXiv:2505.20793*, 2025. 3, 7

[40] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685, New Orleans, LA, USA, 2022. IEEE. 8

[41] A. Saxena, Min Sun, and A.Y. Ng. Make3D: Learning 3D Scene Structure from a Single Still Image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5): 824–840, 2009. 2

[42] Peter Selinger. Potrace: a polygon-based tracing algorithm. *https://potrace.sourceforge.net*, 2003. 5, 6, 2

[43] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, 1998. 2

[44] Yiren Song, Danze Chen, and Mike Zheng Shou. Layertracer: Cognitive-aligned layered svg synthesis via diffusion transformer. *arXiv preprint arXiv:2502.01105*, 2025. 2, 3, 7

[45] Jianchao Tan, Jyh-Ming Lien, and Yotam Gingold. Decomposing images into layers via rgb-space geometry. *ACM Transactions on Graphics (TOG)*, 36(1):1–14, 2016. 3

[46] Minh Q Tran, Khoa HV Vo, Kashu Yamazaki, Arthur Fernandes, Michael T Kidd, and Ngan Le. Aisformer: Amodal instance segmentation with transformer. In *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press, 2022. 3

[47] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014. 6, 2

[48] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners, 2025. arXiv:2509.20328 [cs]. 8

[49] Daniel Winter, Matan Cohen, Shlomi Fruchter, Yael Pritch, Alex Rav-Acha, and Yedid Hoshen. Objectdrop: Bootstrapping counterfactuals for photorealistic object removal and insertion. In *European Conference on Computer Vision*, pages 112–129. Springer, 2024. 2

[50] Ronghuan Wu, Wanchao Su, and Jing Liao. LayerPeeler: Autoregressive Peeling for Layer-wise Image Vectorization, 2025. arXiv:2505.23740 [cs]. 2, 3

11

[51] Yuting Xiao, Yanyu Xu, Ziming Zhong, Weixin Luo, Jiawei Li, and Shenghua Gao. Amodal segmentation based on visible region segmentation and shape prior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2995–3003, 2021. 3

[52] Ximing Xing, Juncheng Hu, Guotao Liang, Jing Zhang, Dong Xu, and Qian Yu. Empowering llms to understand and generate complex vector graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19487–19497, 2025. 1

[53] Jinrui Yang, Qing Liu, Yijun Li, Soo Ye Kim, Daniil Pakhomov, Mengwei Ren, Jianming Zhang, Zhe Lin, Cihang Xie, and Yuyin Zhou. Generative image layer decomposition with visual effects. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 7643–7653, 2025. 2, 3

[54] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth Anything V2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024. 1, 2, 3, 4, 5

[55] Yiying Yang, Wei Cheng, Sijin Chen, Xianfang Zeng, Jiaxu Zhang, Liao Wang, Gang Yu, Xingjun Ma, and Yu-Gang Jiang. OmniSVG: A Unified Scalable Vector Graphics Generation Model, 2025. arXiv:2504.06263 [cs]. 3, 4, 5, 6, 7, 1

[56] Lvmin Zhang and Maneesh Agrawala. Transparent Image Layer Diffusion using Latent Transparency. *ACM Trans. Graph.*, 43(4):100:1–100:15, 2024. 2

[57] Peiying Zhang, Nanxuan Zhao, and Jing Liao. Text-to-Vector Generation with Neural Path Representation. *ACM Trans. Graph.*, 43(4):36:1–36:13, 2024. 7, 3

[58] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 6

[59] Yu-Wei Zhang, Jing Wu, Zhongping Ji, Mingqiang Wei, and Caiming Zhang. Computer-assisted Relief Modelling: A Comprehensive Survey. *Computer Graphics Forum*, 38(2):521–534, 2019. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13655. 8

[60] Ziyu Zhang, Alexander G Schwing, Sanja Fidler, and Raquel Urtasun. Monocular object instance segmentation and depth ordering with cnns. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2614–2622, 2015. 2, 6, 3

[61] Kaibo Zhao, Liang Bao, Yufei Li, Xu Su, Ke Zhang, and Xiaotian Qiao. Less is more: Efficient image vectorization with adaptive parameterization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18166–18175, 2025. 2, 5, 6

[62] Hengyu Zhou, Hui Zhang, and Bin Wang. Segmentation-guided layer-wise image vectorization with gradient fills. In *European Conference on Computer Vision*, pages 165–180. Springer, 2024. 2, 3

[63] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018. 2

[64] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 6

[65] Yan Zhu, Yuandong Tian, Dimitris Metaxas, and Piotr Dollár. Semantic amodal segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1464–1472, 2017. 3

Illustrator's Depth: Monocular Layer Index Prediction for Image Decomposition

Supplementary Material

This supplementary material provides additional details, results, and comparisons to complement our CVPR paper on *Illustrator's Depth*.

7. Evaluation on SVGX

While we trained our model on (a curated subset of) the MMSVG-Illustration dataset [55], we also evaluated our layer index predictions on the SVGX-Core-250k dataset curated by [52] for completeness. Similar to MMSVG, we randomly select 100 images for quantitative analysis. As shown in Tab. 4 and Fig. 11, our model demonstrates strong generalization and maintains excellent performance.
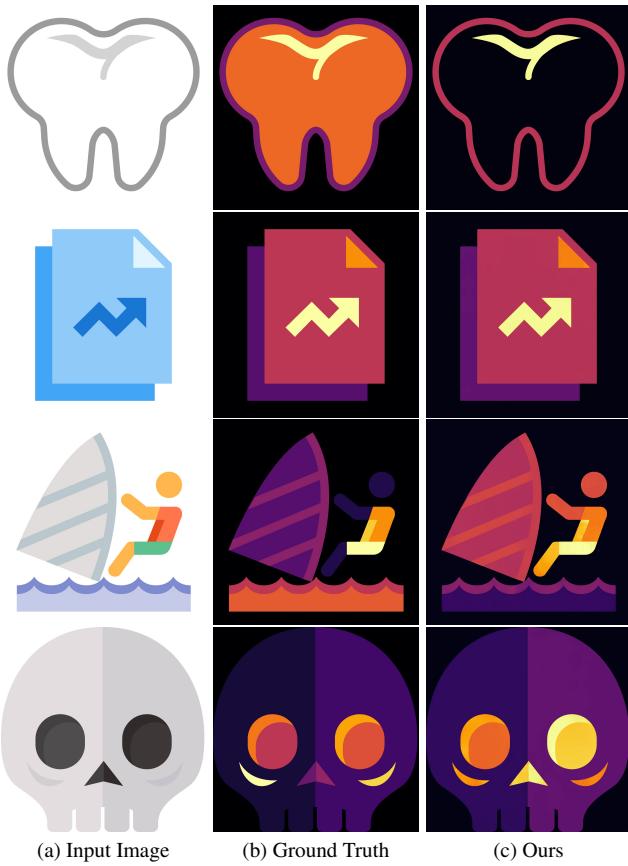
Table 4. **Evaluation of our method on different datasets predicted depth.** Raw outputs of the network.

	Order ↑	MAE ↓	MSE ↓
MMSVG	0.987	0.12	0.26
SVGX-Core-250k	0.984	0.16	0.53

8. Ablation Studies

We present additional qualitative results in Fig. 12 to complement the quantitative findings reported in Table 2 of the main paper. While data cleaning and the use of depth priors lead to pronounced improvements, the choice of layer indices vs. disparity space (d vs. $1/d$) yields more subtle effects, yet still provides noticeable gains in these examples.

Figure 11. **Evaluation of the inferred layer indices.** When evaluated on the SVGX-Core-250k dataset, our method predicts a satisfactory illustrator's depth even if some conventions are different from in our training dataset (for instance, the outline of the tooth is placed *below* the filled-in shape).

Figure 12. **Ablation studies.** Data cleaning (top row) and direct indexing (middle row) ease the burden of the model, resulting in cleaner predictions, while using a depth prior initialization (bottom row) significantly improves our model's performance.

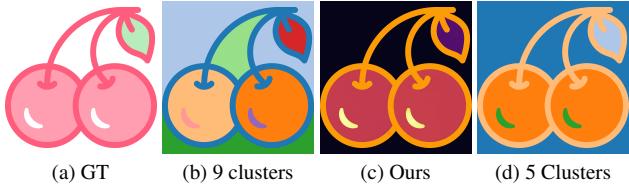(a) GT   (b) 9 clusters   (c) Ours   (d) 5 Clusters

Figure 13. **Depth-aware clustering.** Given an input image (a), VTracer [29] provides a list of color-constant clusters (b). We order these clusters based on our predicted depth map (c) and merge them to form the final decomposition (d).

9. Details on our vectorization pipeline

Our vectorization tests were performed on a pipeline combining VTracer [29] and Potrace [42] with our contributions. Specifically, illustrator's depth based vectorization is achieved as follows:

1. We find color-constant clusters using VTracer (Fig. 13b). The values of several hyper-parameters are important, such as *filter_speckle* to suppress noise, *color_precision* and *layer_difference* to accurately split the image in distinct regions. All of them are provided in our code.
2. Instead of relying on VTracer's heuristics to sort the clusters, we leverage our predicted illustrator's depth (Fig. 13c) for layering, assigning the cluster's depth order to the median of the predicted depth for each cluster.
3. Cluster grouping is important to ensure a well-layered, compact output. After sorting, we further merge layers with neighboring indices if their RGB colors are within a certain threshold $\tau = 0.05$ in the L^2 norm. This results in an ordered clustering image $C \in [1, ...N]^{H \times W}$ (Fig. 13d).
4. While it doesn't affect the final rendering, filling holes and bridging gaps yields simpler, overlapping layers that are compact and easy to edit. Given a cluster with index n, we create a binary mask $\mathbf{1}_{C[i,j]>n}$, and inpaint the missing regions of $\mathbf{1}_{C[i,j]==n}$ using off-the-shelf algorithms (see Sec. 10 and Fig. 14).
5. This layer collection is then vectorized with Potrace [42] and assembled to form the final vector graphics.

10. Inpainting

While not part of our contributions, we also show examples of inpainting strategies once our layer index prediction has been generated, see Fig. 14. For vector graphics, we rely on fast, off-the-shelf algorithms provided by Scikit-image [47]. We experimented with two variants in order to fill the missing regions: one that interpolates using the nearest unmasked point, and another based on biharmonic interpolation. Depending on the application, users may prefer one approach over the other: the biharmonic method produces smoother curves, whereas the closest-point interpolation yields sharper, crisper boundaries (see Fig. 14). We generally use the latter in our code due to its faster compu-

tational time. While this simple hole-filling approach is sufficient for most vector graphics, data-driven inpainting may be desired for more involved applications, including raster image editing: here again, leveraging off-the-shelf inpainting models (see Fig. 3) offers a solution that doesn't require any additional training.
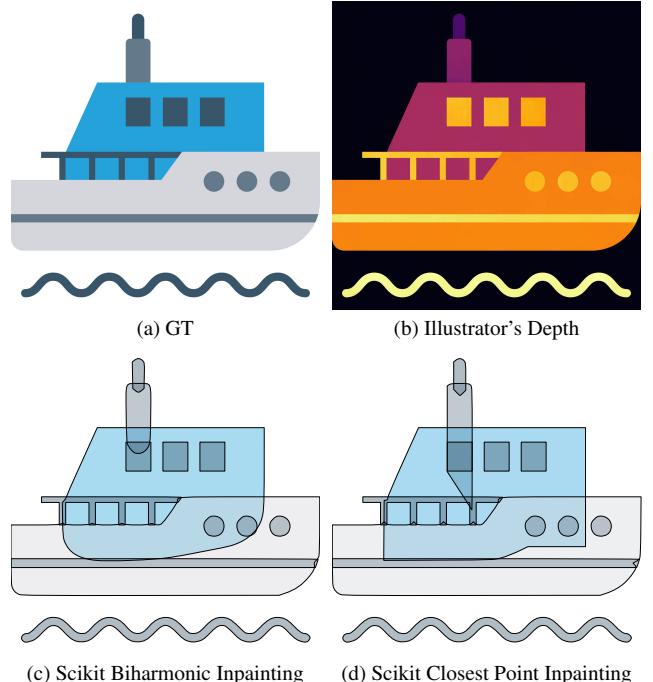

(a) GT   (b) Illustrator's Depth

(c) Scikit Biharmonic Inpainting   (d) Scikit Closest Point Inpainting

Figure 14. **Inpainting with Scikit [47].** Using the boat example (a) from Fig. 4, we display two examples of the same layer-wise decomposition produced by our method (b), where inpainting is done via a biharmonic (c) or closest point (d) variant.

11. Prompts for vector-styled images

The main paper shows two text-to-image examples, one using FLUX [17] and one using Nano Banana [9]. For FLUX (Fig. 6 in the main paper), we found that, given a desired object to be drawn (underlined below), a mix of positive and negative prompts provides clean vector-styled raster images that are easy to process with our pipeline; for instance,

```
{ "prompt": "Vector graphics of a simple
    ↪ cheetah head.",
 "prompt_2": "Vector graphics of a simple
    ↪ cheetah head. SVG file. Filled shapes,
    ↪ minimalist design. Abstract.",
"negative_prompt": "Gradient, 3D. Small
    ↪ details. Fineline details.",
"negative_prompt_2": "Gradient, 3D. Small
    ↪ details. Fineline details.",
"num_inference_steps": 28,
"num_images_per_prompt": 1
}
```

For Nano Banana (Fig. 7 in the main paper), we simply prompt the model through:

```
{ "prompt": "Vector graphic illustration of a
    ↪ cat. SVG style, blue background. Smooth,
    ↪ flowy shapes."
}
```

12. Comparison with Text2Vector Generators

In addition to the results discussed in Sec. 4.3 of the main paper, more examples of text-to-vector-graphics generations are given in Fig. 15. Both text-to-vector generations using Neural Path Representations [57] and NeuralSVG [27] are based on Score Distillation Sampling (SDS) that relies on a pretrained diffusion model to back-propagate gradients to Bézier curve parameters. Consequently, their generated illustrations are relatively simple and lack fine details (we reproduce the images provided in their articles in Fig. 15). Although LayerTracer [44] employs its own custom diffusion model, it exhibits similar limitations, producing simple emoji-like graphics; note that we used the prompting setup provided in their public repository. In contrast, our method can decompose any output into layered SVG representations, effectively decoupling generation from vectorization — and thus fully leveraging the capabilities of modern generative models. Our modular pipeline, compatible with both Flux [17] and Nano Banana [9], produces detail-rich vector illustrations within seconds (see Sec. 11 for the full prompt configurations).

13. Failure cases

Texture artifacts. Since our model is trained on clean SVG data, a failure case arises when the input image contains canvas textures or defects. These issues can be easily mitigated by using a generative model (e.g., Nano Banana [9]) to clean the image before applying our method (see Fig. 16).

Incorrect ordering. Like any machine learning model, ours can occasionally make mistakes (see Fig. 17, bottom row). Quantitatively, such errors are rare: as shown in Tab. 1, over 98% of randomly sampled pixel pairs are correctly ordered in our experiment with MMSVG.

Foreground Focus. Our training set primarily contains single objects over white backgrounds. Consequently, the model sometimes neglects background elements, which may be undesirable in certain scenarios (see Fig. 17, top row). Future work could address this limitation by training on more complex or synthetic SVG datasets that include background elements.

14. Depth ordering consistency metric

To compute the depth ordering consistency from a ground-truth illustrator's depth map D and a predicted map D_θ, we adapt the approach of [60] and proceed as follows:

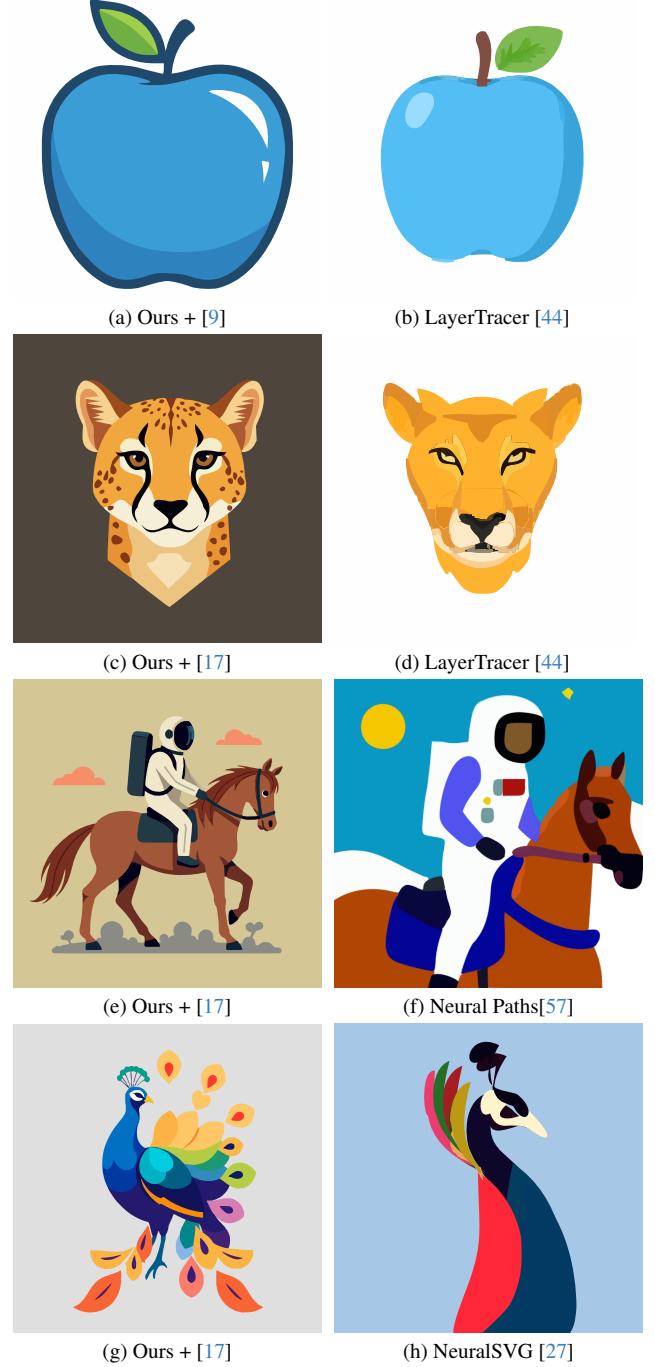| (a) Ours + [9] | (b) LayerTracer [44] |
(c) Ours + [17]	(d) LayerTracer [44]
(e) Ours + [17]	(f) Neural Paths[57]
(g) Ours + [17]	(h) NeuralSVG [27]

Figure 15. **Text2Vector models.** Pairing Illustrator's Depth with powerful image generative models produces more complex and detailed illustrations than current text-to-vector diffusion models. In our results (left column), the models are prompted as described in Sec. 11 using "a blue apple", "a cheetah head", "an astronaut riding a horse", and "a colorful peacock".

1. we uniformly sample $\frac{H \times W}{50}$ random pairs of pixel locations (i, j) and (k, l) and keep only those corresponding to two different layers in D, i.e., such that $D[i, j] \neq$

(a) Input Image

(b) Illustrator's Depth

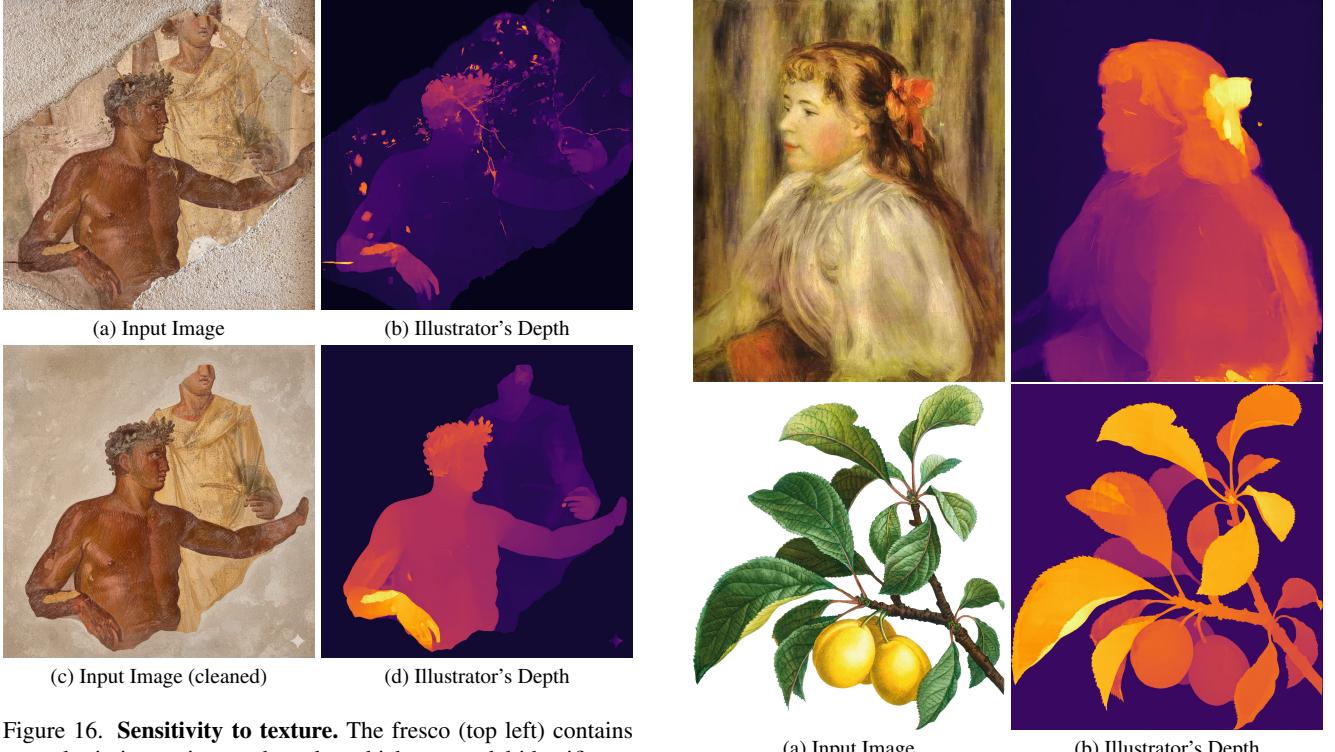(c) Input Image (cleaned)

(d) Illustrator's Depth

Figure 16. **Sensitivity to texture.** The fresco (top left) contains several missing regions and cracks, which our model identifies as foreground elements (top right). If these artifacts are undesired, one can first use Nano Banana [9] to remove defects, then reapply our model to obtain a cleaner result.

$D[k, l]$;

2. we then check whether the relative ordering is preserved by comparing the signs of $(D[i, j] - D[k, l])$ and $(D_\theta[i, j] - D_\theta[k, l])$.

3. Finally, we compute the average consistency score $\bar{s}$ over all pairs by the ratio of preserved ordering over total number of pixel pairs.

This formulation quantifies how effectively the predicted illustrator's depth maintains correct relative depths, independent of absolute scale. This metric, inherently stochastic as it relies on randomly sampled pixel pairs from the image, exhibits strong stability: sampling $50,000$ pairs on 1536×1536 images yielded no significant variations in our experiments (see Fig. 18). And as Tabs. 1-3 from the original paper demonstrate, it offers a complementary measure of layering quality.

15. Additional results

For completeness, we also provide a histogram of the number of layers present in our curated training dataset in Fig. 19, as well as a figure demonstrating another potential use of our illustrator's depth in Fig. 20, where a painting is automatically turned into a multi-layered pop-up card.

(a) Input Image

(b) Illustrator's Depth

Figure 17. **Failure cases.** Our models can ignore the background elements, such as the stripes (top row), or incorrectly predict the illustrator's depth: in the bottom row, the leftmost plum should be on top of the leaf rather than behind.
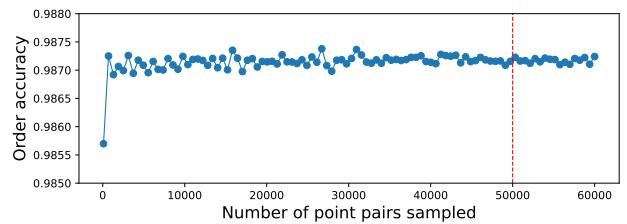

Figure 18. **Stability of order metric.** We plot the order metric when sampling one of our results from 100 to 60K points.
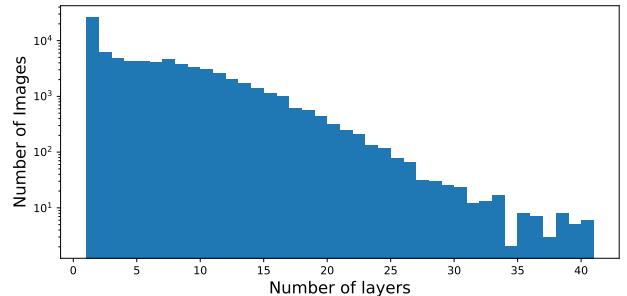

Figure 19. **Number of layers in MMSVG training set.** We plot the histogram of the number of layers in our training dataset. Note that each layer may have many connected components, resulting in a large number of paths.

4

Figure 20. **Automatic pop-up card generation.** From an image (top left) and our predicted illustrator's depth (bottom left), a multi-layered pop-up card can easily be created using our method — see video for animation.