Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

NDC: Neural Diffusion Curves for Image Vectorization

Yifan Huang
Zhejiang Sci-Tech University
Wuyu Yang
Zhejiang Sci-Tech University
Hao Xie
Communication University of Zhejiang
Yu-Kun Lai
Cardiff University
Yao Jin
jinyao@zstu.edu.cn

Zhejiang Sci-Tech University
Huaxiong Zhang
Zhejiang Sci-Tech University
Lili He
Zhejiang Sci-Tech University

Research Article

Keywords: Diffusion curves, Image vectorization, Differentiable curve extraction, Differentiable rendering, Fourier neural operator

Posted Date: May 22nd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9620952/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at The Visual Computer on July 20th, 2026.
See the published version at https://doi.org/10.1007/s00371-026-04594-9.

NDC: Neural Diffusion Curves for Image Vectorization

Yifan Huang[1,2], Wuyu Yang[1,2], Hao Xie[3], Yu-Kun Lai[4], Yao Jin[1,2*], Huaxiong Zhang[1*], Lili He[1,2*]

[1]Zhejiang Sci-Tech University, Hangzhou, China.
[2]Zhejiang Provincial Innovation Center of Advanced Textile Technology, Shaoxing, China.
[3]Communication University of Zhejiang, Hangzhou, China.
[4]Cardiff University, Cardiff, United Kingdom.

*Corresponding author(s). E-mail(s): jinyao@zstu.edu.cn; zhxhz@zstu.edu.cn; llhe@zju.edu.cn;
Contributing authors: 2023220603028@mails.zstu.edu.cn; 2025210602017@mails.zstu.edu.cn; xiehao@cuz.edu.cn; LaiY4@cardiff.ac.uk;

Abstract

We present Neural Diffusion Curves (NDC), a differentiable framework that vectorizes raster images into compact, editable diffusion-curve representations in a single forward pass. Unlike traditional multi-stage pipelines that rely on hand-crafted feature extraction and non-differentiable numerical solvers, NDC unifies curve geometry prediction, bilateral color control point extraction, and PDE-based rendering within a learnable architecture. Specifically, a Transformer-based curve decoder with optimal-transport matching produces a data-adaptive set of sparse Bézier curves; a lightweight 1D convolutional network extracts color constraints along curve normals; and a Fourier Neural Operator (FNO) serves as a differentiable surrogate for the Poisson solver, enabling gradient propagation across the entire pipeline. On **512 × 512** flat illustrations, NDC achieves about **18×** pipeline-level speedup over the fastest classical diffusion-curve baseline and a nearly **900×** speedup over the slowest, while reducing the curve count by over **75%** and producing significantly longer, semantically coherent curves. NDC attains competitive perceptual fidelity (LPIPS) at the cost of a moderate increase in pixel-level error (RMSE), a trade-off that favors applications where speed, compactness, and direct editability are of primary importance.

Keywords: Diffusion curves, Image vectorization, Differentiable curve extraction, Differentiable rendering, Fourier neural operator

1 Introduction

Smooth vector graphics rely on compact geometric primitives to represent continuous color variations like soft shadows and gradients. Diffusion curves (DCs) [1] excel at this by defining color constraints along curves and solving a Poisson equation, ensuring global smoothness and physical consistency. They find wide applicability in vector illustration and graphic design, particularly for flat illustrations and other piecewise-smooth content with sharp boundaries, the regime that the DC formulation natively targets. Yet traditional DC vectorization remains inefficient, producing fragmented curves and relying on non-differentiable numerical solvers that preclude end-to-end learning. Recovering both geometry and color from a raster while preserving compactness and editability is a challenging inverse problem.

Existing vectorization methods fall into three main categories, each with limitations. Traditional DC pipelines [1–3] use handcrafted stages (edge detection → color sampling → numerical optimization), leading to low efficiency and fragmented curve outputs. Differentiable SVG optimization [4] enables end-to-end training but lacks the PDE (Partial Differential Equation) formulation of DCs, limiting its ability to model smooth shading. Neural implicit methods [5, 6] offer generative flexibility but sacrifice explicit geometric editability that is essential for practical editing.

To address these, we propose Neural Diffusion Curves (NDC), a fully automated pipeline that learns DC representations directly from raster images (Fig. 1). It comprises: (1) a Transformer-based network predicting sparse multi-segment Bézier curves; (2) a color extraction network that samples features along curve normals and predicts sparse bilateral color control points along each curve; and (3) a differentiable Fourier Neural Operator (FNO) [7] that approximates the Poisson solver, enabling efficient differentiable rendering.

NDC produces compact, directly editable curve representations and reduces per-image processing from seconds to approximately 0.1 s, making diffusion-curve vectorization feasible for interactive applications. In summary, our contributions are:

1. A unified differentiable pipeline that replaces handcrafted multi-stage processing with a staged learnable framework for automatic raster-to-DC conversion.
2. Decoupled geometry and appearance modeling via a Transformer-based curve decoder with optimal transport based SimOTA matching [8], producing a data-adaptive number of compact curve predictions in a single forward pass.
3. A differentiable FNO-based renderer that serves as an efficient surrogate for the Poisson solver, enabling gradient-based optimization across modules.
4. A large-scale dataset providing aligned supervision for curve geometry, color attributes, and rendered images, supporting DC vectorization research.

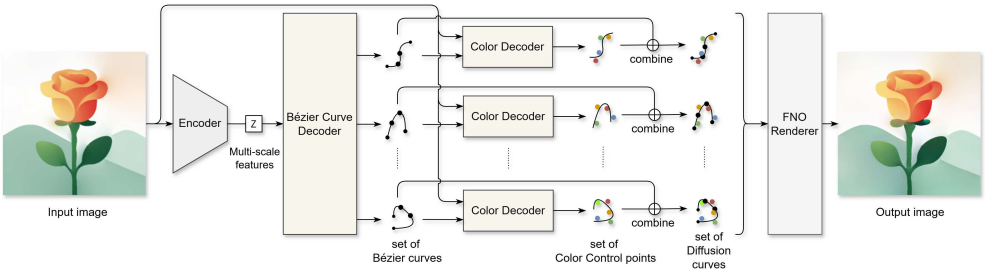


Fig. 1: Overall framework of the proposed Neural Diffusion Curves. The pipeline comprises a multi-scale feature encoder, a Transformer-based curve decoder that predicts Bézier geometry, a per-curve color decoder that predicts bilateral color control points, and an FNO (Fourier Neural Operator)-based differentiable renderer that approximates the Poisson solver to reconstruct the image.

2 Related Work

Image vectorization converts raster images into resolution-independent geometric representations. Early contour-based methods such as Potrace [9] and piecewise Bézier fitting [10] work well for flat-color illustrations but produce fragmented polygons in regions with smooth shading, compromising compactness [11, 12]. To model continuous tone variations more compactly, subsequent research has developed representations rooted in diffusion curves, mesh-based primitives, differentiable path optimization, and neural implicit models.

Diffusion curves (DCs) [1] define color constraints along spline curves and reconstruct the image by solving a Poisson equation, guaranteeing globally smooth color transitions while preserving sharp boundaries. The formulation has been extended with diffusion coefficients [13], Poisson vector graphics [14], and hybrid boundary integral methods [15]. Classical DC rendering relies on numerical solvers such as multigrid [1], finite elements [16], or GPU-accelerated Laplacian solvers [17], which are accurate but computationally expensive and inherently non-differentiable. Automating the inverse problem, i.e., recovering curve geometry and bilateral colors from a raster, has led to several multi-stage pipelines: Orzan et al. [1] combine edge detection, color sampling, and Douglas–Peucker simplification; Xie et al. [2] fit DCs hierarchically in the Laplacian domain; and Zhao et al. [3] cast the task as inverse shape optimization with FEM rendering. While these methods produce high-quality results, they remain fragmented, difficult to integrate into end-to-end learning, and typically slow [11]. Video extensions [18] inherit the same limitations.

Mesh-based representations approximate an image by partitioning the domain into triangular or rectangular patches and defining a color function inside each element. Gradient meshes [19] assign colors to vertices and interpolate them smoothly, producing globally continuous fields suited for photorealistic vector graphics; automatic topology-preserving generation [20], feature-adaptive subdivision [21], and TCB-splines [22] have improved flexibility and discontinuity handling. Image triangulation, in contrast, uses a simple polynomial (often constant) per triangle, deliberately

trading smoothness for an extremely compact, stylised low-poly look [23, 24]. Modern mesh-based methods increasingly leverage differentiable objectives to refine the representation: Fang et al. [25] reduce triangle count under an error bound via differential evolution, Wang et al. [26] employ cubic Bézier edges and differentiable rendering to directly minimize reconstruction loss, and He et al. [27] use Monte Carlo integration to build a differentiable error function that jointly optimizes geometry and per-face colors. Compared to diffusion curves, mesh-based representations require explicit construction that conforms to image content and do not provide the same edge-aligned, curve-level editability; our work focuses on the DC paradigm, leaving hybrid DC-mesh approaches as future work.

Differentiable vector graphics reformulate vectorization as gradient-based inverse optimization, primarily for SVG-like path primitives. DiffVG [28] introduced a differentiable SVG rasterizer, enabling primitive-wise optimization. This sparked a line of layer-wise decomposition methods: LIVE [4] progressively stacks filled SVG paths, later refined by autoregressive peeling [29] and convexified shape layers with depth ordering [30]. Orthogonal approaches reduce representational complexity by top-down path reduction [31] or adaptive parameterization [32]. Other differentiable frameworks explore alternative objectives and representations, including gradient-domain reconstruction [33], linear gradient layer decomposition [34], and texture-encapsulated shape parameterization for artistic imagery [35]. Despite these advances, most methods rely on standard SVG primitives filled with solid colors or simple gradients. Without a PDE-based formulation, they often require many overlapping shapes to approximate smooth shading, trading off compactness and direct editability compared to DC-based models.

Neural implicit methods introduce learned priors to overcome the sensitivity of optimization-based vectorization. Early works already explored neural networks for compact image representation, e.g., vector regression functions [36] that use a multilayer perceptron to encode image signals. More recently, NIVeL [6] decomposes vector content into implicit layers without discrete control points, and NeuralSVG [5] extends implicit modeling to text-to-vector generation. Neural style transfer for vector graphics [37] incorporates artistic constraints. Although these approaches offer generative flexibility, they operate on latent representations and sacrifice explicit, editable curve geometry grounded in a PDE formulation, limiting their applicability in interactive design and editing scenarios.

In summary, existing methods illustrate a persistent tension between compact editability, PDE-based smooth shading, and end-to-end differentiability. Classical DC pipelines achieve mathematically expressive gradients but are inefficient and non-differentiable; differentiable SVG frameworks enable learning-based optimization yet cannot model globally harmonic interpolation without a PDE; neural implicit methods provide powerful generative priors but forgo explicit curve-level control. Our Neural Diffusion Curves (NDC) bridge these gaps by combining a Transformer-based curve predictor, a learnable bilateral color extractor, and an FNO-based differentiable Poisson surrogate within a unified learnable pipeline, achieving fast, compact, and fully editable diffusion-curve vectorization.

3 Method

3.1 Overview

We formulate image vectorization as an inverse problem: given a raster image $\mathcal{I} \in \mathbb{R}^{H\times W\times 3}$, we aim to predict a collection of diffusion curves $\mathcal{S} = \{s_i\}_{i=1}^{N}$ that can reproduce $\mathcal{I}$ through a differentiable rendering process. This task is challenging because the mapping from a sparse set of curves to a continuous color field is governed by a partial differential equation, making the inverse problem highly non-linear and ill-posed. Traditional pipelines decompose this challenge into a sequence of hand-crafted stages (edge detection, color sampling, numerical optimization), each tuned independently. Our goal with Neural Diffusion Curves (NDC) is to replace this fragmented pipeline with a unified learnable framework that directly predicts compact and editable curve representations while maintaining the physical fidelity of the diffusion curve model.

Each curve s_i is defined by its geometry (a three-segment cubic Bézier curve) and two sets of color constraints (left and right) along the curve. The geometry is parameterized by control points $\mathbf{P}_i \in \mathbb{R}^{10\times 2}$ (ten 2D points), and the colors by sparse control points $\mathbf{C}_{i,l}, \mathbf{C}_{i,r} \in \mathbb{R}^{K\times 4}$, where each control point stores (t, r, g, b) with $t \in [0, 1]$ being the normalized arc-length parameter and (r, g, b) the RGB color. This parameterization naturally separates the inverse problem into three coupled subproblems: locating boundary-aligned curve geometry, assigning compact bilateral color constraints, and propagating these constraints through the diffusion process. We therefore design the pipeline to mirror this structure, so that each module addresses one physically meaningful part of the diffusion-curve representation.

The proposed pipeline (Fig. 1) consists of four components that operate in sequence. The multi-scale feature encoder is a ResNet-50 backbone with a Feature Pyramid Network (FPN), which produces multi-scale feature maps $\{F_l\}_{l=1}^{4}$ that capture both fine details and semantic context for accurate curve localization. These features are consumed by a Transformer-based curve decoder, which iteratively refines a fixed set of learnable queries through self-attention and deformable cross-attention to output the geometric parameters of each curve; the set-valued formulation removes the need for heuristic post-processing such as non-maximum suppression or greedy linking. Given the predicted geometry, a per-curve color decoder samples features along curve normals and predicts sparse bilateral color control points along each curve. Finally, the differentiable renderer is a Fourier Neural Operator that approximates the Poisson solver and maps the resulting curve constraints to a reconstructed image $\hat{\mathcal{I}}$. By operating in the frequency domain, the FNO has a global receptive field with near-linear complexity and supports gradient flow from the rendered image back to the predicted color controls, which is required by the staged training scheme described below.

In our implementation, the curve prediction module, the renderer, and the color extraction module are trained in a staged manner to reduce the discrepancy between $\hat{\mathcal{I}}$ and $\mathcal{I}$. This unified framework reduces the reliance on traditional multi-stage pipelines and improves the final reconstruction quality under staged training.

3.2 Multi-scale Feature Encoder

The feature encoder follows a standard ResNet-50-FPN architecture. The input image is passed through the backbone network to produce feature maps at multiple scales. FPN then generates four multi-scale outputs F_1, F_2, F_3, and F_4, each with channel dimension $d = 256$. These feature maps serve as input to the transformer decoder. The multi-scale representation is crucial because curves may appear at various scales (e.g., fine details vs. coarse boundaries), and the deformable attention mechanism can selectively sample from the appropriate levels.

3.3 Transformer-based Curve Geometry Extraction

Predicting multiple curves from an image is a structured output problem: different images contain different numbers of valid curves, predictions must avoid duplicates while respecting inter-curve adjacency, and there is no natural ordering among them. Traditional methods rely on heuristics like non-maximum suppression and greedy linking. Query-based Transformers, as pioneered in object detection (DETR), are well-suited for this task because they directly predict a fixed-size candidate set via learnable queries and bipartite matching, while a per-query existence classifier discards background queries to yield a data-adaptive number of output curves. The self-attention mechanism allows the model to reason about inter-curve relationships such as suppressing duplicates, while deformable cross-attention focuses computation on relevant image regions. The benefit of the resulting set-valued formulation, in particular the dynamic SimOTA matching introduced below, is examined empirically in the matching ablation (Tab. 3).

Bézier curve parameterization. Each predicted curve consists of three cubic Bézier segments with shared junctions between adjacent segments. The three segments $(B_j(u), j = 0, 1, 2)$ are defined as:

$$B_j(u) = \sum_{i=0}^{3} \binom{3}{i} (1-u)^{3-i} u^i P_{3j+i}, u \in [0, 1], \tag{1}$$

where the shared junctions $\mathbf{P}_3$ and $\mathbf{P}_6$ enforce positional continuity. To further encourage approximate G^1 consistency at these junctions during training, we additionally constrain the tangent directions of adjacent segments to be aligned through a continuity loss (see $\mathcal{L}_{\text{cont}}$ below). The final ten control points are thus $\{\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_9\}$. This representation balances expressiveness (three segments can model complex bends) and compactness (only ten points per curve), and is easier to edit and scale than polyline approximations that introduce unnecessary vertices.

Decoder architecture. We instantiate the decoder with $N_q = 80$ learnable query vectors by default. Each query predicts at most one candidate curve and is independently scored by an existence classifier (see $\mathcal{L}_{\text{cls}}$ below); queries classified as background are discarded at inference, so the number of output curves per image is data-adaptive and averages 16.9 on our test set (Tab. 1). The decoder consists of $L = 6$ identical layers. Each layer contains the following components:

1. **Layer Normalization & Self-Attention**: Normalize input query features and apply 8-head self-attention to model inter-curve spatial correlations.
2. **Deformable Cross-Attention** [38]: Normalize self-attention outputs and perform multi-scale deformable cross-attention with encoder features $\{\mathbf{F}_l\}_{l=1}^{4}$ to align queries with image spatial features.
3. **Feed-Forward Network (FFN)**: Normalize cross-attention outputs and refine features via a two-layer FFN (GELU activation, hidden dimension 1024).
4. **Geometric Parameter Regression**: Let $\mathbf{z}_l \in \mathbb{R}^{N_q \times d}$ denote the query feature tensor after the l-th decoder layer. The output of the final layer $\mathbf{z}_L$ ($L = 6$) is fed into a three-layer MLP that regresses, for each query, the 10 2D control points of a three-segment cubic Bézier curve.

Our decoder formulation establishes an end-to-end mapping from multi-scale image features to compact curve geometry. Through iterative refinement, each query progressively integrates inter-curve relationships with multi-scale spatial features, yielding coherent and compact Bézier curve representations. The overall decoder architecture is illustrated in Fig. 2.

Assignment strategy. To assign predictions to ground-truth curves, we adapt SimOTA from object detection [8]. The cost matrix $\mathcal{C}_{\text{match}}(i, j)$ is defined as a weighted sum of the classification cost and the Chamfer distance between sampled curve points: $\mathcal{C}_{\text{match}}(i, j) = \alpha C_{ij}^{\text{cls}} + \beta C_{ij}^{\text{chamfer}}$. The Chamfer distance is computed over 60 uniformly sampled points in normalized coordinates, and we set $\alpha = 1$ and $\beta = 3$ in all experiments. For each ground-truth curve i, the top-k_i predictions with the smallest assignment costs are designated as positive samples, where k_i is determined per ground truth as

$$k_i = \min\Big(k_{\max},\ \big|\{j : \mathcal{C}_{\text{match}}(i, j) < \tau_i\}\big|\Big), \tag{2}$$

with $k_{\max} = 5$ and τ_i set to the cost of the 10-th best prediction for ground-truth i (an image-adaptive percentile threshold). Empirically $k_i \in [1, 5]$ across our dataset. This encourages one ground truth to match multiple predictions, improving recall and providing richer supervision, especially for long or complex curves. Unlike fixed bipartite matching, SimOTA adapts to the complexity of each image, leading to more stable training and better coverage.

Training objective. To ensure high-quality curve predictions, we employ a composite loss that targets different aspects of geometric fidelity:

$$\mathcal{L}_{\text{geo}} = \lambda_1 \mathcal{L}_{\text{cls}} + \lambda_2 \mathcal{L}_{\text{coord}} + \lambda_3 \mathcal{L}_{\text{shape}} + \lambda_4 \mathcal{L}_{\text{cont}}, \tag{3}$$

with $\lambda_1 = 1, \lambda_2 = 3, \lambda_3 = 3, \lambda_4 = 0.1$ by default. These weights were selected on the validation set to balance classification, coordinate, shape, and continuity supervision after normalization.

- **Classification loss $\mathcal{L}_{\text{cls}}$**: Focal loss [39] over N_q queries with SimOTA matching to distinguish curves from background and suppress false positives.
- **Coordinate regression $\mathcal{L}_{\text{coord}}$**: Weighted ℓ_1 loss on the ten control points, emphasizing endpoints and junctions ($\omega_{\text{end}} = 2.0, \omega_{\text{mid}} = 1.0$) for precise localization. Here $\mathcal{P}$ denotes the set of matched (ground-truth, prediction) index pairs produced by

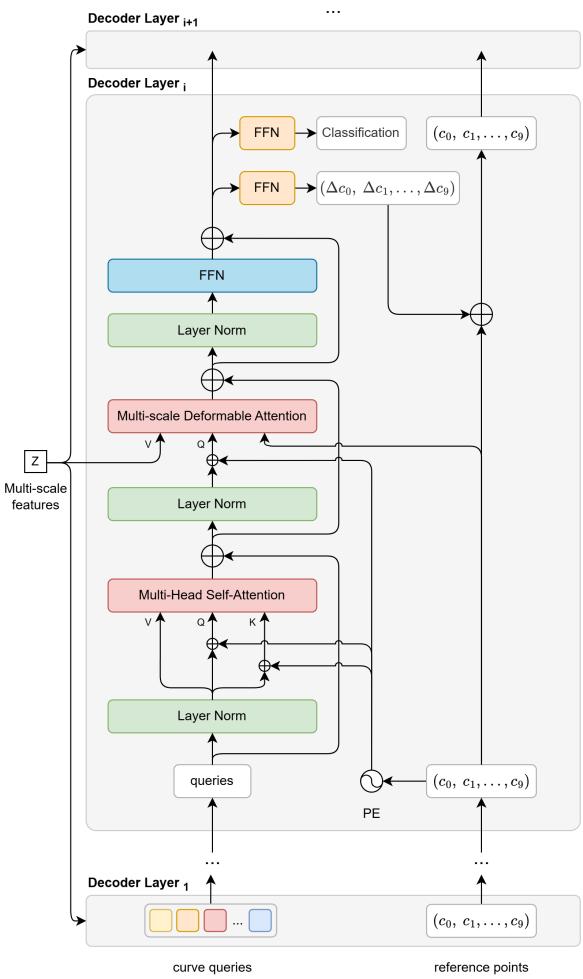


Fig. 2: Architecture of the Transformer-based curve extraction decoder.

SimOTA, and $M = |\mathcal{P}|$ is the number of matched pairs; $\hat{\mathbf{P}}_{j,m}$ is the m-th predicted control point for query j:

$$\mathcal{L}_{\text{coord}} = \sum_{(i,j)\in\mathcal{P}} \frac{1}{M} \sum_{m=0}^{9} \omega_m \left\| \mathbf{P}_{i,m} - \hat{\mathbf{P}}_{j,m} \right\|_1 . \tag{4}$$

where $\omega_m = \omega_{\text{end}}$=2.0 for the four endpoints/junctions $m \in \{0, 3, 6, 9\}$ and $\omega_m = \omega_{\text{mid}}$=1.0 otherwise.

- **Shape consistency $\mathcal{L}_{\text{shape}}$**: Samples $T = 20$ uniformly spaced parameter values $\{u_t\}_{t=1}^{T}$ per segment ($\lambda_{\text{len}} = 0.1, \epsilon = 10^{-6}$). $\hat{B}_j^{(s)}$ is the s-th predicted Bézier segment for query j, and $\ell(\cdot)$ denotes the approximate arc length computed by summing

chord lengths over T sampled points:

$$\mathcal{L}_{\text{shape}} = \sum_{(i,j)\in\mathcal{P}} \left[\frac{1}{3T} \sum_{s=1}^{3} \sum_{t=1}^{T} \left\| B_i^{(s)}(u_t) - \hat{B}_j^{(s)}(u_t) \right\|_1 + \lambda_{\text{len}} \frac{\left| \ell(\hat{B}_j) - \ell(B_i) \right|}{\ell(B_i) + \epsilon} \right] \quad (5)$$

- **Continuity loss $\mathcal{L}_{\text{cont}}$**: Cosine distance between unit tangent vectors at the two interior junctions ($n = 1, 2$). $\mathbf{v}_{\text{pre}}^n$ and $\mathbf{v}_{\text{cur}}^n$ are the outgoing tangent of segment $n-1$ and the incoming tangent of segment n at their shared junction, respectively:

$$\mathcal{L}_{\text{cont}} = \sum_{(i,j)\in\mathcal{P}} \sum_{n=1}^{2} \left(1 - \cos\langle \mathbf{v}_{\text{pre}}^n, \mathbf{v}_{\text{cur}}^n \rangle\right). \quad (6)$$

3.4 Curve Color Extraction

Extracting accurate and compact side colors from the input image is challenging. Traditional normal-direction sampling followed by DP (Douglas–Peucker) simplification is heuristic and non-differentiable. Our learnable module samples RGB and gradient features along normals, processes them with a 1D CNN, and regresses sparse control points, enabling end-to-end optimization.

Normal-profile feature extraction. Given predicted geometry, for each curve we uniformly sample $N_s = 128$ points $\{\mathbf{p}_i\}$ along $[0, 1]$ with unit normals $\mathbf{n}_i$. Around each point, $M = 9$ positions are sampled along the normal:

$$\mathbf{p}_{i,m} = \mathbf{p}_i + \delta_m \mathbf{n}_i, \quad \delta_m \in \{-4, \ldots, 4\} \text{ (pixels)}, \quad (7)$$

collecting RGB (bilinear interpolation) and gradient components (Sobel). Feature $\mathbf{f}_i \in \mathbb{R}^{9\times5+2}$ concatenates RGB, gradients, and the normal vector, forming a sequence $\{\mathbf{f}_i\}_{i=1}^{128}$.

Sparse control-point prediction. A lightweight 1D CNN (five residual Conv1D blocks, kernel size 5, hidden 128) processes this sequence and produces two outputs: dense per-point left/right colors $\mathbf{c}_i^L, \mathbf{c}_i^R \in [0, 1]^3$ via sigmoid activation, and $K - 2$ interior parametric positions $t_k^L, t_k^R \in (0, 1)$ per side ($K = 5$) regressed by a global average pooling followed by an MLP.

The two endpoints are fixed at $t_0 = 0$ and $t_{K-1} = 1$; the $K - 2$ interior positions $\{t_k\}_{k=1}^{K-2}$ are regressed as continuous scalars. The color at each regressed position t_k is obtained by piecewise linear interpolation in t on the dense per-point color sequence: locating the two neighboring sample indices on the 128-point grid, $\mathbf{c}_k$ is computed as a convex combination of the two sample colors with weights smoothly determined by t_k. The operation is therefore differentiable in t_k, and gradients propagate back to the position regressor during training. This adaptive placement captures color variations compactly, while continuous t regression keeps the operation differentiable and avoids the discrete-index quantization of fixed-grid sampling.

Training objective. The color network is trained against the input raster image through the frozen FNO renderer:

$$\mathcal{L}_{\text{color}} = \|\hat{\mathcal{I}} - \mathcal{I}\|_1, \tag{8}$$

where $\hat{\mathcal{I}}$ is the image rendered by the frozen FNO renderer from the input tensor X constructed using the predicted sparse color set $\mathcal{C}$ (dynamically sampled with $K = 5$ control points per side via the regressed coordinates t_k). We use ℓ_1 rather than ℓ_2 for robustness to outliers caused by occasional boundary mis-localization, where a few mispredicted pixels can otherwise dominate the loss.

3.5 Differentiable Rendering via FNO

Rendering diffusion curves requires solving a Poisson equation, traditionally via iterative numerical solvers that are computationally expensive and non-differentiable. Since the color extractor is supervised through image reconstruction, the renderer must be both efficient and differentiable; otherwise each color update would require an expensive numerical solve. We therefore employ a FNO [7, 40] as a differentiable surrogate, learning the solution operator in the frequency domain with $\mathcal{O}(N \log N)$ complexity and enabling single-pass rendering.

Input representation. Let $\Omega = [0, H] \times [0, W]$ denote the (continuous) image domain, discretized on the $H \times W$ pixel grid. The FNO renderer takes a tensor $X \in \mathbb{R}^{H \times W \times 6}$ with channel layout $[w_x, w_y, M_{\text{bound}}, R, G, B]$. Let $\mathbf{w} = (w_x, w_y) : \Omega \to \mathbb{R}^2$ denote the channel-aggregated gradient constraint vector field induced by the diffusion curves: each curve is uniformly sampled along arc length and its contributions are bilinearly splatted to nearby pixels; at a sample corresponding to curve i at parameter t, $\mathbf{w}$ is defined from the channel-averaged bilateral color contrast $\big(\mathbf{c}_{i,l}(t) - \mathbf{c}_{i,r}(t)\big)$ along the normal direction $\mathbf{n}_i(t)$, and contributions from multiple samples that fall into the same pixel are averaged. Here $\mathbf{c}_{i,l}(t)$ and $\mathbf{c}_{i,r}(t)$ denote the bilateral colors at parameter t obtained by piecewise-linear interpolation of the predicted control points $\mathbf{C}_{i,l}, \mathbf{C}_{i,r}$, and $\mathbf{n}_i(t)$ is the unit normal at t. The third channel is a binary boundary mask M_{bound} covering curve pixels and the image border, and the last three channels store the boundary colors $C_{\text{bound}} \in \mathbb{R}^3$ on the masked pixels. This representation is fully differentiable, allowing gradients to flow back to the color network.

FNO architecture. The FNO first lifts the 6-channel input to a hidden dimension of $d_{\text{fno}} = 96$ channels, then applies $N_F = 4$ Fourier layers. Let $v_t \in \mathbb{R}^{H \times W \times 96}$ denote the hidden feature map after the t-th layer; each layer performs:

$$v_{t+1}(\mathbf{x}) = \sigma\Big(W v_t(\mathbf{x}) + \mathcal{F}^{-1}\big(R \cdot \mathcal{F}(v_t)\big)(\mathbf{x})\Big), \tag{9}$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the 2D discrete Fourier transform and its inverse, respectively; $R \in \mathbb{C}^{96 \times 96 \times m \times n}$ ($m = n = 48$) is a learnable spectral weight matrix truncated to the lowest 48 frequencies in each spatial dimension; W is a learned 1×1 convolution providing a local skip path; and σ denotes the GELU activation. A final linear projection yields RGB $\hat{\mathcal{I}} \in [0, 1]^{H \times W \times 3}$ (Fig. 3). The FNO is pre-trained to mimic a numerical Poisson solver, then frozen during subsequent training.

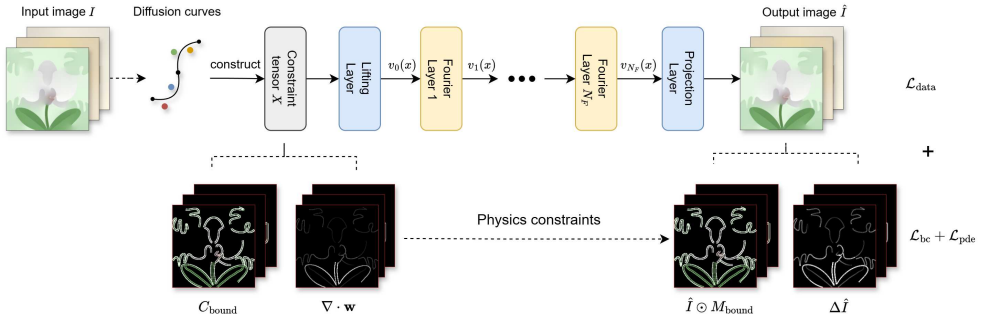


Fig. 3: FNO-based differentiable renderer approximating the Poisson solver. The 6-channel constraint tensor X is lifted, processed by N_F Fourier layers, and projected to the reconstructed image $\hat{I}$, which is supervised by the data term $\mathcal{L}_{\text{data}}$. The boundary term $\mathcal{L}_{\text{bc}}$ aligns $\hat{I} \odot M_{\text{bound}}$ with C_{bound}, and the PDE residual $\mathcal{L}_{\text{pde}}$ aligns $\Delta\hat{I}$ with $\nabla \cdot \mathbf{w}$ on interior pixels.

Training objective. The FNO renderer is pre-trained on our constructed dataset (Sec. 4.1) to mimic a numerical Poisson solver, with the input X constructed from ground-truth curves and colors. The training objective combines a data term, a Dirichlet boundary term, and a PDE residual term:

$$\mathcal{L}_{\text{fno}} = \lambda_{\text{data}}\mathcal{L}_{\text{data}} + \lambda_{\text{bc}}\mathcal{L}_{\text{bc}} + \lambda_{\text{pde}}\mathcal{L}_{\text{pde}}, \tag{10}$$

using $\lambda_{\text{data}} = 1$, $\lambda_{\text{bc}} = 0.1$, $\lambda_{\text{pde}} = 0.1$.

- $\mathcal{L}_{\text{data}} = \|\hat{\mathcal{I}} - \mathcal{I}_{\text{gt}}\|_2^2$ (MSE over all pixels). This is the primary supervision, encouraging the FNO to output images close to the ground-truth renderings.
- $\mathcal{L}_{\text{bc}} = \|(\hat{\mathcal{I}} - C_{\text{bound}}) \odot M_{\text{bound}}\|_2^2$ enforces Dirichlet conditions, ensuring the rendered image respects the specified boundary colors.
- $\mathcal{L}_{\text{pde}} = \|\Delta\hat{\mathcal{I}} - \nabla \cdot \mathbf{w}\|_{\text{interior}}^2$ is the Poisson equation residual measured over interior pixels ($M_{\text{bound}} = 0$), where Δ is the discrete Laplacian and $\nabla \cdot \mathbf{w}$ is the divergence of the channel-aggregated gradient constraint field, directly penalizing violations of the underlying PDE and improving physical consistency of the recovered color field.

After pre-training, the FNO is frozen for all subsequent training stages.

3.6 Staged Training Procedure

Jointly optimizing all three modules from scratch is unstable because the curve extractor, the renderer, and the color extractor each rely on different sources of supervision. We therefore adopt a staged training procedure that decouples their objectives:

1. **Stage 1 — Geometry extractor.** The Transformer-based curve extractor is trained with $\mathcal{L}_{\text{geo}}$ (Eq. (3)) using ground-truth Bézier curves as supervision.

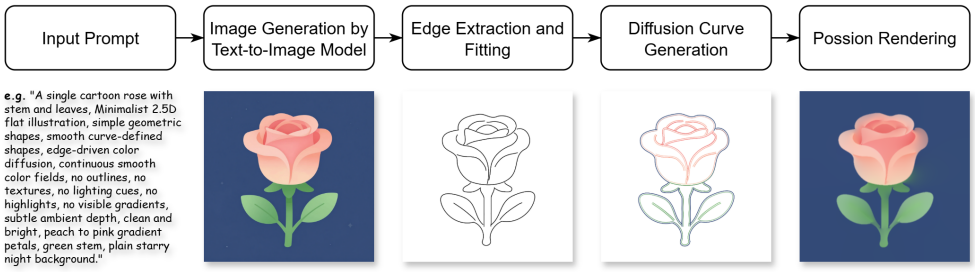


Fig. 4: Overview of the task-specific dataset construction pipeline. Starting from a structured text prompt, we generate a flat illustration, extract cartoon-specific curve structures, fit them with Bézier curves, construct the corresponding diffusion curves, and render the final image using a numerical Poisson solver. The pipeline provides aligned supervision across the original illustration, curve geometry, diffusion-curve representation, and rendered output.

2. **Stage 2 — FNO renderer.** The FNO is pre-trained with $\mathcal{L}_{\text{fno}}$ (Eq. (10)) on ground-truth diffusion-curve inputs to approximate the numerical Poisson solver. Once converged, its weights are frozen.
3. **Stage 3 — Color extractor.** With the frozen FNO providing a differentiable rendering surrogate, the color network is trained with $\mathcal{L}_{\text{color}}$ (Eq. (8)) directly against the input raster image, allowing gradients to flow from the rendered output back through the FNO to the color predictions.

This staged scheme ensures that each component learns its intended mapping under explicit supervision, avoiding the instability of joint end-to-end training while retaining a differentiable rendering path for color learning.

4 Experiments

We evaluate NDC primarily against traditional diffusion-curve vectorization baselines, focusing on reconstruction quality, runtime, compactness, and editability. These baselines share the same diffusion-curve formulation as NDC and are therefore directly comparable in terms of PDE-based rendering, curve count, curve length, and bilateral color controls. We additionally provide a supplementary reference to two recent SVG-based neural vectorization methods, LIVE [4] and O&R [31], to contvextualize the representational difference between SVG-path-based vectorization and our diffusion-curve formulation. We further ablate the matching strategy, geometry losses and color extraction design, validate the FNO renderer against numerical solvers, and demonstrate the editability of the extracted representations. Detailed implementation parameters are given in Sec. 4.2.

4.1 Task-specific Dataset Construction

Since existing datasets lack aligned supervision for Bézier curve geometry, diffusion-curve attributes, and rendered images, we construct a dedicated dataset tailored to our

framework. We first generate a large-scale set of flat-illustration images using the text-to-image model *Z-Image-Turbo* [41] with structured prompts based on ImageNet-style semantic categories (e.g., animals, plants, fruits, vehicles, buildings). The prompts explicitly encourage minimalist 2.5D flat illustrations with simple geometric composition, smooth boundaries, clean color regions, weak texture, and reduced lighting cues, characteristics well-suited for diffusion-curve representation.

For each generated image, we do not use the final vectorized outputs of the compared diffusion-curve baselines as supervision. Instead, we first extract cartoon-specific curve structures using the method of Cheng [42], which is designed to recover coherent cartoon curves by linking reliable curve points into structure curves and removing unreliable short or noisy responses. The recovered structure curves are then converted into parametric Bézier curves via inverse curve fitting, providing geometric supervision. Based on the fitted curves, we follow the standard diffusion-curve representation to assign two-sided color attributes and render the final image using a numerical Poisson solver. Consequently, each sample comprises: (i) the original illustration, (ii) the extracted Bézier curve geometry, (iii) the corresponding diffusion-curve attributes, and (iv) the rendered image, as illustrated in Fig. 4. The final dataset consists of 60,000 images with curve-geometry supervision, among which 10,000 samples are further annotated with diffusion-curve attributes and corresponding rendered outputs. All images are resized to 512×512 and split into training, validation, and test sets with a ratio of 8:1:1.

Compared with using the fragmented outputs of existing diffusion-curve pipelines as supervision, this construction provides more coherent structural traces that are better aligned with the three-segment G^1-continuous Bézier representation adopted in our framework.

4.2 Experimental Setup

Evaluation metrics. We evaluate the proposed method against traditional diffusion-curve baselines across three dimensions: reconstruction quality, measured by RMSE and LPIPS between the rendered output and the input image; runtime efficiency, measured as end-to-end wall-clock time from raster input to final rendered output on identical hardware; and representation compactness and editability, quantified by the number of curves, average curve length in pixels, and total number of explicit bilateral color control points per image. These metrics are appropriate for the main comparison because all compared methods in Tab. 1 produce diffusion-curve-like constraints or curve-based Poisson renderings. All quantitative results are evaluated on held-out test subsets using the best checkpoint selected on the validation set; Tab. 1 is averaged over a 1,000-sample test subset.

Baselines. We compare against three representative automatic diffusion-curve vectorization methods that span the design space of classical pipelines:

1. **Orzan et al. [1]**: a heuristic pipeline with multi-scale Canny edge detection, normal-direction color sampling, and Douglas-Peucker simplification, rendered via a multigrid Poisson solver.

2. **Xie et al.** [2]: a hierarchical approach that fits diffusion curves by operating in the Laplacian domain across multiple scales, capturing both coarse structures and fine details.
3. **Zhao et al.** [3]: an optimization-driven method that iteratively refines curve geometry and colors by formulating the problem as inverse shape optimization with FEM rendering.

Since these diffusion-curve baselines do not expose a unified parameter to control color control point budgets, we keep their original pipelines intact and report the resulting representation complexity explicitly. Minor hyperparameter tuning is performed on the validation set to better align with our dataset. Specifically, for Orzan et al. [1], we set Canny thresholds to 0.1/0.3, polyline tolerance to 1 pixel, minimum edge length to 5 pixels, DP color threshold to 16.0 (CIE Lab space), and normal sampling distance to 3 pixels. Rendering follows the original multigrid Poisson solver with Jacobi relaxation. For Xie et al. [2], we use $L = 4$ hierarchy levels, bilateral filter parameters $\sigma_{s0} = 2$, $\sigma_{r0} = 25.5$, endpoint linking distance 5 pixels, minimum chain length 10 pixels, width threshold 3, and voting ratio threshold 0.5. Non-maximum suppression applies fixed absolute thresholds of 0.01 (Laplacian) and 0.1 (bi-Laplacian). For Zhao et al. [3], we adopt 5 iso-levels, a maximum of 100 optimization iterations, convergence tolerance 10^{-3}, curve-length regularization weight 0.001, and FEM rendering with a 2-pixel color offset.

We additionally provide a supplementary cross-family reference against neural SVG vectorization methods (LIVE [4] and O&R [31]) at the end of this section. Since these methods output layered SVG paths rather than diffusion curves and are not directly comparable in primitive type or color model, we report them separately rather than including them in the main DC baseline table. Both baselines follow their authors' official configurations. LIVE uses the `experiment_32x1` setting, progressively adding 32 closed cubic Bézier paths one at a time with component-wise initialization, yielding 32 SVG paths per image. O&R uses the default schedule of $256 \to 128 \to 64$ paths over three stages of 100 iterations with the L1+CLIP loss (α_{L1}=0.95) and geometric regularizer; the canvas is set to 512×512 and all other hyperparameters follow the defaults, yielding 64 SVG paths per image after reduction.

Implementation details. The encoder is a ResNet-50-FPN backbone. The Transformer decoder uses $N_q = 80$ learnable queries and $L = 6$ decoder layers, each with 8-head self-attention, deformable cross-attention, and a two-layer FFN (GELU, hidden dim 1024). The color network comprises five residual Conv1D blocks (128 channels, kernel size 5, dropout 0.1). The FNO renderer has 4 Fourier layers with $m = n = 48$ modes and hidden width 96. Training uses AdamW (batch size 16). The geometry extractor and FNO are first trained for 30 epochs (lr 5×10^{-5} with cosine annealing to 5×10^{-6}); the color network is then trained for 30 additional epochs with the FNO frozen. All experiments are conducted on a single NVIDIA GeForce RTX 5090 GPU, and all reported runtimes use batch size 1 for fair comparison with non-batchable baselines. To support resolution generalization of the FNO renderer (Sec. 4.5), the FNO is trained with a randomized multi-resolution schedule: at each iteration, the input resolution is sampled uniformly at random from $\{256 \times 256, 512 \times 512, 1024 \times 1024\}$



(a) Cake (b) Hedgehog (c) Banana (d) Kitten

Fig. 5: Ground-truth flat-illustration images corresponding to the four examples shown in Fig. 6.

before constructing the constraint tensor X, so that the spectral weights see all three discretizations during training.

4.3 Main Results

We evaluated NDC on the 1,000-sample test set (Sec. 4.1) with minor baseline hyperparameter tuning on the validation set. Tab. 1 reports the main comparison with traditional diffusion-curve vectorization baselines at 512×512, the resolution at which most classical DC baselines were originally implemented. Color control point budgets vary across DC baselines as they are not directly controllable, so comparisons reflect each method's native trade-offs.

Within the diffusion-curve family, NDC offers two key advantages. First, it is substantially faster (0.106 s per image), roughly 18× faster than the fastest baseline of Orzan et al. [1] (1.91 s) and up to $\sim$ 900× faster than the slowest baseline of Xie et al. [2] (95.3 s) on identical hardware, supporting near-real-time per-image inference. Second, NDC yields more compact and practically editable representations. In terms of geometry, the extracted curves are fewer (16.9 vs. 69∼333) and substantially longer (172.3 px vs. 30.7∼68.5 px), meaning each curve carries more semantic content and is easier to select and manipulate individually. In terms of color, NDC exposes explicit bilateral color stops (K=5 per side, 82.0 control points per side on average) that can be modified directly without any re-optimization. By contrast, Xie et al. [2] solve color in the Laplacian domain via Green's functions, leaving no explicit per-curve color parameter to edit; Zhao et al. [3] embed color in a FEM triangulation rather than at the curve level, making direct curve-level color adjustment impractical. Orzan et al. [1] do expose bilateral color samples, but the dense, fragmented curve layout (265 control points per side) makes selective color editing tedious in practice.

On pixel-level RMSE, NDC reports 0.0384 against 0.0104∼0.0248 for the baselines, consistent with its emphasis on speed, compactness, and direct editability. Since RMSE measures only pixel-wise deviations and correlates weakly with perceptual quality, we further report LPIPS as a perceptual metric. NDC achieves LPIPS = 0.0493, below Orzan et al. (0.0615) and Xie et al. (0.0582), showing that the RMSE gap does not translate proportionally into perceptual degradation. Together with the qualitative results in Fig. 6, these numbers place NDC at a distinct operating point within



Fig. 6: Qualitative comparison with traditional diffusion-curve vectorization baselines. Each method shows the rendered result and the corresponding curve visualization. Text below each method reports RMSE, runtime, and structural statistics.

Table 1: Main comparison with traditional diffusion-curve vectorization baselines on the 1,000-sample test subset.

Method	RMSE ↓	LPIPS ↓	Time (s) ↓	#Curves ↓	Len. (px) ↑	Clr. Pts (L/R)[1]
Orzan et al. [1]	0.0248	0.0615	1.912	69.0	57.5	265.4 / 264.9
Xie et al. [2]	0.0147	0.0582	95.265	332.9	30.7	–
Zhao et al. [3]	**0.0104**	**0.0215**	48.184	132.6	68.5	–
Ours	0.0384	0.0493	**0.106**	**16.9**	**172.3**	**82.0 / 82.0**

[1]Reported only for methods with explicit bilateral color controls comparable to ours.

the diffusion-curve family, characterized by fast inference, compact representation, and editable curve-level control. The corresponding ground-truth images are shown in Fig. 5. We discuss content-adaptive curve budgeting as a future direction in Sec. 4.7.

As a supplementary cross-family reference, we additionally compare against two open-source neural SVG vectorization methods, LIVE [4] and O&R [31], on a smaller subset of N_{ext}=100 images using the authors' default hyperparameters. Since both

Table 2: Supplementary cross-family reference with recent SVG-based vectorization methods on a 100-sample subset.

Method	Representation (#shapes)	MSE[1] ↓	LPIPS ↓	Time (s) ↓	Clr. Pts (L/R)[2]
LIVE [4]	layered SVG paths (32)	**0.884**	0.1171	5196.392	–
O&R [31]	top-down SVG paths (64)	1.204	0.2045	333.447	–
Ours	diffusion curves (14.7)	1.308	**0.0376**	**0.110**	**73.6 / 73.6**

[1]MSE $\times 10^{-3}$, computed on RGB images normalized to $[0, 1]$.

[2]Reported only for methods with explicit bilateral color controls.

methods optimize SVG paths per image with pixel-wise rasterization losses (MSE in LIVE; L1 with α_{L1}=0.95 in O&R, plus a small CLIP term), their objectives explicitly favor pixel-level reconstruction, and this is reflected in their lower MSE values. NDC instead targets a different regime, learning curve geometry with a compact set of bilateral color controls and a PDE-based rendering model; as a result, it achieves substantially lower LPIPS (0.0376 vs. 0.117 and 0.204), indicating more globally coherent appearance, while maintaining competitive pixel-level accuracy without per-image fitting. In addition, because LIVE and O&R rely on iterative per-image optimization over SVG primitives, their inference time scales with the number of paths and optimization steps, whereas NDC performs a single forward pass with a learned renderer, leading to significantly lower runtime.

The structural difference between the two representations further limits direct comparison. LIVE and O&R fill region interiors with per-path solid or simple-gradient colors and rely on path count to approximate smooth shading, while NDC defines color discontinuities along curves and recovers smooth interiors via the Poisson equation, so its compactness comes from the PDE rather than per-shape fills. Consistently, LIVE and O&R use 32 and 64 SVG paths respectively, whereas NDC uses on average 14.7 diffusion curves on the same images, each carrying bilateral color stops diffused globally by the renderer. Since both methods output layered SVG paths rather than bilateral DC constraints, DC-specific statistics (curve length, color control points) are not directly comparable; we therefore report only image-domain metrics and runtime in Tab. 2, and treat this comparison as contextual rather than like-for-like.

4.4 Ablation studies

We conducted ablation studies to isolate and quantify the contribution of each key component in our framework. All ablation experiments use the same training settings and rendering configuration as the full model, and are evaluated on the corresponding held-out test subsets. For curve extraction, we report Chamfer Distance (CD), RMSE, and LPIPS; for color extraction, we report RMSE, compactness, and runtime.

Sensitivity to query budget. We analyze the sensitivity of NDC to the query budget N_q, which controls the maximum number of candidate curves predicted by the Transformer decoder. Tab. 3 reports the results.

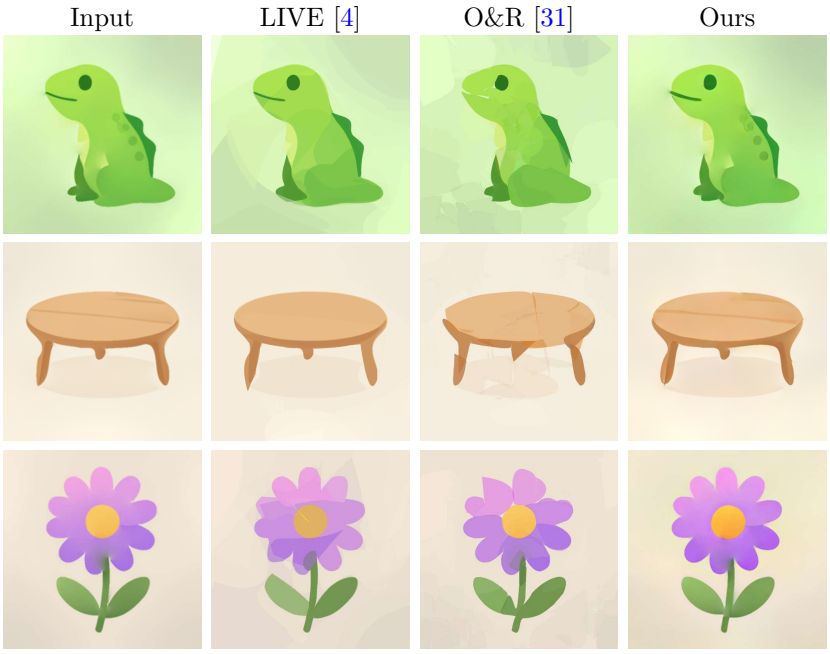


Fig. 7: Qualitative comparison with neural SVG vectorization methods on the extended subset. LIVE and O&R produce layered SVG paths, whereas NDC produces diffusion curves with bilateral color constraints and PDE-based smooth shading.

The default value $N_q = 80$ provides a good balance between curve coverage and prediction compactness. Reducing the budget to $N_q = 40$ limits the number of available candidates and weakens coverage for long or complex structures, while increasing it to $N_q = 120$ introduces more unmatched queries and makes the classification problem more imbalanced. Both deviations lead to mild degradation, indicating that $N_q = 80$ is an effective trade-off for our dataset.

Matching strategy. Query-based curve prediction requires assigning predicted curves to ground-truth curves during training. We compared two assignment strategies: standard Hungarian matching (a fixed one-to-one bipartite matching) and dynamic SimOTA matching [8], which adaptively selects multiple positive predictions per ground truth based on cost thresholds. As shown in Tab. 3, SimOTA significantly outperforms Hungarian matching, reducing CD from 0.0154 to 0.0047 and RMSE from 0.0693 to 0.0419. This improvement stems from SimOTA's ability to provide richer supervision, especially for long or complex curves, by allowing multiple queries to learn from the same ground truth, thereby improving recall and geometric fidelity. The dynamic nature of SimOTA also stabilizes training by avoiding the hard thresholding of fixed matching.

Geometry supervision. Our geometry loss combines three complementary terms: coordinate regression ($\mathcal{L}_{\text{coord}}$), shape consistency ($\mathcal{L}_{\text{shape}}$), and continuity

Table 3: Ablation on the curve extraction module, including query budget, matching strategy, and geometry-loss components, evaluated on the corresponding test subset.

Group	Setting	CD ↓	RMSE ↓	LPIPS ↓
Query budget	$N_q = 40$	0.0058	0.0461	0.0537
	$N_q = 80$[1]	**0.0047**	**0.0419**	**0.0502**
	$N_q = 120$	0.0061	0.0473	0.0552
Matching strategy	Hungarian	0.0154	0.0693	0.0823
	SimOTA[1]	**0.0047**	**0.0419**	**0.0502**
Geometry supervision	w/o $\mathcal{L}_{\text{shape}}$	0.0129	0.0655	0.0791
	w/o $\mathcal{L}_{\text{cont}}$	0.0073	0.0427	0.0556
	Full model[1]	**0.0047**	**0.0419**	**0.0502**

[1]Default configuration.

($\mathcal{L}_{\text{cont}}$). To assess their individual importance, we train models with each term removed while keeping all other settings identical. Tab. 3 summarizes the results.

Removing $\mathcal{L}_{\text{shape}}$ causes the most severe degradation (CD increases to 0.0129, RMSE to 0.0655), confirming that point-wise control point regression alone is insufficient to capture global curve geometry. The shape loss, which samples points along the curve and penalizes deviations, is essential for accurate fitting, particularly in high-curvature regions. Note that removing $\mathcal{L}_{\text{cont}}$ also harms performance (CD 0.0073, RMSE 0.0427), indicating that enforcing tangent continuity across segment junctions improves structural coherence and prevents kinks. The full model achieves the best results, demonstrating that all three terms play synergistic roles: coordinate regression provides precise localization, shape loss ensures global fidelity, and continuity loss promotes smooth, visually pleasing junctions.

Color extraction design. We next isolated the contribution of our color extraction module by fixing the predicted curve geometry and using the same frozen FNO renderer across all variants. This controlled setup allows us to assess how different color parameterization strategies affect reconstruction quality, compactness, and the runtime of the color extraction stage itself. We compared three approaches:

- **BS+DP**: A traditional heuristic baseline that performs bilateral sampling along curve normals, followed by DP sparsification to reduce the number of color control points. This mimics the color extraction step in classical diffusion curve pipelines [1] and serves as a non-learned reference.
- **NP+U**: Our learnable normal-profile feature network, but with a simplified color output: instead of learning control point positions, we uniformly placed $K = 5$ control points per side along the curve parameter t and predict their colors directly. This variant isolates the benefit of learned sparsification.
- **NP+LS** (ours): The full color extraction model, which uses the same normal-profile features but regresses continuous interior control point positions $t_k \in (0, 1)$ per side, enabling adaptive, non-uniform placement of the $K = 5$ control points.

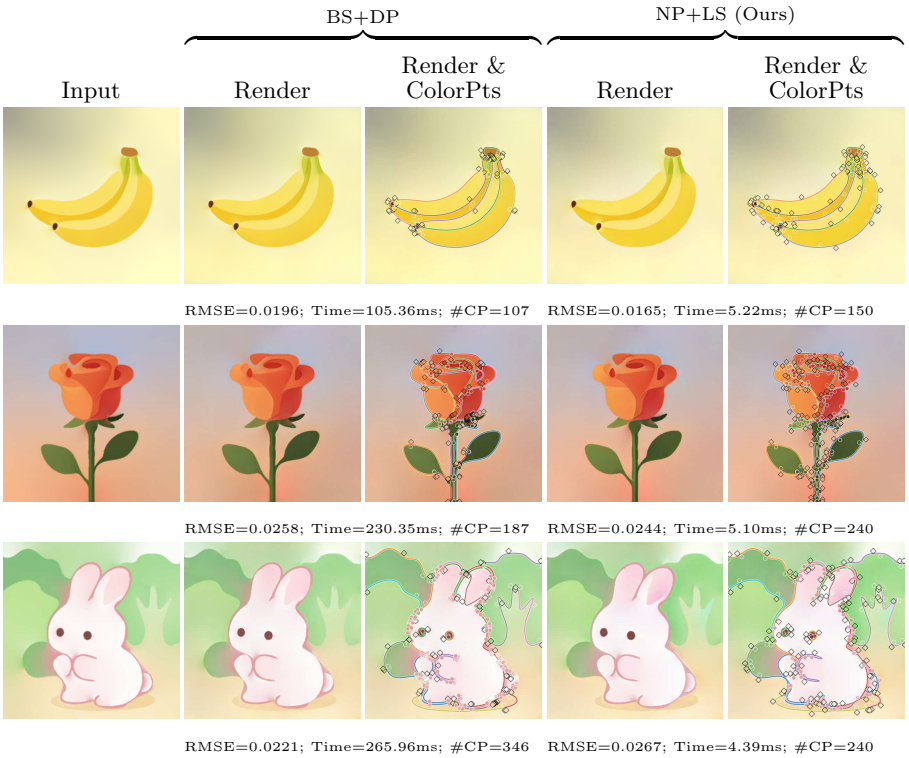


Fig. 8: Qualitative comparison of color extraction strategies. Text below each example reports RMSE, runtime (image-to-controls, excluding rendering), and the total number of color control points (#CP) per image.

Tab. 4 summarizes image-domain RMSE, total color control points per image (#ColorPts), and color-extraction runtime (excluding rendering). BS+DP achieves the lowest RMSE (0.0232) with moderate control points (146), but is substantially slower (205 ms) due to dense sampling and iterative DP. Among learning-based variants, NP+LS slightly outperforms NP+U (0.0269 vs. 0.0271 RMSE) with the same #ColorPts (184) and negligible runtime overhead (4.46 ms vs. 4.32 ms). Both learnable methods are about 45× faster than BS+DP, demonstrating the efficiency of our feedforward design.

Qualitatively (Fig. 8), BS+DP preserves sharp boundaries but oversimplifies subtle gradients; NP+U produces smoother results yet may miss fine details due to fixed control-point spacing; NP+LS strikes a balance, accurately reconstructing both sharp transitions and smooth shading while maintaining compactness. These results validate that learning both color values and their adaptive positions yields an efficient feed-forward color extractor that matches the heuristic baseline's compactness while running substantially faster.

Table 4: Ablation on color extraction design. Curve geometry is fixed and the FNO renderer is frozen across all variants. Time measures color extraction only.

Setting	RMSE ↓	#ColorPts	Time (ms) ↓
BS+DP	**0.0232**	146	205.45
NP+U	0.0271	184	**4.32**
NP+LS (ours)	0.0269	184	4.46

Table 5: Quantitative evaluation of renderers at 512×512. FD is the multigrid Poisson solver run to convergence and serves as the reference. All metrics are averaged over the test set; LPIPS is computed against the FD reference.

Method	RMSE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time (ms) ↓	Speedup ↑[1]
FD (reference)	—	—	—	—	803.94	—
FEM	**0.0108**	**39.93**	0.9916	0.0357	1041.82	0.77×
FNO (ours)	0.0172	36.07	**0.9924**	**0.0141**	**75.24**	**10.69**×

[1]Speedup $= T_{\mathrm{FD}}/T_{\mathrm{method}}$, where $T_{\mathrm{FD}} = 803.94\,\mathrm{ms}$.

4.5 Validation of the FNO Renderer

We validated the proposed FNO as a differentiable surrogate for diffusion curve rendering. An ideal renderer must satisfy three criteria: (i) high fidelity to the true Poisson solution, (ii) computational efficiency, and (iii) differentiability for gradient-based learning. Traditional numerical solvers satisfy the first criterion but fail on the latter two. We therefore evaluate the FNO from three aspects: renderer accuracy and efficiency, the effect of physics-informed constraints, and multi-resolution generalization.

Renderer accuracy and efficiency. We quantitatively and qualitatively compare the FNO against two classical solvers: a high-quality finite-difference (FD) implementation and a finite-element method (FEM) baseline.

Our FD solver follows the original diffusion curve pipeline [1]: color sources and gradient constraints are rasterized, and the Poisson equation is solved using a GPU-accelerated V-cycle multigrid scheme with Jacobi relaxations (5 iterations per level on a multigrid hierarchy down to 4×4), run to convergence to serve as the reference solution. The FEM solver is a fully reproduced baseline that discretizes the domain with linear elements. Both solvers produce high-quality reference images, but are non-differentiable and computationally intensive.

We rendered the full held-out test set of diffusion-curve inputs using FD, FEM, and FNO at 512×512. Let I_{FD}, I_{FEM}, and I_{FNO} denote the output RGB images. Treating I_{FD} as the reference, we measured RMSE, PSNR, SSIM, and LPIPS. Runtime is measured under identical hardware conditions with batch size 1. Each metric is computed per image and then averaged.



(a) FD reference (b) FEM result (c) FNO result (d) Error map

Fig. 9: Visual comparison of FD, FEM, and the proposed FNO renderer, with a per-pixel error map (FNO vs. FD).

Table 6: Ablation on the physics-informed objective for the FNO renderer.

Setting	RMSE ↓	PSNR ↑	SSIM ↑	LPIPS ↓
FNO-Data	0.0175	35.89	0.9913	0.0153
Full model	**0.0172**	**36.07**	**0.9924**	**0.0141**

Tab. 5 summarizes the results. Both FEM and FNO achieve high fidelity to the FD reference, but the two methods exhibit complementary error structures. FEM obtains a lower RMSE (0.0108 vs. 0.0172) and a higher PSNR (39.93 vs. 36.07), reflecting smaller pixel-wise deviations on average. FNO, in contrast, attains a higher SSIM (0.9924 vs. 0.9916) and a substantially lower LPIPS (0.0141 vs. 0.0357), indicating better structural and perceptual similarity to the FD reference. This complementary behavior suggests that the two approximations exhibit different error distributions: the FNO's residuals are mainly concentrated near high-frequency curve boundaries (Fig. 9), which affects RMSE more than perceptual similarity. The decisive advantage of the FNO is its speed: it renders an image in only 75.24 ms, 10.69× faster than FD (803.94 ms) and 13.85× faster than FEM (1041.82 ms). This acceleration, together with full differentiability, makes the FNO a practical surrogate for the Poisson solver in a learning-based pipeline.

As shown in Fig. 9, the FNO output is visually close to the FD reference, with residual errors concentrated near high-frequency curve boundaries while large smooth interior regions exhibit near-zero error. This confirms that the FNO approximates the global diffusion behavior of diffusion curves with only small residual errors. We therefore freeze the pre-trained FNO during color network training, trading a small amount of approximation error for substantial gains in efficiency and differentiability.

Physics-informed constraints. To verify the benefit of incorporating the boundary condition and PDE residual terms into the training objective, we compared the full model against a data-only baseline (FNO-Data) that retains only $\mathcal{L}_{\text{data}}$ and removes both $\mathcal{L}_{\text{bc}}$ and $\mathcal{L}_{\text{pde}}$, with all other architecture and training settings kept identical.

As shown in Tab. 6, incorporating the boundary condition loss $\mathcal{L}_{\text{bc}}$ and the PDE residual term $\mathcal{L}_{\text{pde}}$ consistently improves all four metrics. The practical benefit of

Table 7: Multi-resolution generalization of the FNO renderer. Resolutions marked with * are unseen during training.

Resolution	RMSE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	Time (ms) ↓	Speedup ↑
256×256	0.0164	36.60	0.9891	0.0104	67.12	6.29×
$384 \times 384^*$	0.0172	36.11	0.9913	0.0131	70.43	8.90×
512×512	0.0172	36.07	0.9924	0.0141	75.24	10.69×
$768 \times 768^*$	0.0186	35.40	0.9936	0.0160	89.55	13.36×
1024×1024	0.0198	34.83	0.9931	0.0173	120.11	12.92×
$1280 \times 1280^*$	0.0232	33.28	0.9937	0.0181	149.62	12.75×

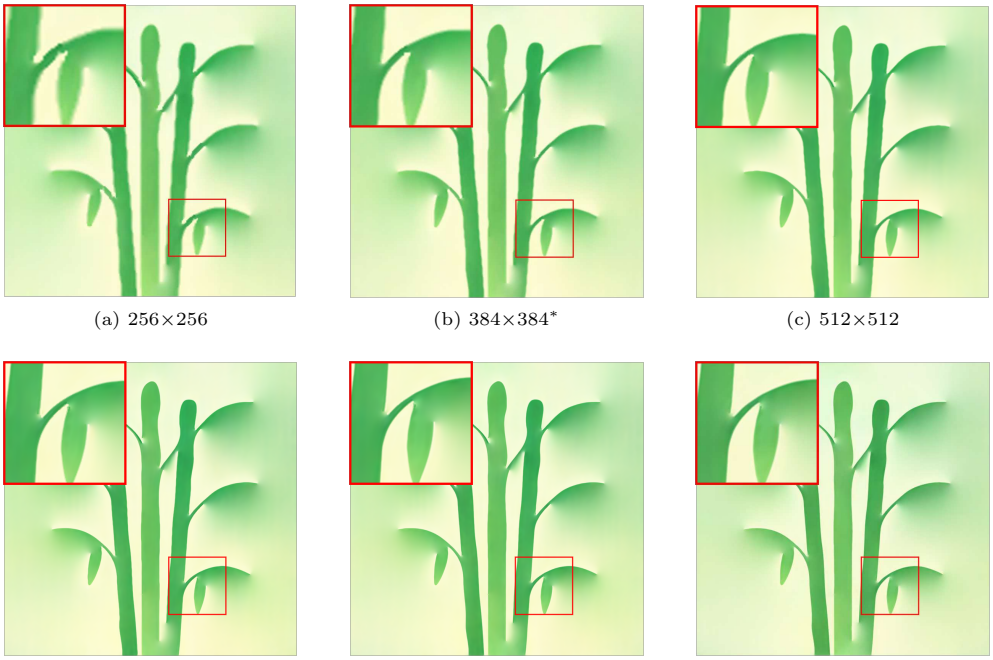
the physics-informed terms lies in suppressing color drift at curve boundaries ($\mathcal{L}_{\text{bc}}$) and maintaining smooth diffusion in interior regions ($\mathcal{L}_{\text{pde}}$). These results confirm that physics-informed supervision improves rendering consistency beyond what data supervision alone can achieve.

Multi-resolution generalization. Beyond accuracy at a fixed resolution, an important property of the FNO is its resolution generalization: neural operators parameterize integral operators in function space, theoretically decoupling the learned weights from the discrete grid resolution and enabling generalization to unseen resolutions. To empirically verify this property, we evaluated the FNO renderer (trained with a multi-resolution strategy spanning 256×256, 512×512, and 1024×1024) on six resolutions from 256×256 to 1280×1280. Resolutions not seen during training are marked with * in Tab. 7. FD at each resolution serves as the reference baseline, and we additionally report the speedup of FNO over FD at each scale.

Tab. 7 shows that the FNO renderer maintains consistent rendering quality across all six resolutions, including the three unseen settings (384, 768, and 1280): RMSE grows mildly from 0.0164 to 0.0232 as resolution increases, while SSIM stays above 0.989 throughout, confirming that the learned spectral weights generalize to discretizations beyond those seen in training. In terms of efficiency, FNO inference time grows from 67.12 ms at 256×256 to 149.62 ms at 1280×1280 (about 2.2×), while FD's runtime grows much faster over the same range. As a result, the speedup ratio increases from 6.29× at 256×256 to a peak of 13.36× at 768×768, and remains above 12× at 1024×1024 and 1280×1280. Overall, the efficiency advantage of the FNO becomes increasingly pronounced as resolution grows, precisely where traditional Poisson solvers are most costly. A qualitative example is shown in Fig. 10. Across both training and unseen resolutions, the main color diffusion patterns remain stable, consistent with the fixed-resolution error analysis in Fig. 9.

4.6 Editability Demonstration

Beyond quantitative evaluation, we demonstrate the editability of our representation through a simple manual editing example. Each output curve is stored as ten 2D Bézier control points together with K=5 bilateral color stops $\{(t_k, r_k, g_k, b_k)\}$ per side, and we provide a lightweight editor in which the user clicks on a curve to select it, drags any control point to update its geometry, or changes a color stop through a



(a) 256×256 (b) 384×384* (c) 512×512

(d) 768×768* (e) 1024×1024 (f) 1280×1280*

Fig. 10: Visual quality of the FNO renderer across six resolutions, including three resolutions unseen during training (marked with *). The same diffusion-curve input is rendered at each resolution; sharp curve boundaries and smooth interior shading are preserved consistently across all scales, providing a qualitative confirmation of the resolution-generalization behaviour reported in Tab. 7.

standard color picker. Fig. 11 shows two sequential edits on the extracted curves: we first adjust the Bézier control points of the flower stem to change its shape, and then modify the bilateral color stops of the petal curves to change their color. After each edit, the updated parameter set is fed once through the frozen FNO renderer, with no re-optimization or re-extraction.

Both geometry and color can be edited directly through the stored curve parameters, and the updated representation is re-rendered by a single FNO forward pass without re-optimization or re-extraction. Such curve-level access is a property of the output representation itself: the fragmented curves of Orzan et al. [1] couple a single semantic structure across multiple short primitives, while Xie et al. [2] and Zhao et al. [3] encode color in the Laplacian domain or a FEM triangulation rather than at the curve level, leaving no per-curve color parameter directly accessible to the user. NDC is therefore well suited for downstream manual adjustment and interactive editing.



(a) Ground truth (b) Geometry edit (c) Color edit

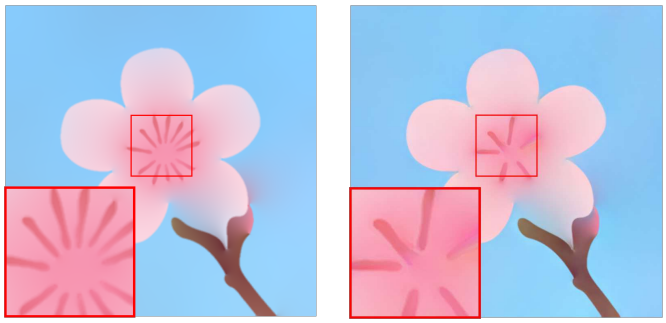
Fig. 11: Editability demonstration of the proposed representation. Starting from the extracted diffusion curves, a geometry edit adjusts the flower stem shape (b), followed by a color edit that modifies the petal color (c). Both operations are applied directly to curve parameters and re-rendered via a single FNO forward pass.

4.7 Limitations

NDC builds on the diffusion-curve formulation, which is inherently designed for flat illustrations and piecewise-smooth content with clear region boundaries rather than for natural photographs with dense small-scale textures [1, 11]. Within this intended scope, NDC has two main limitations.

A first limitation comes from the fixed representation budget. The current model uses a fixed upper budget of N_q=80 queries and represents each curve as a three-segment cubic Bézier. This design keeps the output compact and works well for most images in our dataset, but it is not optimal for all cases. Images with densely packed boundaries, thin structures, or highly curved local geometry may require more curves or more segments per curve than the fixed budget allows. When adjacent contours lie very close to each other, the geometry extractor may merge nearby boundaries, miss small structures, or produce locally misaligned predictions, which further affects bilateral color prediction and leads to local rendering artifacts. A representative example is shown in Fig. 12.

Secondly, the FNO renderer is a learned surrogate of a numerical Poisson solver. Although the quantitative and multi-resolution experiments in Sec. 4.5 show that it approximates the FD reference well across several resolutions, it still introduces small approximation errors, especially near high-frequency curve boundaries. For applications that require exact numerical solutions or strict physical accuracy, a conventional Poisson solver may still be preferable at the cost of speed and differentiability. Future work will explore content-adaptive query allocation, variable per-curve segment counts, stronger boundary-sensitive geometry modeling, and hybrid representations that combine diffusion curves with texture residuals or gradient-mesh interiors for more complex images.



(a) Ground truth (b) Failure case

Fig. 12: Representative limitation of NDC on locally complex geometry. The flower center (inset) contains densely packed radial boundaries; the geometry extractor fails to separate the closely spaced contours, leading to merged or missing curves that propagate errors to the color prediction stage and produce visible rendering artifacts in the output.

5 Conclusion and Future Work

We have presented Neural Diffusion Curves (NDC), a differentiable framework that unifies Transformer-based curve geometry prediction, learnable bilateral color extraction, and FNO-based differentiable rendering for automatic diffusion-curve vectorization. Extensive experiments on a constructed dataset of 60,000 flat-illustration images demonstrate that NDC processes a 512×512 image in 0.106 s, achieving an $\sim 18\times$ speedup over the fastest classical diffusion-curve pipeline and a nearly $900\times$ speedup over the slowest. Beyond runtime, NDC reduces the curve count by over 75% and produces curves that are 2.5–5.6× longer on average, yielding more coherent and practically editable structures. On perceptual LPIPS, NDC outperforms the heuristic baselines of Orzan et al. and Xie et al., confirming that the gains in speed and compactness do not come at the expense of perceived visual quality. The compact, explicitly parameterized representation further enables direct editing of both curve geometry and bilateral colors without re-optimization, a capability that is generally unavailable in neural implicit methods and practically limited in classical pipelines.

The limitations discussed in Section 4.7 point to several directions for future work: content-adaptive query allocation, variable per-curve segment counts, stronger boundary-sensitive geometry modeling, and hybrid representations that combine diffusion curves with texture residuals or gradient-mesh interiors to extend NDC beyond flat illustrations.

Declarations

Funding

The research was supported by the Shaoxing Municipal Science and Technology Program (Grant No. 2025A11014) and the Key Research and Development Program of Zhejiang Province (Grant No. 2024C01210).

Competing interests

The authors have no competing interests to declare that are relevant to the content of this article.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Data availability

The dataset generated in this study, comprising 60,000 flat-illustration images with aligned Bézier curve geometry and 10,000 samples additionally annotated with diffusion-curve attributes and rendered outputs, is not publicly available at the time of submission but will be made publicly available after publication. Before public release, the data are available from the corresponding author upon reasonable request.

Materials availability

Not applicable.

Code availability

The source code of NDC, including the Transformer-based curve extractor, the color extraction network, and the FNO-based differentiable renderer, will be released publicly upon acceptance of this paper.

Author contributions

Y. Huang: Methodology, Software, Validation, Formal analysis, Investigation, Writing – Original Draft, Writing – Review & Editing. W. Yang: Methodology, Software, Validation, Investigation. H. Xie: Formal analysis, Writing – Review & Editing. Y.-K. Lai: Formal analysis, Writing – Review & Editing. Y. Jin: Supervision, Conceptualization, Methodology, Formal analysis, Writing – Review & Editing. H. Zhang: Supervision, Writing – Review & Editing. L. He: Supervision, Writing – Review & Editing. All authors reviewed and approved the final manuscript.

References

[1] Orzan, A., Bousseau, A., Winnemöller, H., Barla, P., Thollot, J., Salesin, D.: Diffusion curves: a vector representation for smooth-shaded images. ACM Transactions on Graphics (ToG) **27**(3), 1–8 (2008)

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

[3] Zhao, S., Durand, F., Zheng, C.: Inverse diffusion curves using shape optimization. IEEE transactions on visualization and computer graphics **24**(7), 2153–2166 (2017)

[4] Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y., Shi, H.: Towards layer-wise image vectorization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16314–16323 (2022)

[5] Polaczek, S., Alaluf, Y., Richardson, E., Vinker, Y., Cohen-Or, D.: Neuralsvg: An implicit representation for text-to-vector generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 15458–15468 (2025)

[6] Thamizharasan, V., Liu, D., Fisher, M., Zhao, N., Kalogerakis, E., Lukac, M.: Nivel: Neural implicit vector layers for text-to-vector generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4589–4597 (2024)

[7] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., Anandkumar, A.: Fourier neural operator for parametric partial differential equations (2021)

[8] Ge, Z., Liu, S., Li, Z., Yoshie, O., Sun, J.: Ota: Optimal transport assignment for object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 303–312 (2021)

[9] Selinger, P.: Potrace: a polygon-based tracing algorithm (2003)

[10] Chang, H.-H., Yan, H.: Vectorization of hand-drawn image using piecewise cubic bezier curves fitting. Pattern recognition **31**(11), 1747–1755 (1998)

[11] Tian, X., Günther, T.: A survey of smooth vector graphics: Recent advances in repr esentation, creation, rasterization, and image vectorization. IEEE Transactions on Visualization and Computer Graphics **30**(3), 1652–1671 (2022)

[12] Dziuba, M., Jarsky, I., Efimova, V., Filchenkov, A.: Image vectorization: a review. Journal of Mathematical Sciences **285**(2), 155–168 (2024)

[13] Lin, H., Zhang, J., Xu, C.: Diffusion curves with diffusion coefficients. Computational Visual Media **4**(2), 149–160 (2018)

[14] Hou, F., Sun, Q., Fang, Z., Liu, Y.-J., Hu, S.-M., Qin, H., Hao, A., He, Y.: Poisson vector graphics (pvg). IEEE transactions on visualization and computer graphics **26**(2), 1361–1371 (2018)

[15] Bang, S., Serkh, K., Stein, O., Jacobson, A.: An adaptive fast-multipole-accelerated hybrid boundary integral equation method for accurate diffusion curves. ACM Transactions on Graphics (TOG) **42**(6), 1–28 (2023)

[16] Sun, X., Xie, G., Dong, Y., Lin, S., Xu, W., Wang, W., Tong, X., Guo, B.: Diffusion curve textures for resolution independent texture mapping. ACM Transactions on Graphics (TOG) **31**(4), 1–9 (2012)

[17] Jeschke, S., Cline, D., Wonka, P.: A gpu laplacian solver for diffusion curves and poisson image editing. In: ACM SIGGRAPH Asia 2009 Papers, pp. 1–8 (2009)

[18] Li, Y., Wang, C., Hong, J., Zhu, J., Guo, J., Wang, J., Guo, Y., Wang, W.: Video vectorization via bipartite diffusion curves propagation and optimization. IEEE Transactions on Visualization and Computer Graphics **28**(9), 3265–3276 (2021)

[19] Sun, J., Liang, L., Wen, F., Shum, H.-Y.: Image vectorization using optimized gradient meshes. ACM Transactions on Graphics (TOG) **26**(3), 11 (2007)

[20] Lai, Y.-K., Hu, S.-M., Martin, R.R.: Automatic and topology-preserving gradient mesh generation for image vectorization. ACM Transactions on Graphics (TOG) **28**(3), 1–8 (2009)

[21] Zhou, J., Hettinga, G., Houwink, S., Kosinka, J.: Feature-adaptive and hierarchical subdivision gradient meshes. In: Computer Graphics Forum, vol. 41 (2022)

[22] Zhu, H., Cao, J., Xiao, Y., Chen, Z., Zhong, Z., Zhang, Y.J.: Tcb-spline-based image vectorization. ACM Transactions on Graphics (TOG) **41**(3), 34–13417 (2022)

[23] Lawonn, K., Günther, T.: Stylized image triangulation. Computer Graphics Forum **38**(1), 221–234 (2019)

[24] Xiao, Y., Cao, J., Chen, Z.: Image representation on curved optimal triangulation. Computer Graphics Forum **41**(6), 23–36 (2022)

[25] Fang, Z.-D., Guo, J.-P., Xiao, Y., Fu, X.-M.: Error-bounded image triangulation. Computer Graphics Forum **42**(7), 14967 (2023)

[26] Wang, W., Chen, Z., Fang, L., Cao, J.: Curved image triangulation based on differentiable rendering. Computer Graphics Forum **43**(7), 15232 (2024)

[27] He, K., Roerdink, J.B.T.M., Kosinka, J.: Monte Carlo optimization for gradient meshes. Graphical Models **143**, 101320 (2026)

[28] Li, T.-M., Lukáč, M., Gharbi, M., Ragan-Kelley, J.: Differentiable vector graphics rasterization for editing and learning. ACM Transactions on Graphics (TOG) **39**(6), 1–15 (2020)

[29] Wu, R., Su, W., Liao, J.: Layerpeeler: Autoregressive peeling for layer-wise image vectorization. In: Proceedings of the SIGGRAPH Asia 2025 Conference Papers, pp. 1–20 (2025)

[30] Law, H., Kang, S.H.: Image vectorization with depth: convexified shape layers with depth ordering. SIAM Journal on Imaging Sciences **18**(2), 963–1001 (2025)

[31] Hirschorn, O., Jevnisek, A., Avidan, S.: Optimize & reduce: A top-down approach for image vectorization. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 2148–2156 (2024)

[32] Zhao, K., Bao, L., Li, Y., Su, X., Zhang, K., Qiao, X.: Less is more: Efficient image vectorization with adaptive parameterization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 18166–18175 (2025)

[33] Chakraborty, S., Batra, V., Phogat, A., Jain, V., Ranawat, J.S., Dhingra, S., Wampler, K., Fisher, M., Lukáč, M.: Image vectorization via gradient reconstruction. In: Computer Graphics Forum, vol. 44, p. 70055 (2025). Wiley Online Library

[34] Du, Z., Kang, L., Tan, J., Gingold, Y.I., Xu, K.: Image vectorization and editing via linear gradient layer decomposition. ACM Transactions on Graphics (TOG) **42**(4), 97–19713 (2023)

[35] Chen, Y., Ni, B., Liu, J., Huang, X., Chen, X.: Towards high-fidelity artistic image vectorization via texture-encapsulated shape parameterization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15877–15886 (2024)

[36] Song, Y., Wang, J., Wei, L.-Y., Wang, W.: Vector regression functions for texture compression. ACM Transactions on Graphics (TOG) **35**(1), 1–10 (2015)

[37] Jarsky, I., Efimova, V., Chebykin, A., Shalamov, V., Filchenkov, A.: Neural style transfer for vector graphics. In: VISIGRAPP (3): VISAPP, pp. 686–693 (2024)

[38] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection (2021)

[39] Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object

detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)

[40] Li, Z., Huang, D.Z., Liu, B., Anandkumar, A.: Fourier neural operator with learned deformations for pdes on general geometries. Journal of Machine Learning Research **24**(388), 1–26 (2023)

[41] Cai, H., Cao, S., Du, R., Gao, P., Hoi, S., Hou, Z., Huang, S., Jiang, D., Jin, X., Li, L., et al.: Z-image: An efficient image generation foundation model with single-stream diffusion transformer. arXiv preprint arXiv:2511.22699 (2025)

[42] Cheng, M.-M.: Curve structure extraction for cartoon images. In: Proceedings of the 5th Joint Conference on Harmonious Human Machine Environment, pp. 13–25 (2009)