2D Neural Fields with Learned Discontinuities

CHENXI LIU, University of Toronto, Canada
SIQI WANG, New York University, Courant Institute of Mathematical Sciences, USA
MATTHEW FISHER, Adobe Research, USA
DEEPALI ANEJA, Adobe Research, USA
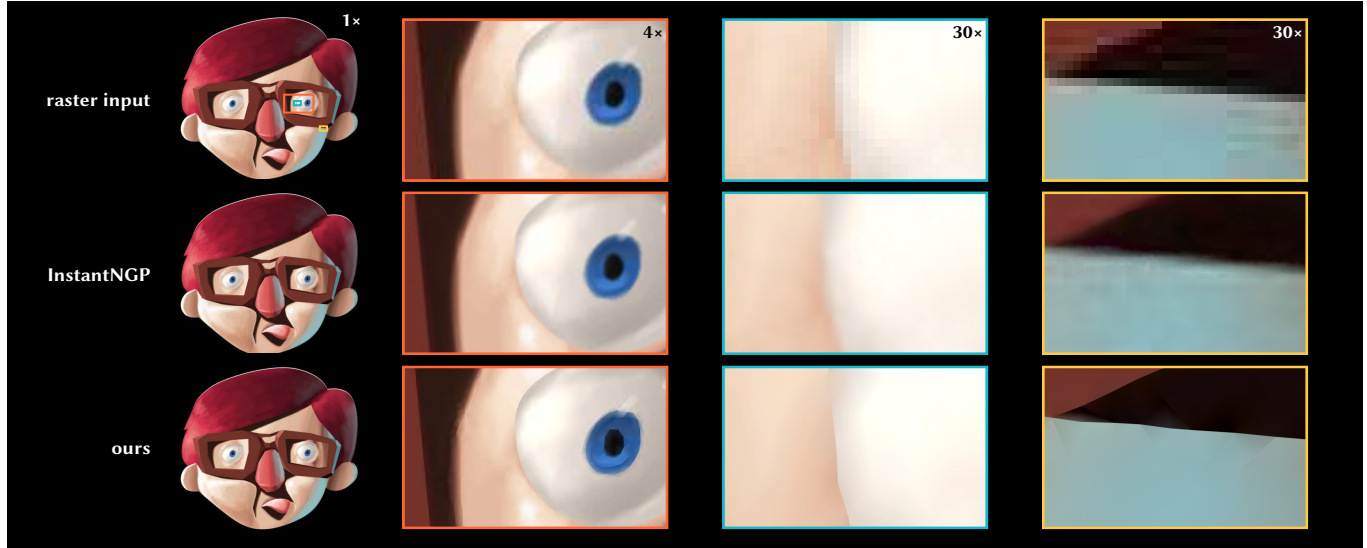ALEC JACOBSON, University of Toronto, Canada and Adobe Research, Canada

Fig. 1. We propose a discontinuous 2D neural field that can jointly approximate the target image and recover unknown discontinuities. Our neural field supports edge-preserving denoising and arbitrary resolution while continuous neural field, such as InstantNGP [Müller et al. 2022], blurs region boundaries under close view. Zoom scales indicate inference scale. Zoom in to see details.

Effective representation of 2D images is fundamental in digital image processing, where traditional methods like raster and vector graphics struggle with sharpness and textural complexity respectively. Current neural fields offer high-fidelity and resolution independence but require predefined meshes with known discontinuities, restricting their utility. We observe that by treating all mesh edges as potential discontinuities, we can represent the magnitude of discontinuities with continuous variables and optimize. Based on this observation, we introduce a novel discontinuous neural field model that jointly approximate the target image and recovers discontinuities. Through systematic evaluations, our neural field demonstrates superior performance in denoising and super-resolution tasks compared to InstantNGP, achieving improvements of over 5dB and 10dB, respectively. Our model also outperforms Mumford-Shah-based methods in accurately capturing discontinuities, with Chamfer distances 3.5× closer to the ground truth. Additionally, our approach shows remarkable capability in approximating complex artistic and natural images and cleaning up diffusion-generated depth maps.

CCS Concepts: • **Computing methodologies → Image representations**; **Reconstruction**; **Neural networks**.

Authors' addresses: Chenxi Liu, University of Toronto, Toronto, Canada, chenxil@cs.toronto.edu; Siqi Wang, New York University, Courant Institute of Mathematical Sciences, New York, USA, siqi.wang@nyu.edu; Matthew Fisher, Adobe Research, San Francisco, USA, matfishe@adobe.com; Deepali Aneja, Adobe Research, Seattle, USA, aneja@adobe.com; Alec Jacobson, University of Toronto, Toronto, Canada, jacobson@cs.toronto.edu and Adobe Research, Toronto, Canada, alecjacobson@adobe.com.

Additional Key Words and Phrases: Discontinuity, neural fields, denoising, super-resolution, vectorization

1 INTRODUCTION

Digital image representations — such as pixel arrays or vector graphics — discretize *image functions* that map 2D locations to colors. For a variety of reasons (occlusions in captured 3D scenes, layers in graphic designs, material boundaries, etc.), typical image functions are well modeled as continuous functions *almost everywhere* with sparse discontinuities appearing along 1D curves. Unfortunately, images stored as regular grids of pixel colors do not directly model discontinuities and assume a fixed resolution. Meanwhile, vector graphics formats (e.g., `.svg`) represent resolution-independent discontinuities directly with curves, fill boundaries, or layer overlaps, but these formats have minimal support for continuous signals elsewhere (e.g., solid colors, basic gradients).

Recently, Belhe et al. [2023] proposed storing images as the output of a small neural network fed a spatially varying feature vector. That feature vector is carefully interpolated over a triangle mesh constructed to ensure discontinuities along certain edges. The weights of the neural network are optimized to reconstruct samples of an image function. This exciting representation is very compact and
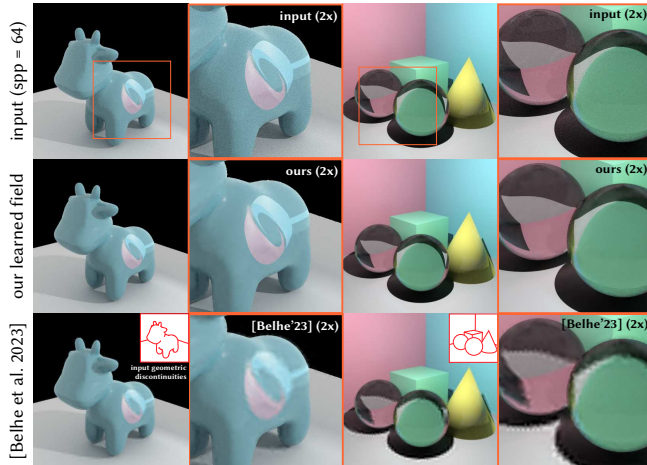
Fig. 2. Discontinuity-aware 2D neural field [Belhe et al. 2023] requires accurate 2D discontinuities as input. In their example of denoising 3D renderings, all types of discontinuities may not always be available. False negatives caused by sharp texture and refracted geometries lead to blurs.

especially well suited for noisy input samples. Unfortunately, discontinuities must be given in advance as input, and the function space used for interpolation by Belhe et al. does not immediately make it clear how to treat discontinuity locations as optimization variables. When discontinuities are missing — even partially – their reconstruction is noticeably poor (see Fig. 2).

In this paper, we propose a non-trivial change to Belhe et al.'s method so that image fitting no longer requires discontinuities to be known in advance. During fitting, we treat *all* mesh edges as potential discontinuities, introducing variables to model the magnitude of value-jump across the edge. These variables are *continuous* and happily optimized along with feature vectors and mesh vertex positions during gradient-based reconstruction. Unlike Belhe et al.'s function space, our function space easily affords a post-processing procedure to identify and merge almost-continuous edges, greatly improving storage efficiency while maintaining reconstruction fidelity.

We demonstrate that our neural fields with learned discontinuities directly support denoising and super-resolution. Using a novel systematically synthesized diffusion curve dataset, we show that our method outperforms InstantNGP [Müller et al. 2022] with matching size by > 5dB in the denoising task and > 10dB when examined under super-resolution. Visually, our neural field maintains sharp region boundaries at large zoom levels (30× in Fig. 1,3,9) while InstantNGP blurs boundaries. Our method recover more accurate discontinuities than Mumford-Shah-based denoising [Wang et al. 2022]: 3.5× smaller Chamfer distance to the groundtruth. We show that our neural fields can approximate typical vector graphics images corrupted by JPEG compression. The use cases of our method also include general 2D data, such as diffusion-generated depth maps, which our method segments with clear cuts between depth discontinuities. Finally, we stress test our method with complicated artistic drawings and natural images (Fig. 1,10).

2 RELATED WORK

Geometric representation for images. Geometric image representations, such as vector graphics, address raster image limitations by encoding discontinuities as shapes with simple color functions. Methods combining accurate geometric boundaries with interior samples focus on representation design, interpolation, and real-time rendering [Bala et al. 2003; Parilov and Zorin 2008; Pavić and Kobbelt 2010; Ramanarayanan et al. 2004; Reshetov and Luebke 2016; Sen 2004; Tarini et al. 2005; Tumblin and Choudhury 2004]. Another approach represents digital images with discrete curves or region boundaries and smooth interior functions [Tian and Günther 2022]. Diffusion curves [Orzan et al. 2008] defines images as harmonic functions with curve Dirichlet boundary conditions, a space our method can accurately approximate (Section 5). Triangle mesh-based representations [Davoine et al. 1996; Demaret et al. 2006; Su and Willis 2004; Tu and Adams 2011] and more advanced curve-and patch-based primitives [Lai et al. 2009; Lecot and Levy 2006; Sun et al. 2007; Xia et al. 2009; Zhao et al. 2017] are introduced for vectorization of natural images. These approaches, while similar to ours in merging discrete geometries and functional representations for interiors, often construct geometric boundaries separately, relying on edge detection, segmentation, or user input. In contrast, our method jointly optimizes discontinuity locations and interior colors using a mesh without predefined cuts.

Neural fields. Neural fields are widely used for representing spatial functions like images and signed distance fields (SDFs), often parameterized by neural networks [Xie et al. 2022]. Early work by Song et al. [2015] introduced coordinate neural networks for image encoding, a technique applicable to large image compression [Mehta et al. 2021; Mildenhall et al. 2021; Sitzmann et al. 2020]. Although these methods capture high-frequency details, they often fail to represent true discontinuities, causing blurring at high zoom levels.

Hybrid neural fields, or feature fields, combine neural networks with discrete data structures like grids [Chen et al. 2022, 2021; Martel et al. 2021; Müller et al. 2022; Shen et al. 2021; Takikawa et al. 2021, 2023; Yu et al. 2021]. These offer reduced computation, better network capacity utilization, and explicit geometric representation. While most hybrid fields don't explicitly address discontinuities, recent methods like ReLU fields [Karnewar et al. 2022] attempt to approximate them with steep ReLUs, but still face blurring issues as shown by Belhe et al. [2023]. Other approaches, like those by Reddy et al. [2021], represent true discontinuities but are tailored for specific cases like fonts. Discontinuity-aware 2D neural fields [Belhe et al. 2023], an inspiration for our approach, show promise but require user-provided discontinuous edges (Fig. 2). Our method fits unknown discontinuities and is compatible with discontinuity-aware 2D neural field offering efficient storage and inference.

Concurrent work also uses supervised learning to generate a field of patches for discontinuity detection, especially in noisy environments [Polansky et al. 2024]. We compare to its preceding work [Verbin and Zickler 2021] and show that their focus is on denoising rather than accurate approximation (Fig. 3d).

Differentiable rendering. Addressing visual discontinuity in differentiable rendering is a persistent challenge [Spielberg et al. 2023;
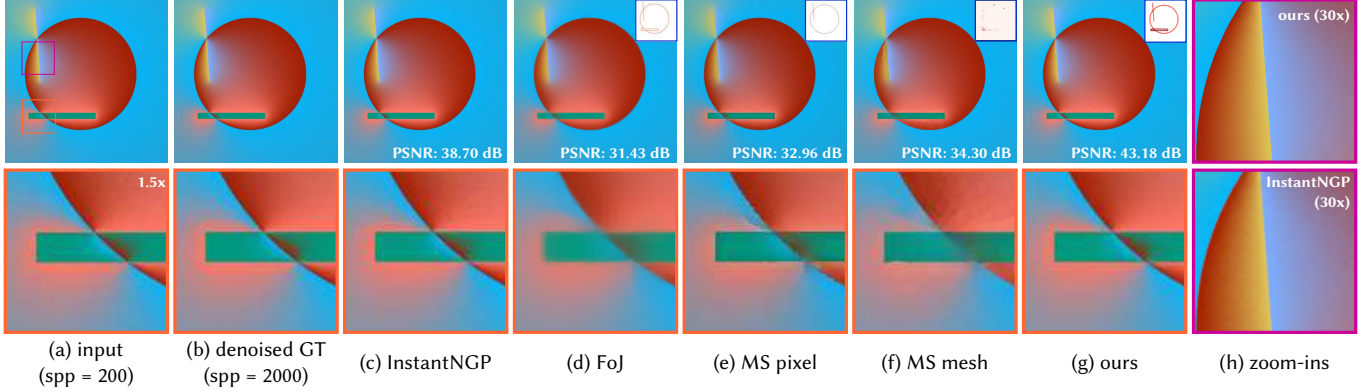
Fig. 3. (a,b) Diffusion curves define an example harmonic function field with sharp discontinuities [Orzan et al. 2008]. (c) Continuous neural fields, such as InstantNGP [Müller et al. 2022], do not represent discontinuities, resulting in blur image when zoomed-in. (d) Denoising method, Field of Junctions (FoJ) [Verbin and Zickler 2021], fails to recover clear discontinuities due to using constant-patch-based approximation. (e, f) Mumford-Shah functional based methods jointly approximate the target and detect discontinuities (see red edges in insets) [Wang et al. 2022], similar to our method. However, both versions fail to achieve both goals because of limited function expressiveness. (g) Our accurate approximation and recovered discontinuities.

Zhao et al. 2020]. Traditional automatic differentiations often overlook discontinuities' contributions to gradients [Bangaru et al. 2021]. Differentiable rendering techniques include smoothed rasterization [Kato et al. 2018; Laine et al. 2020; Liu et al. 2019; Loper and Black 2014], Monte Carlo-based methods with boundary and interior sampling [Bangaru et al. 2020; Li et al. 2018, 2020; Loubet et al. 2019; Zhang et al. 2023], analytic solutions [Bangaru et al. 2021], and finite difference approaches [Deliot et al. 2024; Yang et al. 2022]. Our approach introduces a differentiable neural primitive capable of representing discontinuities. By framing our approximation problem as an inverse rendering task, our method efficiently handles discontinuities by leveraging existing differentiable rendering techniques.

Mumford–Shah functional. The Mumford–Shah (MS) functional, proposed for image segmentation [Mumford and Shah 1989], models images as piecewise-smooth functions with explicit discontinuities. The Ambrosio-Tortorelli approximation [Ambrosio and Tortorelli 1990] made solving for MS functionals tractable with techniques like ADMM. This functional has been widely used in 2D image tasks [Le et al. 2022; Tsai et al. 2001; Vese and Chan 2002] and applied to 3D surfaces [Bonneel et al. 2018; Tong and Tai 2016; Wang et al. 2022] and volumes [Coeurjolly et al. 2016]. Our method, similar to the MS functional, jointly recovers piecewise-smooth functions and discrete discontinuities. Unlike level-set-based methods [Esedoglu and Shen 2002; Vese and Chan 2002], ours supports open boundaries and produces more accurate discontinuities free of narrow bands [Bonneel et al. 2018; Tong and Tai 2016] and staircasing artifacts [Le et al. 2022; Tsai et al. 2001]. While Wang et al. [2022] also discretize discontinuities on mesh edges, their representation (constant colors per face) lacks the expressiveness of our neural model (Fig. 3).
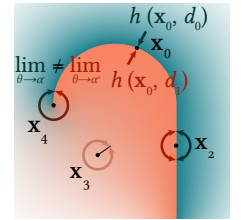
3 METHOD

Given a 2D image function $I(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^2$, we aim to approximate it with a 2D neural field that is continuous everywhere, except for locations with sharp changes in the input. In our implementation, $I(\mathbf{x})$ is given by nearest-neighbor sampling of a raster image.

The desired properties of such 2D neural fields are formulated by Belhe et al. [2023]. Let Ω be a 2D domain and $\Gamma = \{\gamma_0, \cdots, \gamma_{n-1} \mid \gamma_i \in \Omega, \forall i\}$ be curves that can only intersect with each other at endpoints $\partial\Gamma$. The directional discontinuity is defined with respect to a point $\mathbf{x} \in \mathbb{R}^2$, a polar coordinate system centered at $\mathbf{x}$ and a angular coordinate $\alpha \in \mathbb{R}$. The *directional limit* of a function at a position $\mathbf{x}$ along a direction θ is defined as $h(\mathbf{x}, \theta) = \lim_{r \to 0^+} f(C(r, \theta))$, where $C(r, \theta)$ maps the polar coordinate to Cartesian coordinate (see $\mathbf{x}_0$ in inset). A function is directionally discontinuous at $\mathbf{x}$ and α if

$$\lim_{\theta \to \alpha^+} h(\mathbf{x}, \theta) = \lim_{\theta \to \alpha^-} h(\mathbf{x}, \theta). \tag{1}$$

Following this continuity criteria, a field $f : \Omega \to \mathbb{R}^d$ is: (1) continuous at $\mathbf{x} \in \Omega \setminus \Gamma$ (e.g., $\mathbf{x}_3$ in inset); (2) directionally discontinuous for the two tangent directions at $\mathbf{x} \in \Gamma \setminus \partial\Gamma$ (e.g., $\mathbf{x}_2$); (3) directionally discontinuous at the tangent direction pointing inwards to the curve(s) at $\mathbf{x} \in \partial\Gamma$ (e.g., $\mathbf{x}_4$). Note that γ_i can be a curve [Belhe et al. 2023]. For simplicity,

we only consider the case where γ_i is a line segment and approximate curve geometries by adjusting triangle density.

Overview. Our neural field is built upon an underlying triangle mesh and discontinuous feature function defined locally within each vertex one-ring. To see how it works on triangle meshes, we first analyze the 1D case then extend the formulation to 2D (Section 3.1). This discontinuous feature function allows discontinuity across the mesh edges from which we learn the target discontinuous edge set Γ. We approximate the target image by initializing a triangle mesh (Section 3.2.1), fitting a neural field with all edges being potentially discontinuous (Section 3.2.2), and refining to enforce continuity on almost-continuous edges (Section 3.2.3).
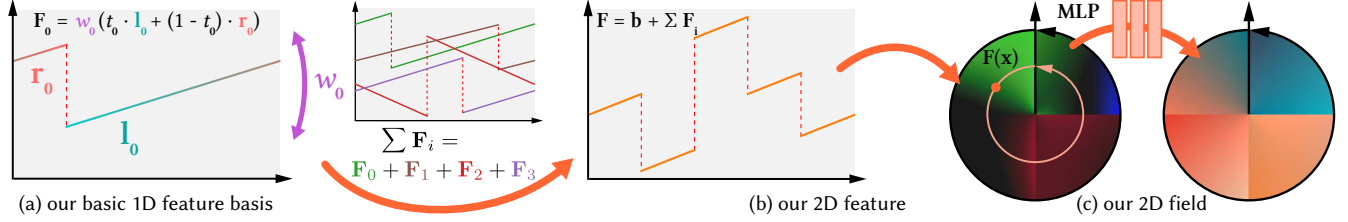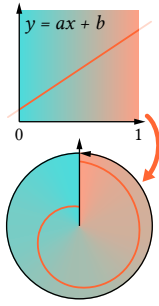
Fig. 4. (a) Our 1D feature basis functions fit discontinuous target function with continuous variables $(w, \mathbf{l}, \mathbf{r})$. (b) In a 2D vertex neighborhood, our 1D feature basis functions are defined along the radial direction. (c) The features have a constant piecewise slope per dimension, which is enhanced by an MLP.

3.1 Neural Field with Discontinuous Features

Given a triangle mesh in $\mathbb{R}^2$, we define learnable discontinuous feature function basis within vertex one-rings. Optimization of discontinuous functions would typically require discrete operations, challenging automatic differentiation. We observe that by treating all mesh edges as being *potentially discontinuous*, we can represent the magnitude of discontinuity with *continuous* variables. In this way, we can optimize the continuous variables with the standard autodiff-gradient-based approach. We first introduce our feature function basis in $\mathbb{R}$ (Section 3.1.1) then extend it to $\mathbb{R}^2$ (Section 3.1.2).

3.1.1 Discontinuous field in 1D.
Consider a simple 1D linear function $g(x) = ax + b$. We can restrict its domain to $x \in [0, 1)$ and map this function into the unit circle $\mathbb{S}^1$. In this way, we construct a discontinuity at $x = 0$ when $a \neq 0$. This construction enables standard autodiff to calculate the left and right derivatives of $g(x)$ at the discontinuity location. Given a closed polyline $S = (V, E)$ and an arbitrary reference coordinate $x \in \mathbb{S}^1$ on it, we define a local coordinate $x_i = t_i(x)$ centered at each vertex $v_i \in V$ and a linear function $\varphi_i(x_i) = a_i x_i + b_i$. These functions form a basis for piecewise linear functions

$$g(x) = \sum_{v_i \in V} \varphi_i(t_i(x)) = \sum_{v_i \in V} (a_i t_i(x) + b_i), \quad (2)$$

which is discontinuous at v_i if $a_i \neq 0$. Note that its discontinuous pieces all share the same slope of $\sum_{v_i \in V} a_i$ (see Fig. 4a).

This single-value basis has limited expressiveness. To improve this, we extend to k-dimensional features and over-parameterize.

$$\mathbf{F}(x) = \mathbf{b} + \sum_{v_i \in V} \mathbf{F}_i(t_i(x)) = \mathbf{b} + \sum_{v_i \in V} w_i (t_i(x) \cdot \mathbf{l}_i + (1 - t_i(x)) \cdot \mathbf{r}_i)$$
$$(3)$$

$$= \mathbf{b} + \sum_{v_i \in V} (w_i(\mathbf{l}_i - \mathbf{r}_i)t_i(x) + w_i \mathbf{r}_i) \quad (4)$$
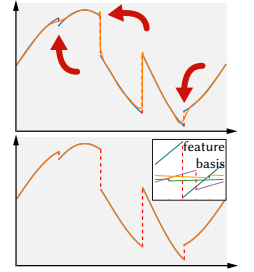
where, intuitively, $w_i \in \mathbb{R}$ controls whether a discontinuity is eliminated; $\mathbf{l}_i, \mathbf{r}_i \in \mathbb{R}^k$ are left and right features defined on either side of a vertex v_i; the last term in Eq. 3 defines a linear interpolation between these two features; $\mathbf{b} \in \mathbb{R}^k$ is a global bias (Fig. 4a).

We pass feature $\mathbf{F}(x)$ to an MLP ($\mathbb{R}^k \mapsto \mathbb{R}^3$) for our neural field

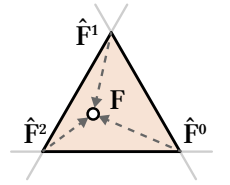$$f(x) = \text{MLP}(\mathbf{F}(x)). \quad (5)$$

Our neural fields use shallow MLPs: two layers of 64 neurons for 1D, two layers of 128 neurons for 2D, with tanh activations.

We conduct a simple 1D fitting experiment (inset). The target function (blue) is a function of translated sine pieces and discontinuities at four points. For reference, we construct a 1D feature field ($k = 5$) with features defined at these four locations and linearly interpolated between them (orange,top). We compare this reference to our 1D model ($k = 2$) with potential discontinuities at the matching points (orange,below; feature basis functions are $\mathbf{F}_i$ in Eq. 3). Contracted with our faithful approximation, the reference field not only fits the discontinuities (red dashed lines) with inaccurate steep continuous jumps but also fails to stay close to the continuous pieces.

3.1.2 Discontinuous field in 2D.
We extend this 1D field into 2D domain Ω on a triangle mesh $M = (V, T)$, where V, T are vertex and face set. The feature value at a point $\mathbf{x} \in \Omega$ intersecting a triangle $T_i = (v_i^0, v_i^1, v_i^2)$ is determined by the barycentric interpolation of three per-vertex local features

$$\mathbf{F}(\mathbf{x}) = (1 - \lambda_1 - \lambda_2)\hat{\mathbf{F}}_i^0(\mathbf{x}) + \lambda_1\hat{\mathbf{F}}_i^1(\mathbf{x}) + \lambda_2\hat{\mathbf{F}}_i^2(\mathbf{x}), \lambda_1, \lambda_2 \in [0, 1], \quad (6)$$
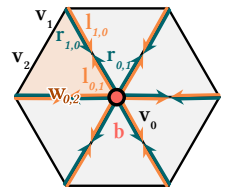
The local features are constructed similarly to the 1D feature introduced in Section 3.1.1. Given a vertex $\mathbf{v}_i$, the local feature is

$$\hat{\mathbf{F}}_i(\mathbf{x}) = \mathbf{b}_i + \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), i \neq j} \mathbf{F}_{i,j}(t_i^j(\mathbf{x})) \quad (7)$$

$$= \mathbf{b}_i + \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), i \neq j} w_{i,j} \left(t_i^j(\mathbf{x}) \cdot \mathbf{l}_{i,j} + (1 - t_i^j(\mathbf{x})) \cdot \mathbf{r}_{i,j} \right) \quad (8)$$

$$= \mathbf{b}_i + \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), i \neq j} \left(w_{i,j}(\mathbf{l}_{i,j} - \mathbf{r}_{i,j})t_i^j(\mathbf{x}) + w_{i,j}\mathbf{r}_{i,j} \right). \quad (9)$$

Here $\mathcal{N}(\mathbf{v}_i)$ is $\mathbf{v}_i$ one-ring and $\mathbf{v}_i, \mathbf{v}_j$ are connected by an edge; $\mathbf{b}_i \in \mathbb{R}^k$ is a bias defined at the center $\mathbf{v}_i$; $t_i^j(\mathbf{x})$ maps $\mathbf{x}$ to the local polar coordinate system centered at $\mathbf{v}_i$ with $(\mathbf{v}_i, \mathbf{v}_j)$ as the polar axis. To make the basis in Eq. 8 a linear interpolation, we normalize $t_i^j(\mathbf{x})$ from $[0, 2\pi)$ to $[0, 1)$. Note

that the bias $\mathbf{b}_i$ is introduced so a vertex reduces to a regular continuous vertex [Belhe et al. 2023] when all adjacent edges are continuous. The feature $\mathbf{l}_{i,j}, \mathbf{r}_{i,j} \in \mathbb{R}^k$ and the discontinuity weight $w_{i,j} \in \mathbb{R}$ are defined in the same way as the 1D case. The only difference is how $\mathbf{l}_{i,j}, \mathbf{r}_{i,j}, w_{i,j}$ are defined between adjacent vertices. The features $\mathbf{l}_{i,j} \neq \mathbf{l}_{j,i}, \mathbf{r}_{i,j} \neq \mathbf{r}_{j,i}$ are defined on the half-edges granting freedom for vertices to have local "colors" while the discontinuity weight $w_{i,j} = w_{j,i}$ are shared between two vertices for consistent continuity behavior along an edge.

Like our 1D case, we pass the interpolated feature $\mathbf{F}(\mathbf{x})$ through a shallow MLP (Algorithm 1 where AngleCCW computes CCW angle between two vectors).

Algorithm 1: Field Inference

Function *infer* (**x**, M)

    $T, \lambda_1, \lambda_2 \leftarrow$ PointInFace($\mathbf{x}, M$) ;

    forall $\mathbf{v}_i$ *in* $T = (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$ **do**

        forall e_i^j *in* *AdjacentHalfEdge*($\mathbf{v}_i$) **do**

            $\theta_i^j \leftarrow$ AngleCCW$((\mathbf{x} - \mathbf{v}_i), e_i^j)$;

            $t_i^j \leftarrow$ fmod$(\frac{\theta - \theta_i^j}{2\pi} + 1, 1)$;

        end

        $\hat{\mathbf{F}}_i(\mathbf{x}) \leftarrow$

        $\mathbf{b}_i + \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), i \neq j} \left(w_{i,j}(\mathbf{l}_{i,j} - \mathbf{r}_{i,j})t_i^j + w_{i,j}\mathbf{r}_{i,j} \right)$

        Eq. 8;

    end

    $\mathbf{F}(\mathbf{x}) \leftarrow (1 - \lambda_1 - \lambda_2)\hat{\mathbf{F}}_0(\mathbf{x}) + \lambda_1\hat{\mathbf{F}}_1(\mathbf{x}) + \lambda_2\hat{\mathbf{F}}_2(\mathbf{x})$ Eq. 6 ;

    return MLP($\mathbf{F}(\mathbf{x})$);

end

3.1.3 Comparison to feature space in [Belhe et al. 2023]. Belhe et al.'s per-vertex features are also defined in local polar coordinate

$$\hat{\mathbf{F}}_i = \mathbf{F}_i^{\text{cw}} \frac{\theta_i^{\text{ccw}}}{\theta_i^{\text{cw}} + \theta_i^{\text{ccw}}} + \mathbf{F}_i^{\text{ccw}} \frac{\theta_i^{\text{cw}}}{\theta_i^{\text{cw}} + \theta_i^{\text{ccw}}}. \tag{10}$$

This interpolation scheme (Eq. 2 [Belhe et al. 2023]) finds the closest discontinuous edges clockwise ("cw") and counter-clockwise ("ccw"), then radially (based on θs) interpolates the half-edge features $\mathbf{F}_i^{\text{cw}}, \mathbf{F}_i^{\text{ccw}}$ (corresponding to our $\mathbf{l}, \mathbf{r}$ features). Unlike the entire coverage of 2π of our interpolation, theirs only covers the domain between two consecutive discontinuities. Additionally, our local
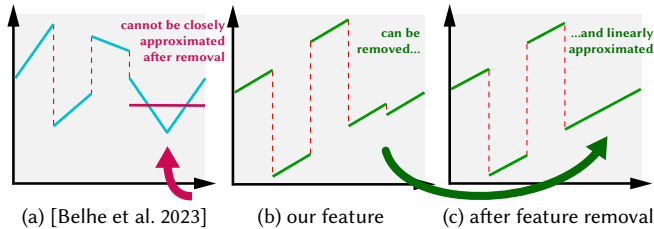

Fig. 5. (a) Our feature (green) differs from Belhe et al.'s feature (cyan). (b) This allows us to easily discard almost-continuous edges.
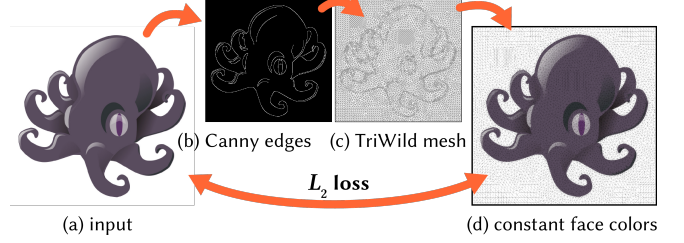

Fig. 6. We initialize by triangulating Canny edges [Canny 1986] with Tri-Wild [Hu et al. 2019], then deforming and remeshing interleavedly. The deformation is posed as per-face constant color approximation.

feature function have a constant slope in each piece while theirs can have various slopes (e.g., Fig. 5a). However, our design is tailored to handle unknown discontinuities. When a basis vanishes, the edge is continuous, we can easily remove its associated features. Compared to ours, Belhe et al.'s feature definition allows almost-continuous edges have significantly different slopes in its two adjacent domains. This difference makes linear approximation (magenta line in Fig. 5a) inaccurate after redundant feature removal, and as a result, Belhe et al.'s needs to save more redundant features.

The disadvantage of having constant piecewise slopes is mitigated by the MLP as experimentally demonstrated in Section 5. Furthermore, our feature and interpolation scheme can be easily shown to satisfy the continuity criteria proposed by Belhe et al. (Section 3). The discontinuous edge set Γ is a subset of the mesh edges E and consists of edges where $w_i(\mathbf{l}_{i,j} - \mathbf{r}_{i,j}) \neq \mathbf{0}$. On each edge, we define the largest feature value jump among all half edges and feature dimensions as $D_{i,j} = \max(w_i(\mathbf{l}_{i,j} - \mathbf{r}_{i,j}))$. This value, *discontinuity indicator*, reflects the magnitude of the feature space discontinuity.

3.2 Learning Discontinuous 2D Neural Field

3.2.1 Mesh initialization. We initialize the triangle mesh $M = (V, T)$ to be roughly aligned with target discontinuities (Fig. 6). Note that our method does not require exact edges as it is able to refine edge locations in the next optimization step (Section 3.2.2). But rough alignments are still necessary to obtain discontinuous edges in the early iterations so we can apply differentiable rendering techniques to modify discontinuous edge locations (Fig. 7).

We first roughly detect discontinuities with a Canny edge detector [Canny 1986]. We then connect positive pixels to their corresponding 8-neighbors and apply TriWild [Hu et al. 2019] to generate a triangle mesh M_0. Since the results from the Canny edge detector are inaccurate and limited to the pixel grid, it is only for ensuring the desired triangle density around potential discontinuities.

We use a field $f_0(\mathbf{x}; M)$ with per-face constant colors as a proxy for our initialization. We approximate the target with f_0 by optimizing

$$\min_M \int_\Omega \|f_0(\mathbf{x}; M) - I(\mathbf{x})\|^2 \, d\mathbf{x} + \lambda_{\text{boundary}} \frac{1}{|\partial M|} \sum_{\mathbf{v} \in \partial M} \|\mathbf{v} - \mathbf{v}^0\|^2, \tag{11}$$

where the second term is a MSE loss for softly fixing the boundary (∂M) to their initial positions $\mathbf{v}^0$s ($\lambda_{\text{boundary}} = 10^{-2}$). We discretize
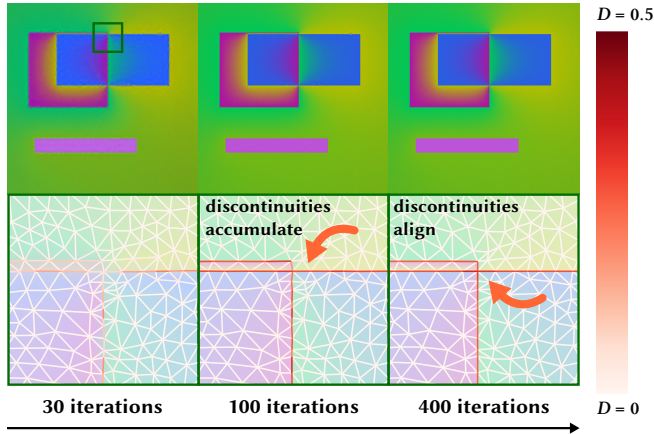
Fig. 7. We jointly optimize our field and its underlying mesh. (a) In the early epochs, our indicated discontinuities begin to accumulate around the target discontinuities. (b) As optimization progresses, our method produces close interior color approximation and simultaneously aligns discontinuities.

the first term by stratified sampling triangles and compute gradients using SoftRasterizer [Liu et al. 2019], which can be replaced by other differentiable renderers. We interleave the deformation with optional remeshing steps. See our supplemental document for details.

3.2.2 Field optimization. Given a roughly aligned triangle mesh M, we optimize our neural field to approximate the input (Fig. 7). Our loss function is defined as

$$L = \int_\Omega \|f(\mathbf{x}; V, \Theta) - I(\mathbf{x})\|^2 \, d\mathbf{x} + \lambda_{\text{discont}} \int_E \|w(\mathbf{1} - \mathbf{r})\|_1 \, d\mathbf{x}, \quad (12)$$

where the first term is a regular L_2 fitting loss with varying vertex positions V and neural field parameters Θ (MLP parameters, feature functions $\mathbf{F}$, and biases $\mathbf{b}$); the second term is a sparsity inducing term ($\lambda_{\text{discont}} = 5 \times 10^{-3}$) penalizing feature space value jumps across edges. We discretize the second term into $\sum_{e_{i,j} \in H} \|e_{i,j}\| \cdot \|w_{i,j}(\mathbf{l}_{i,j} - \mathbf{r}_{i,j})\|_1$, where $e_{i,j} \in H$ is a half-edge with length $\|e_{i,j}\|$, and $w_{i,j}, \mathbf{l}_{i,j}, \mathbf{r}_{i,j}$ are corresponding feature variables.

Note that our loss is similar to that of the MS functional, which contains a fitting term (corresponding to our first term), a smoothness term and a discontinuity sparsifying term (corresponding to our second term). We do not include an explicit smoothness term and rely on MLP's smoothness bias.

The first term depends on our field function f with discontinuities defined by vertices V and feature functions $\mathbf{F}$. To correctly estimate the gradient of this integral, we apply edge-sampling Monte Carlo estimation [Li et al. 2018]. We observe that discontinuity indicator $D = \max(w(\mathbf{1} - \mathbf{r}))$ gives a reasonable estimation about discontinuous edges. As continuous edges do not contribute to this gradient, we importance sample the discontinuous edges for efficiency. Although the initial mesh contains edges close to the exact discontinuity locations, the imperfection can result in low discontinuity indication in areas adjacent to true discontinuities. Because of this, we extend the important edge set to edges with $D > \beta$ and their adjacent edges.
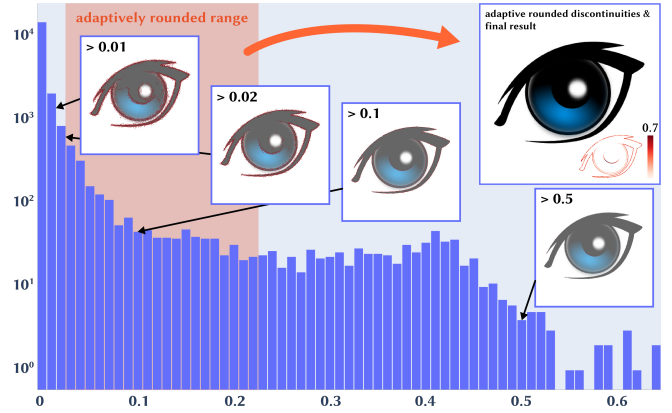

Fig. 8. *Example log histogram of discontinuity indicator D.* The majority of edges are identified as continuous, allowing for efficient optimization and storage of our neural field. We apply an adaptive greedily rounding strategy to flexibly determine the final discontinuous edges.

We sample these important edges with 5× probabilities than other edges. For detailed gradient formulas, see our supplemental.

3.2.3 Rounding. Once our neural field sufficiently approximates the input, we discard features on almost-continuous edges. We refer to this step as "rounding", a term borrowed from integer programming [Conforti et al. 2014]. We only round in one direction—setting $w = 0$ on almost-continuous edges while leaving the remaining ws free.

We employ an adaptive rounding strategy (Fig. 8). First, we round by simply thresholding the discontinuity indicator D, labeling all edges with $D < \beta$ as continuous. Then, we greedily discard edges using a priority queue of D. We discard an edge if the accumulated MSE increase is less than σ and keep it otherwise. To discard an edge, we push its contribution $0.5 \cdot w(\mathbf{1} - \mathbf{r}) + w\mathbf{r}$ to the center vertex bias. Thanks to the locality of our features, this greedily rounding can be done efficiently. After the rounding step, we continue refining this new rounded neural field and the mesh with more iterations.

4 IMPLEMENTATION

We implemented our method in PyTorch [Paszke et al. 2019]. To avoid flipped triangles caused by sparse gradients, we apply the large-step precondition [Nicolet et al. 2021] for mesh initialization and field optimization , with weight of 1, 0.5 respectively. We use ADAM [Kingma and Ba 2015] for both steps with $(\beta_1, \beta_2) = (0.9, 0.999)$ and learning rates of 1 and 2×10^{-2} respectively.

To ensure our ws serving as a control over discontinuities, we assign $w = \text{sigmoid}(10\tilde{w})$ where $\tilde{w}$ is the actual feature parameter. During the optimization, we use subpixel stratified sampling of spp $= 2^2$, a key to seamlessly accounting for input anti-aliasing; we set the edge sample number to $4^2 \times W \times H$. We accumulate gradients from all interior and edge samples per epoch, scheduling our optimization with 70 epochs of interior fitting and 130 epochs of interior and edge optimization. After the rounding step, we continue optimization with both sampling for additional 200 epochs.

We demonstrate a typical distribution of D (Fig. 8). The threshold β defines the hard cutoff of continuous edges and mostly affect the

Table 1. Quantitative measures.

Methods	Denoising			Denoising + Super-resolution (2×)	
	PSNR ↑	LPIPS ↓	Median Chamfer ↓	PSNR ↑	LPIPS ↓
Per-Vertex	30.758	0.0593	-	30.344	0.0747
InstantNGP	39.016	0.0431	-	32.715	**0.0261**
MS Pixel	35.563	0.0319	-	-	-
MS Mesh	36.159	0.0321	0.580	36.118	0.0466
Ours	**44.486**	**0.0261**	**0.165**	**43.913**	0.0423

computational cost. The MSE change threshold σ directly affects the result. We set $\beta = 0.02$, $\sigma = 5 \times 10^{-6}$ when not explicitly specified.

5 EVALUATION

We evaluate our method with various applications: denoising, super-resolution, segmentation and vectorization. We approximate different types of target functions: constant functions, gradients in vector graphics, harmonic functions in diffusion curves, more complicated functions in synthetic 3D renderings (Fig. 2), human-drawn artistic images, natural images, and spatial data (depth maps). We compare to continuous neural fields, represented by a simple reference feature field with per-vertex features and InstantNGP [Müller et al. 2022], and demonstrate that discontinuity representation is necessary for these tasks. We evaluate the accuracy of our detected discontinuities by comparing against pixel-based MS functional and a recent triangle-mesh-based MS method [Wang et al. 2022]. See our supplemental document for setup details.

Denoising. Noise in images come from many sources. We test on diffusion curves with noisy Monte Carlo samples rendered using walk-on-spheres (WoS) method [Muller 1956; Sawhney and Crane 2020]. We randomly generate 40 diffusion curves in the shapes of line segments, rectangles, and circles. Half of these only contains integer-coordinate rectangles and thus has no anti-aliasing. These random diffusion curves are rendered with a low number of samples per pixel (spp = 200) in 512^2 resolution and the ground truth (GT) denoised images are generated with spp = 2000. As demonstrated in Fig. 3,9, our field simultaneously approximates the target harmonic functions and denoises thanks to the smoothness bias of MLP, while the comparison methods either struggles to approximate or over-fits to noise. We conduct quantitative comparisons (Table 1), in which our method achieves a significant improvement (> 5dB) over the second best method, InstantNGP. Moreover, we verify our method's capability of recovering discontinuities. We quantify the performance by measuring Chamfer distance between the ground truth diffusion curve *geometries* and the discontinuous *edges* detected by mesh-based MS [Wang et al. 2022] and our method. Mesh-based MS produces discontinuous edges that are 3.5× Chamfer distance away from the GT compared to our results. See Fig. 9 for visuals.

Additionally, we qualitatively evaluate our method with the task of denoising JPEG-compressed vector images (constant color fills and human-created gradients). Our method not only approximates these two types of functions but also reduces JPEG compression artifacts without prior knowledge about the task.

Super-Resolution. We qualitatively show zoom-ins (1.5×, 2×, 4×, 30×) of the approximation results across this paper (Fig. 1,2,3,9). As presented by the previous work Belhe et al. [2023], although continuous neural fields like InstantNGP closely approximate the input in the original scale, once zoomed in, especially at high zoom levels, they start to exhibit blurs due to lack of discontinuity representation. We combine the super-resolution quantitative evaluation with the denoising evaluation by measuring the same resulting approximations with 2× zooms. We measure against the clean diffusion curve images with spp = 2000 and 1024^2 canvas size. We remark that while the two discontinuity-aware methods, mesh-based MS and ours, output results with similar PSNR as the denoising evaluation, InstantNGP's results experience a drop of PSNR due to the blur artifacts, widening the gap with our method. LPIPS attempts to measure semantic similarity and is less sensitive than PSNR to the absolute function values in the vicinity of discontinuities.

Segmentation. Beyond RGB(A) images, 2D image functions also include general 2D functions, *spatial data*, such as depth, normal maps, optical flows, and even CLIP feature maps [Radford et al. 2021]. We evaluate on diffusion-generated depth maps [Ke et al. 2023] (Fig. 11). Unlike typical scanned depth data, they contain noise originated from neural network, especially around depth discontinuities. We demonstrate that our method can conduct edge-based segmentation using clean cuts separating depth discontinuities.

Vectorization. Apart from the synthetic images, we approximate artistic drawings and natural images. This approximation process can be considered as vectorization since the resulting discontinuous neural fields are resolution-independent, an important property of vector images. In Fig. 10a, we compare our approximation of artistic inputs to the vectorization of Adobe Illustrator Image Trace, a typical tool using solid fills. Contrasted with the limited expressiveness of solid fills, our neural field is able to preciously represent the targets and support clear zoom-in views. As solid-fill vectorization tools, e.g., Adobe Illustrator, Potrace, dominate for this task, we also demonstrate a blending application. By adjusting the rounding parameters, our neural field can blend solid-fill regions, creating smooth color gradients (Fig.12). We stress test our method on natural images. We show in Fig. 10b two successful approximations with mild denoising effects, similar to the edge-preserving bilateral filter.

6 LIMITATIONS AND FUTURE WORK

Our method only supports regular triangle meshes instead of the curved triangle meshes as Belhe et al.'s, which remains future work. We expect this extension to further improve the ability of our neural field to approximate natural images (Fig. 10). Additionally, our fitting procedure replies on the initial mesh to be reasonably aligned with the target discontinuities. This may not hold when the target visual feature is small and close to the size of noise or when the colors across the discontinuities are close. Applying subdivision or remeshing to our neural fields as well as exploring more flexible underlying structures and adapting our proposed feature functions to these structures could be interesting future directions.

Our field is single-level compared to neural fields with multi-level feature grids, such as InstantNGP [Müller et al. 2022]. This design

provides denoising power and compressed field size—sizes of our fields commonly match those of InstantNGPs with 2^{14} hash table size (compared to their default 2^{24}). Despite these benefits, efficient representation of high-frequency details, which can stay homogeneous within a continuous region, is a relevant open question.

7 CONCLUSION

We introduce a mesh-based discontinuous 2D neural field that learns unknown discontinuities, advancing the discontinuity-aware neural fields proposed by Belhe et al. [2023]. By treating all mesh edges as potential discontinuities, our method solves for the magnitude of discontinuities through continuous optimization. Our approach outperforms continuous neural fields, such as InstantNGP [Müller et al. 2022], in denoising and super-resolution tasks, maintaining sharp boundaries even at high zoom levels. Our improvement of Belhe et al.'s initial framework provides immediate benefits for applications, including the vectorization of artistic drawings and photos, and the cleanup of diffusion-generated depth data.

REFERENCES

Marc Alexa. 2019. Harmonic triangulations. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–14.

Luigi Ambrosio and Vincenzo Maria Tortorelli. 1990. Approximation of functional depending on jumps by elliptic functional via Γ-convergence. *Communications on Pure and Applied Mathematics* 43, 8 (1990), 999–1036.

Kavita Bala, Bruce Walter, and Donald P Greenberg. 2003. Combining edges and points for interactive high-quality rendering. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 631–640.

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased warped-area sampling for differentiable rendering. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–18.

Sai Praveen Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. 2021. Systematically differentiating parametric discontinuities. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–18.

Yash Belhe, Michaël Gharbi, Matthew Fisher, Iliyan Georgiev, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Discontinuity-aware 2D neural fields. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2023)* 41, 6 (2023). https://doi.org/10.1145/3550454.3555484

Nicolas Bonneel, David Coeurjolly, Pierre Gueth, and Jacques-Olivier Lachaud. 2018. Mumford-Shah Mesh Processing using the Ambrosio-Tortorelli Functional. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 75–85.

John Canny. 1986. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), 679–698.

Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. 2022. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*. Springer, 333–350.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. 2021. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8628–8638.

David Coeurjolly, Marion Foare, Pierre Gueth, and Jacques-Olivier Lachaud. 2016. Piecewise smooth reconstruction of normal vector field on digital data. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 157–167.

Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. 2014. *Integer programming models*. Springer.

Franck Davoine, Marc Antonini, J-M Chassery, and Michel Barlaud. 1996. Fractal image compression based on Delaunay triangulation and vector quantization. *IEEE Transactions on Image Processing* 5, 2 (1996), 338–346.

Thomas Deliot, Eric Heitz, and Laurent Belcour. 2024. Transforming a Non-Differentiable Rasterizer into a Differentiable One with Stochastic Gradient Estimation. *arXiv preprint arXiv:2404.09758* (2024).

Laurent Demaret, Nira Dyn, and Armin Iske. 2006. Image compression by linear splines over adaptive triangulations. *Signal Processing* 86, 7 (2006), 1604–1616.

Selim Esedoglu and Jianhong Shen. 2002. Digital inpainting based on the Mumford–Shah–Euler image model. *European Journal of Applied Mathematics* 13, 4 (2002), 353–370.

Yixin Hu, Teseo Schneider, Xifeng Gao, Qingnan Zhou, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2019. TriWild: robust triangulation with curve constraints. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–15.

ImageMagick. [n. d.]. *ImageMagick*. https://imagemagick.org

Inkscape. [n. d.]. *Inkscape*. https://inkscape.org/

Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. 2022. ReLU fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3907–3916.

Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. 2023. Repurposing diffusion-based image generators for monocular depth estimation. *arXiv preprint arXiv:2312.02145* (2023).

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA.

Yu-Kun Lai, Shi-Min Hu, and Ralph R Martin. 2009. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–8.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (ToG)* 39, 6 (2020), 1–14.

Hoang Trieu Vy Le, Marion Foare, and Nelly Pustelnik. 2022. Proximal based strategies for solving Discrete Mumford-Shah with Ambrosio-Tortorelli penalization on edges. *IEEE Signal Processing Letters* 29 (2022), 952–956.

Gregory Lecot and Bruno Levy. 2006. Ardeco: Automatic region detection and conversion. In *17th Eurographics Symposium on Rendering-EGSR'06*. 349–360.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–11.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*. 7708–7717.

Matthew M Loper and Michael J Black. 2014. OpenDR: An approximate differentiable renderer. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*. Springer, 154–169.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

Julien NP Martel, David B Lindell, Connor Z Lin, Eric R Chan, Marco Monteiro, and Gordon Wetzstein. 2021. Acorn: Adaptive coordinate networks for neural scene representation. *arXiv preprint arXiv:2105.02788* (2021).

Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. 2021. Modulated periodic activations for generalizable local functional representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 14214–14223.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. NeRF: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Mervin E Muller. 1956. Some continuous Monte Carlo methods for the Dirichlet problem. *The Annals of Mathematical Statistics* (1956), 569–589.

Thomas Müller. 2021. *tiny-cuda-nn*. https://github.com/NVlabs/tiny-cuda-nn

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

David Bryant Mumford and Jayant Shah. 1989. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on pure and applied mathematics* (1989).

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–13.

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion curves: a vector representation for smooth-shaded images. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 1–8.

Werner Palfinger. 2022. Continuous remeshing for inverse rendering. *Computer Animation and Virtual Worlds* 33, 5 (2022), e2101.

Evgueni Parilov and Denis Zorin. 2008. Real-time rendering of textures with feature curves. *ACM Transactions on Graphics (TOG)* 27, 1 (2008), 1–15.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

Darko Pavić and Leif Kobbelt. 2010. Two-Colored Pixels. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 743–752.

Mia Gaia Polansky, Charles Herrmann, Junhwa Hur, Deqing Sun, Dor Verbin, and Todd Zickler. 2024. Boundary Attention: Learning to Localize Boundaries under High Noise. arXiv:2401.00935 [cs.CV]

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

Ganesh Ramanarayanan, Kavita Bala, and Bruce Walter. 2004. *Feature-based textures*. Technical Report. Cornell University.

Pradyumna Reddy, Zhifei Zhang, Zhaowen Wang, Matthew Fisher, Hailin Jin, and Niloy Mitra. 2021. A multi-implicit neural representation for fonts. *Advances in Neural Information Processing Systems* 34 (2021), 12637–12647.

Alexander Reshetov and David Luebke. 2016. Infinite resolution textures.. In *High Performance Graphics*. 139–150.

Rohan Sawhney and Keenan Crane. 2020. Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains. *ACM Transactions on Graphics* 39, 4 (2020).

Pradeep Sen. 2004. Silhouette maps for improved texture magnification. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. 65–73.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. 2021. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems* 34 (2021), 6087–6101.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems* 33 (2020), 7462–7473.

Ying Song, Jiaping Wang, Li-Yi Wei, and Wencheng Wang. 2015. Vector regression functions for texture compression. *ACM Transactions on Graphics (TOG)* 35, 1 (2015), 1–10.

Andrew Spielberg, Fangcheng Zhong, Konstantinos Rematas, Krishna Murthy Jatavallabhula, Cengiz Oztireli, Tzu-Mao Li, and Derek Nowrouzezahrai. 2023. Differentiable visual computing for inverse problems and machine learning. *Nature Machine Intelligence* 5, 11 (2023), 1189–1199.

Dan Su and Philip Willis. 2004. Image interpolation by pixel-level data-dependent triangulation. In *Computer graphics forum*, Vol. 23. Wiley Online Library, 189–201.

Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. 2007. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics (TOG)* 26, 3 (2007), 11–es.

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11358–11367.

Towaki Takikawa, Thomas Müller, Merlin Nimier-David, Alex Evans, Sanja Fidler, Alec Jacobson, and Alexander Keller. 2023. Compact Neural Graphics Primitives with Learned Hash Probing. In *SIGGRAPH Asia 2023 Conference Papers*. 1–10.

Marco Tarini, Paolo Cignoni, et al. 2005. Pinchmaps: Textures with customizable discontinuities. In *Computer Graphics Forum*, Vol. 24. 557–568.

Xingze Tian and Tobias Günther. 2022. A survey of smooth vector graphics: Recent advances in representation, creation, rasterization and image vectorization. *IEEE Transactions on Visualization and Computer Graphics* (2022).

Weihua Tong and Xuecheng Tai. 2016. A variational approach for detecting feature lines on meshes. *Journal of Computational Mathematics* (2016), 87–112.

Andy Tsai, Anthony Yezzi, and Alan S Willsky. 2001. Curve evolution implementation of the Mumford-Shah functional for image segmentation, denoising, interpolation, and magnification. *IEEE transactions on Image Processing* 10, 8 (2001), 1169–1186.

Xi Tu and Michael D Adams. 2011. Image representation using triangle meshes with explicit discontinuities. In *Proceedings of 2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. IEEE, 97–101.

Jack Tumblin and Prasun Choudhury. 2004. Bixels: Picture samples with sharp embedded boundaries. *Rendering Techniques* 255 (2004), 264.

Dor Verbin and Todd Zickler. 2021. Field of Junctions: Extracting Boundary Structure at Low SNR. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 6869–6878.

Luminita A Vese and Tony F Chan. 2002. A multiphase level set framework for image segmentation using the Mumford and Shah model. *International journal of computer vision* 50 (2002), 271–293.

Chunxue Wang, Zheng Liu, and Ligang Liu. 2022. Feature-preserving Mumford–Shah mesh processing via nonsmooth nonconvex regularization. *Computers & Graphics* 106 (2022), 222–236.

Tian Xia, Binbin Liao, and Yizhou Yu. 2009. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Transactions on Graphics (TOG)* 28, 5 (2009), 1–10.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2022. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 641–676.

Yuting Yang, Connelly Barnes, Andrew Adams, and Adam Finkelstein. 2022. A δ: autodiff for discontinuous programs-applied to shaders. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–24.

Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. 2021. Plenoctrees for real-time rendering of neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 5752–5761.

Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. 2023. Projective Sampling for Differentiable Rendering of Geometry. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–14.

Shuang Zhao, Frédo Durand, and Changxi Zheng. 2017. Inverse diffusion curves using shape optimization. *IEEE transactions on visualization and computer graphics* 24, 7 (2017), 2153–2166.

Shuang Zhao, Wenzel Jakob, and Tzu-Mao Li. 2020. Physics-based differentiable rendering: a comprehensive introduction. *ACM SIGGRAPH 2020 Courses* 14 (2020), 1–14.
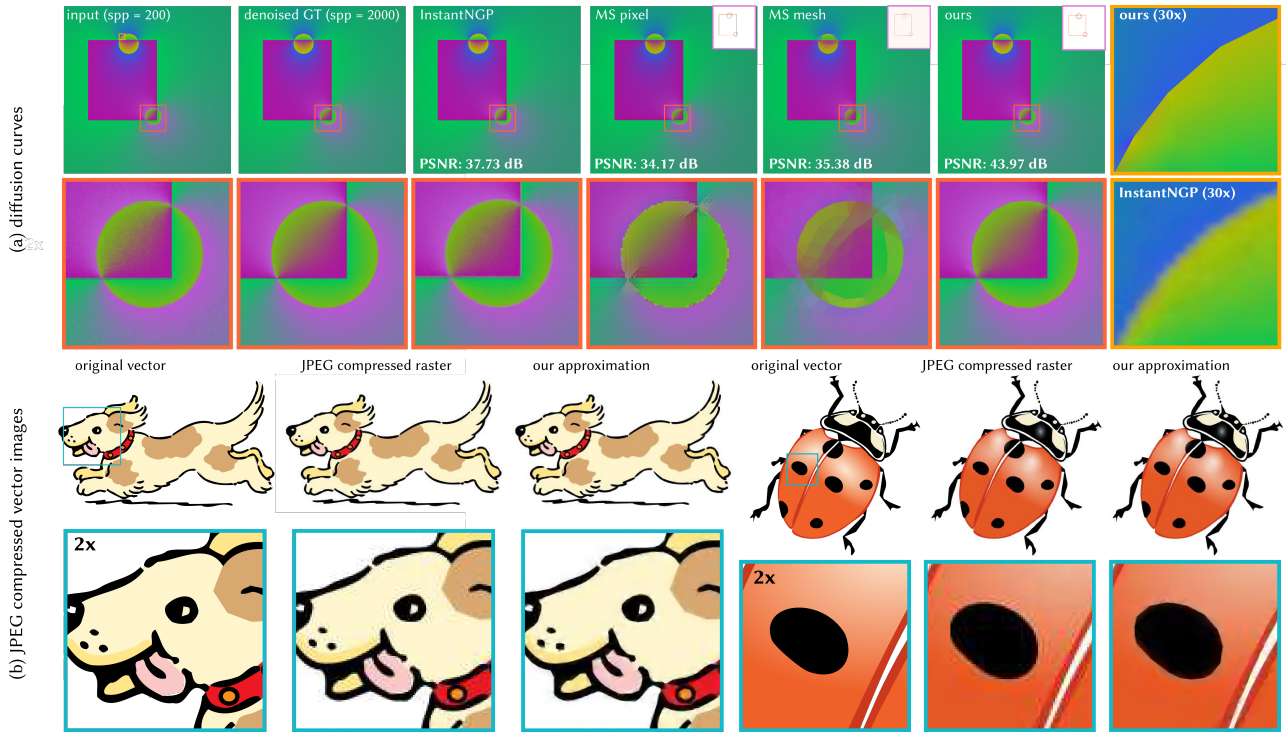
Fig. 9. *Denoising.* (a) We evaluate our model on randomly generated diffusion curves and assess the results for denoising effects and image quality under zoom, compared against continuous neural fields, e.g., InstantNGP [Müller et al. 2022], and MS functional based methods [Wang et al. 2022]. (b) We verify that our neural field can approximate simple constant color fills and gradients under JPEG compression.

(a) artistic drawing approximations
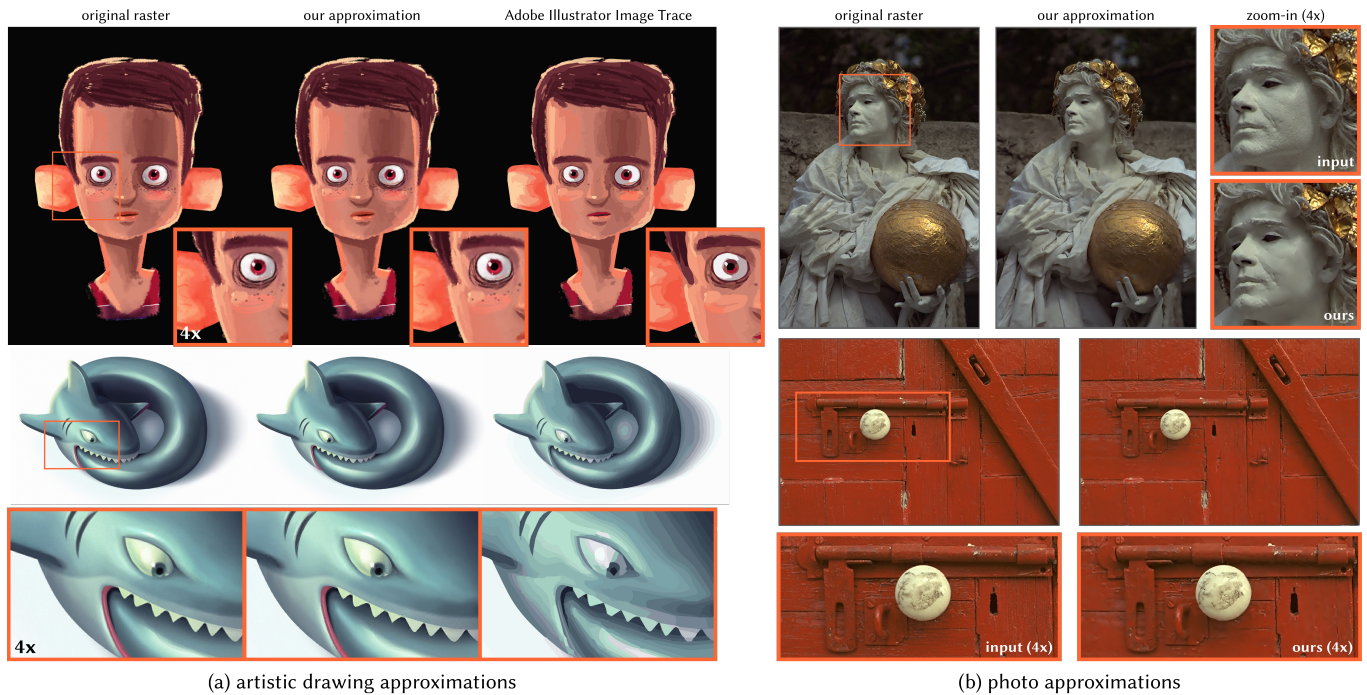
(b) photo approximations

Fig. 10. *Vectorization of artistic and natural images.* (a) Our method qualitatively outperforms common vectorization methods in terms of approximation and representation. (b) Our algorithm can vectorize natural images into resolution-independent images with sharp region boundaries when zoomed in.

(a) diffusion generated depth map & our cleanup (b) lifted point clouds (c) point cloud zoom-ins
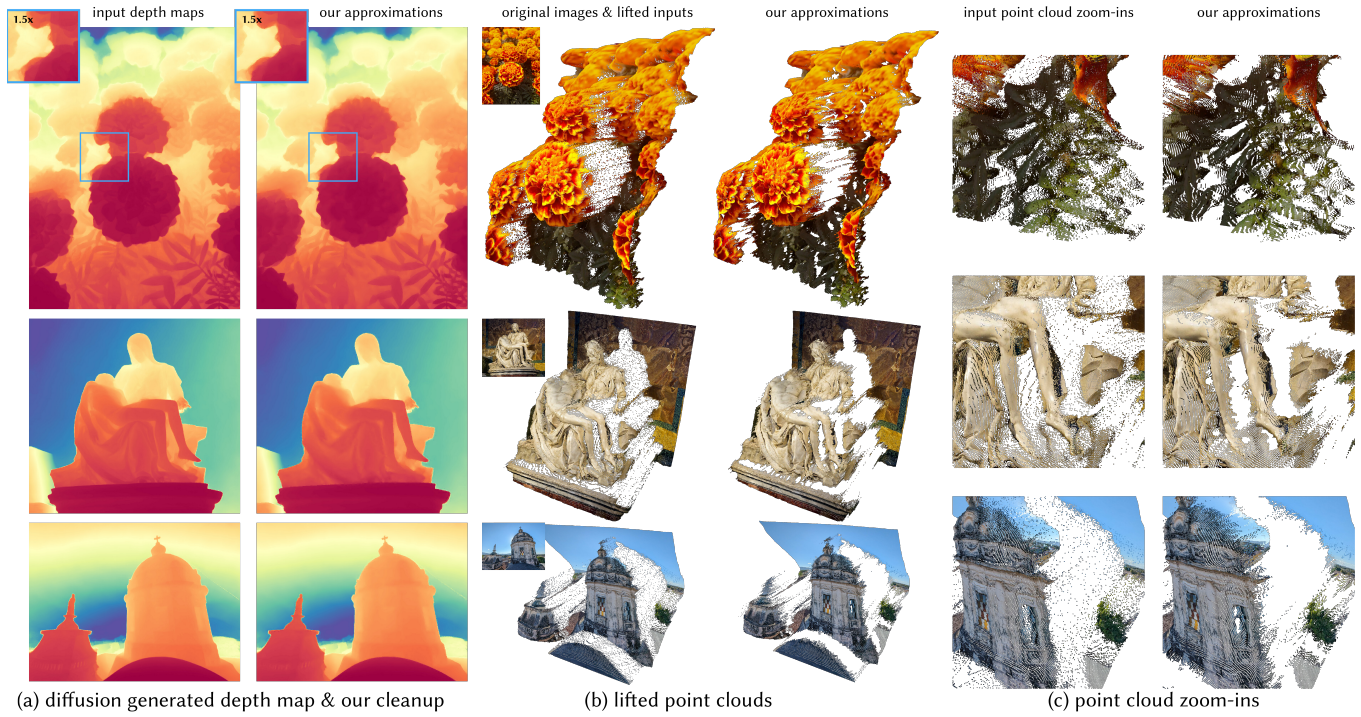
Fig. 11. *Segmentation of diffusion-generated depth data.* (a) Depth map generated by diffusion model [Ke et al. 2023] is corrupted by model-induced noises, particularly around discontinuities. (b) Our neural fields closely approximate input data while yielding clean cuts between depth discontinuities. See (c) for point cloud zoom-ins.
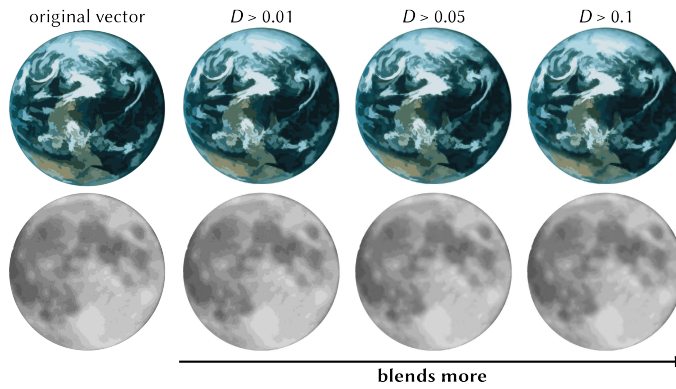

Fig. 12. *Blending posterized vector images.* Vectorization using only solid color fills generates posterized vector images. Our method can be applied to blend solid fills and create a resolution-independent image with color gradients.

A IMPLEMENTATION DETAILS

Mesh initialization. During our mesh initialization, we interleave the deformation with optional remeshing steps. In a remeshing step, we apply in order

(1) Edge collapses: We remove a face with area smaller than 2×10^{-5} of the canvas area or with an angle greater than $120°$ by collapsing its shortest edge.

(2) Edge splits: We split a face with L_2 fitting loss greater than $L_{\text{split}} = 2$ by splitting its longest edge at the mid point.

(3) Edge flips: We first flip edges so the triangulation is Delaunay then apply flips to minimize a hybrid loss of L_2 fitting loss and Delaunay loss.

The hybrid loss is defined as

$$\sum_{e \in F_i, F_j} \left\| f_0(x; F_i, F_j \in M) - I(x) \right\|^2 + \lambda_{\text{Delaunay}} \text{trace}(\mathbf{L}), \quad (13)$$

where F_i, F_j are the two faces defined by an edge; $\mathbf{L}$ is the positive semi-definite cotan Laplacian operator defined by the four vertices in F_i, F_j; $\lambda_{\text{Delaunay}}$ is set to 0.5. It is shown that the trace of cotan Laplacian is decreased by Delaunay flips and reaches a global minimum when the 2D triangulation is Delaunay [Alexa 2019].

Field optimization. To jointly optimize our neural field in triangle interiors and the triangle mesh, we compute gradient of the rendering integral with respect to moving triangle area. This is achieved via common differentiable rendering technique of edge-sampling Monte Carlo estimation [Li et al. 2018]

$$\nabla \int_\Omega \| f(\mathbf{x}; V, \Theta) - I(x) \|^2 \, d\mathbf{x} \quad (14)$$

$$= \left(\nabla^T \int_\Omega f(\mathbf{x}; V, \Theta) d\mathbf{x} \right) \cdot \int_\Omega (f(\mathbf{x}; V, \Theta) - I(\mathbf{x})) d\mathbf{x}. \quad (15)$$

This gradient can be estimated as

$$\nabla \int_\Omega f \, dx \quad (16)$$

$$= \int_{\Omega \backslash E} \frac{\partial}{\partial \Theta} f \, d\mathbf{x} + \sum_{e_i \in E} \int_{\alpha_i(\mathbf{x}) = 0} \frac{\nabla_V \alpha_i(\mathbf{x})}{\| \nabla_V \alpha_i(\mathbf{x}) \|} f(\mathbf{x}) \, dp(\mathbf{x}), \quad (17)$$

where the first term is the regular gradient of the field with respect to the field parameter in the face interiors; the second term is the gradient with respect to the vertices along discontinuous edges. The edges in the second term is defined by implicit functions $\alpha(x)$.

B PARAMETER CONFIGURATIONS

Mesh Initialization. We detect Canny edges with low and high thresholds of 100 and 200. To reduce the salt-and-pepper noises in the 3D renderings, we smooth the image with Gaussian kernel of size 3 before Canny edge detection. We call TriWild with target edge length ratio of 10^{-2}. We deform the mesh with SoftRasterizer (sharpness kernel size of 10^{-1}) for 200 epochs and remesh every 50 epochs. The per-face colors are set to be the face mean color in every iteration rather than kept as variables. Our remeshing implementation is based on the continuous remeshing PyTorch implementation [Palfinger 2022]. We employ per-triangle stratified sampling to avoid flipping very small triangles during mesh initialization. For the artistic drawing inputs where noise is minimal and
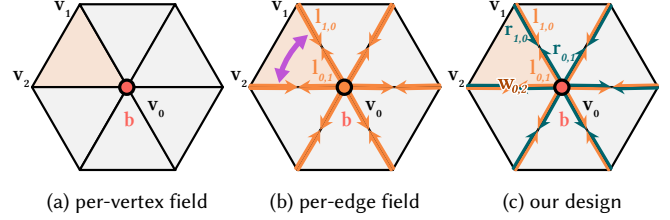

Fig. 13. Different designs of mesh-based feature fields.

faithful approximation is the goal, the TriWild target edge length ratio is adjusted to 3×10^{-3} and the mesh deformation is run for 100 epochs without remeshing.

Field Optimization. We initialize all our neural field weights using standard Xavier normal distribution and all biases as zeros.

C COMPARISON SETUP

Rendering. We render the resolution-independent results of mesh-based MS, InstantNGP, and our method using subpixel grid samples (spp = 4^2) for better visual effects given the target resolution.

JPEG compressed vector images. We generate input raster image by normalizing a vector image such that its longer axis occupies 90% of the 512×512 white canvas, then rasterizing the image with Inkscape and compressing it into a JPEG image with quality of 50 via ImageMagick.

InstantNGP. We use the tiny-cuda-nn [Müller 2021] based implementation for subpixel inference. We match the size of InstantNGP and our neural field by adjusting InstantNGP's hash table size. We ensure InstantNGP's convergence by running for $10k$ iterations.

Mumford-Shah functional. We reimplement a mesh-based MS denoising algorithm [Wang et al. 2022] and run it on the resulting aligned mesh from our method. The pixel-based MS denoising algorithm is implemented by replacing Wang et al.'s discretization with the one on pixel grids. We conduct experiments with parameters: $\alpha = 1, \beta = 0.01, \gamma = 100, epsilon = 0.01$ and 10 iterations of alternating optimization. For comparison fairness, we grid search the discontinuity threshold of mesh-based MS in the range of $v = [0.01, 0.09]$ (step size of 0.01) and $[0.1, 0.5]$ (step size of 0.1) given the Chamfer distance.

Field of Junctions. We use the official implementation of FofJ [Verbin and Zickler 2021]. We manually tune the parameters for the most accurate approximation. Our experiment uses $\eta = 0.01$, $\delta = 0.1$, patch size $R = 17$, stride $s = 5$ for a 512×512 image, consistency weights $\lambda_B = 0.5, \lambda_C = 0.1, N_{init} = 30$ iterations, $N_{iter} = 1000$ iterations, the number of values to query $N = 10$ in Algorithm 2, learning rate of 0.03 for the vertex positions and 0.003 for the junction angles in the refinement step to generate the global boundaries and smoothed image.

D ABLATIONS

Feature design. We compare to two simpler versions of our neural field: per-vertex field and per-edge field (Fig. 13). The per-vertex

Table 2. Ablation of feature design.

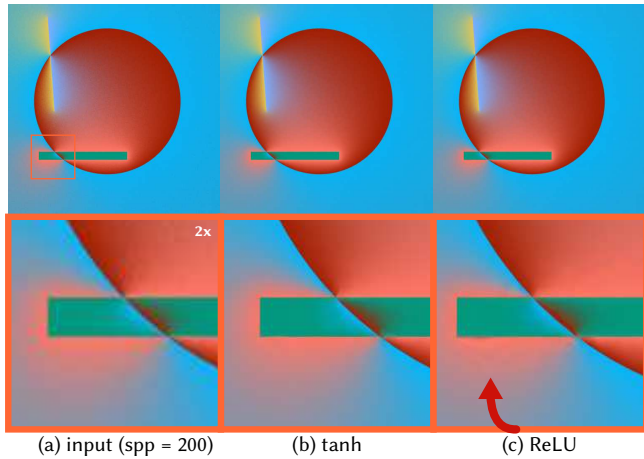| Methods	Denoising		Denoising + Super-resolution (2×)	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
Per-Vertex	30.758	0.0593	30.344	0.0747
Per-Edge	36.682	**0.0178**	35.518	**0.0361**
Ours	**44.486**	0.0261	**43.913**	0.0423

(a) input (spp = 200) (b) tanh (c) ReLU

Fig. 14. ReLU activations generate redundant discontinuities.

field is a hybrid feature field with features stored at vertex. We show and report the results of this per-vertex field as a reference in the main text. We ablate our design of discontinuous edges by introducing a per-edge field. This field interpolates the features stored on the two consecutive half-edges of the query face (purple arrows in Fig. 13). This interpolation is similar to our method but the feature of this configuration is continuous across any edges. Both versions are significantly worse than our model as reported in Table 2 and shown in our supplementary materials. Similar to the InstantNGP results, we observe that LPIPS is less sensitive to the absolute function values in the vicinity of discontinuities.

Activation. We choose tanh for activations since we observe that ReLU compete with discontinuous edges. As utilized by ReLU fields [Karnewar et al. 2022], ReLU activations tend to create steep slopes, serving an overlapping role as our discontinuous features. In Fig. 14, our neural field with ReLU activations is less smooth compared to the version with tanh activations. As we reply on the smoothness of MLP, we decide on tanh for activations.