EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

2D Neural Fields with Learned Discontinuities

Chenxi Liu[1], Siqi Wang[2], Matthew Fisher[3], Deepali Aneja[3], and Alec Jacobson[1,3]

[1]University of Toronto, Toronto, Canada
[2]New York University, New York, USA
[3]Adobe Research, San Francisco, USA

Abstract

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 discontinuity magnitudes as continuous variables and optimize. We further introduce a novel discontinuous neural field model that jointly approximates the target image and recovers discontinuities. Through systematic evaluations, our neural field outperforms other methods that fit unknown discontinuities with discontinuous representations, exceeding Field of Junction and Boundary Attention by over 11dB in both denoising and super-resolution tasks and achieving 3.5× smaller Chamfer distances than Mumford–Shah-based methods. It also surpasses InstantNGP with improvements of more than 5dB (denoising) and 10dB (super-resolution). 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;***

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 (e.g., occlusions in captured 3D scenes, layers in graphic designs, material boundaries), 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 using curves, fill boundaries, or layer overlaps, but these formats provide minimal support for continuous signals elsewhere (e.g., solid colors, basic gradients).

Recently, Belhe et al. [BGF*23] proposed storing images as the output of a small neural network fed with a spatially varying feature vector. That feature vector is carefully interpolated over a triangle mesh that is constructed to ensure discontinuities along certain edges. The weights of the neural network are optimized to reconstruct samples of the image function. This representation is very compact and 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 clearly indicate how to treat discontinuity locations as optimization variables. When discontinuties are missing — even partially — their reconstruction is noticeably poor (see Fig. 1).

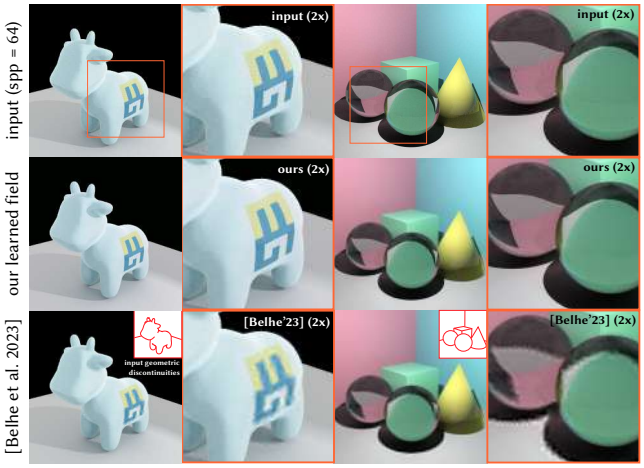


Figure 1: Discontinuity-aware 2D neural field [BGF*23] requires accurate 2D discontinuities as input. In their application of denoising 3D renderings, not all types of discontinuities are always available. False negatives caused by sharp texture and refracted geometries lead to blurs. Zoom scales indicate inference scale.

In this paper, we propose a non-trivial change to Belhe et al.'s method so image fitting no longer requires discontinuities to be known in advance. While our *problem statement* resembles that of

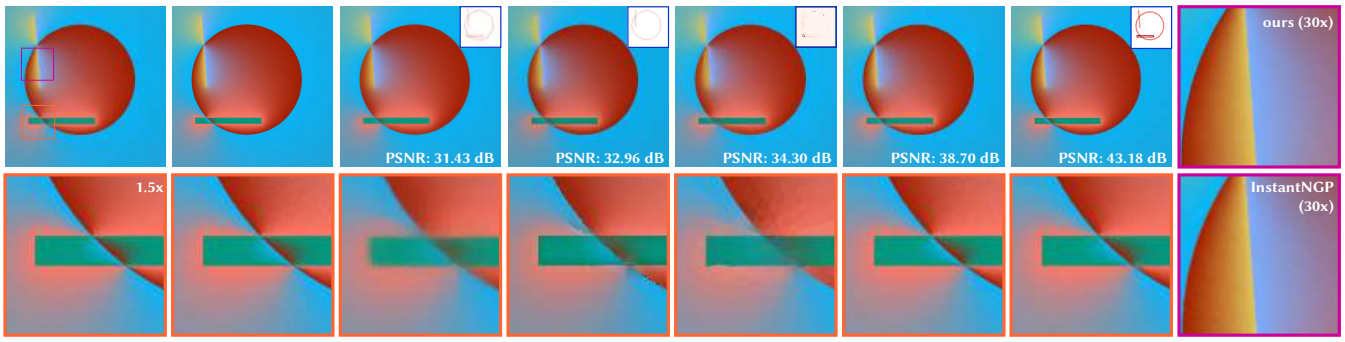


Figure 2: (a,b) Diffusion curves define an example harmonic function field with sharp discontinuities [OBW*08]. (c) Denoising method, Field of Junctions (FoJ) [VZ21], fails to recover clear discontinuities due to using constant-patch-based approximation. (d, e) MS functional based methods jointly approximate the target and detect discontinuities (see red edges in insets) [WLL22], similar to our method. However, both versions fail to achieve these goals due to limited function expressiveness. (f) Continuous neural fields, such as InstantNGP [MESK22], do not represent discontinuities, causing blurs when zoomed-in (h). (g) Our accurate approximation and recovered discontinuities.

Belhe et al., the absence of predefined discontinuities introduces challenges that we address with a significantly different *solution*: optimizing a novel function space that supports parametric discontinuities. During fitting, we treat *all* mesh edges as potential discontinuities, introducing variables to model the magnitude of the 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 convert almost-continuous edges into continuous edges, represented by fewer feature variables. This results in a field with improved storage efficiency and preserved fidelity.

We demonstrate that our neural fields with learned discontinuities directly support denoising and super-resolution. We compare to methods in the same category that fit unknown discontinuities with discontinuous representation: Field of Junction (FoJ) [VZ21], Boundary Attention [PHH*24], and Mumford–Shah based methods. Using a novel systematically synthesized diffusion curve dataset, we show that our method of matching size outperforms FoJ and Boundary Attention by $>$ 11dB (denoising and super-resolution), Mumford–Shah based methods by $>$ 8dB (denoising) and $>$ 7dB (super-resolution), as well as common continuous representation, InstantNGP [MESK22], by $>$ 5dB (denoising) and $>$ 10dB (super-resolution). Visually, our neural field maintains sharp region boundaries at large zoom levels (30$\times$ in Fig. 11,2,9), while InstantNGP blurs boundaries. Our method recovers more accurate discontinuities than Mumford-Shah-based denoising [WLL22]: 3.5$\times$ smaller Chamfer distance to the ground truth. 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. 11,13). We release our data and code here: `discontinuity2d.github.io`.

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 [BWG03; TC04; Sen04; RBW04; TC*05; PZ08; PK10; RL16]. Another approach represents digital images with discrete curves or region boundaries and smooth interior functions [TG22]. Diffusion curves [OBW*08] defines images as harmonic functions with curve Dirichlet boundary conditions, a space our method can accurately approximate (Section 6). Triangle mesh-based representations [DACB96; DDI06; SW04; TA11] and more advanced curve- and patch-based primitives [LL06; SLWS07; XLY09; LHM09; ZDZ17] 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 [XTS*22]. Early work by Song et al. [SWWW15] introduced coordinate neural networks for image encoding, a technique applicable to large image compression [MST*21; SMB*20; MGB*21]. 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 [CLW21; MLL*21; TLY*21; YLT*21; SGY*21; MESK22; CXG*22; TMN*23]. These offer reduced computation, better network capacity utilization, and explicit geometric representation. While most hybrid fields don't explicitly address discontinuities, recent meth-

(a) input (b) FoJ (c) [PHH*24] (d) ours

Figure 3: (a) Natural photos pose challenges for approximation. (b) FoJ [VZ21] with a hand-picked patch size $R = 4$ slightly blurs the input. (c) Its successor, Boundary Attention [PHH*24], significantly over-simplifies the input as it is trained for denoising rather than approximation. (d) Our method approximates the input with sharper edges.

ods like ReLU fields [KRWM22] attempt to approximate them with steep ReLUs, but still face blurring issues as shown by Belhe et al. [BGF*23]. Other approaches, like those by Reddy et al. [RZW*21], represent true discontinuities but are tailored for specific cases like fonts. Discontinuity-aware 2D neural fields [BGF*23], an inspiration for our approach, show promise but require user-provided discontinuous edges (Fig. 1). Our method fits unknown discontinuities and is compatible with discontinuity-aware 2D neural field offering efficient storage and inference.

Closest to our work, Field of Junctions (FoJ) [VZ21] uses a field of patches as their discontinuous representation with hand-picked hyperparameters, especially for denoising tasks. Its succeeding method, Boundary Attention, a concurrent work, uses supervised learning with highly-noisy training data to determine these hyperparameters for the same patch-based representation [PHH*24]. We compare to these two methods and show that their focus is on denoising rather than accurate approximation (Fig. 2c,3).

Differentiable rendering Addressing visual discontinuity in differentiable rendering is a persistent challenge [ZJL20; SZR*23]. Traditional automatic differentiations often overlook discontinuities' contributions to gradients [BMM*21]. Differentiable rendering techniques include smoothed rasterization [LB14; KUH18; LLCL19; LHK*20], Monte Carlo-based methods with boundary and interior sampling [LADL18; LLGR20; LHJ19; BLD20; ZRJ23], analytic solutions [BMM*21], and finite difference approaches [YBAF22; DHB24]. 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.

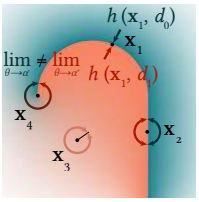
Mumford–Shah functional The Mumford–Shah (MS) functional, proposed for image segmentation [MS89], models images as piecewise-smooth functions with explicit discontinuities. The Ambrosio-Tortorelli approximation [AT90] made solving for MS functionals tractable with techniques like ADMM. This functional has been widely used in 2D image tasks [TYW01; VC02; LFP22] and applied to 3D surfaces [TT16; BCGL18; WLL22] and volumes [CFGL16]. Our method, similar to the MS functional, jointly recovers piecewise-smooth functions and discrete discontinuities. Unlike level-set-based methods [VC02; ES02], ours supports open boundaries and produces more accurate discontinuities free of narrow bands [TT16; BCGL18] and staircasing artifacts [TYW01; LFP22]. While Wang et al. [WLL22] also discretize discontinuities on mesh edges, their representation (constant colors per face) lacks the expressiveness of our neural model (Fig. 2).

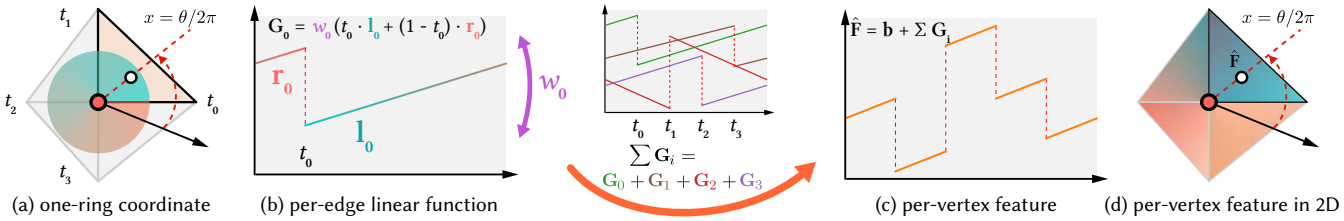
3. Preliminaries

We review the continuity criteria proposed by Belhe et al. [BGF*23]. 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 at locations with sharp changes in 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. [BGF*23]. 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 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 an angular coordinate $\alpha \in \mathbb{R}$. The *directional limit* of a function at a position $\mathbf{x}$ along a direction θ is defined as $h(\mathbf{x}, \theta) = \lim_{r \to 0^+} f(\mathcal{C}(r, \theta))$, where $\mathcal{C}(r, \theta)$ maps polar coordinate to Cartesian coordinate (see $\mathbf{x}_1$ in inset). A function is directionally discontinuous at $\mathbf{x}, \alpha$ if $\lim_{\theta \to \alpha^+} h(\mathbf{x}, \theta) \neq \lim_{\theta \to \alpha^-} h(\mathbf{x}, \theta)$.

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 $\mathbf{x} \in \partial\Gamma$





(a) one-ring coordinate (b) per-edge linear function (c) per-vertex feature (d) per-vertex feature in 2D

Figure 4: (a) Given a query position (white dot), per-vertex feature is constructed in the normalized angular coordinate centered at the corresponding vertex (red dot). (b) A potential discontinuity is positioned at each edge by defining a linear function $\mathbf{G}_i$. The function $\mathbf{G}_i$ is parameterized by continuous variables $(w, \mathbf{l}, \mathbf{r})$ with respect to a local coordinate t_i at this edge (indicated by the cyan-to-orange gradient). (c) The per-vertex feature $\hat{\mathbf{F}}$ is the sum of these per-edge function $\mathbf{G}_i$, potentially discontinuous, and a bias. (d) If all edges are associated with $w_i \neq 0$, this per-vertex feature function is discontinuous exactly across these edges.

(e.g., $\mathbf{x}_4$). Note that γ_i can be a curve [BGF*23]. For simplicity, we only consider the case where γ_i is a line segment and approximate curve geometries by adjusting the triangle density.

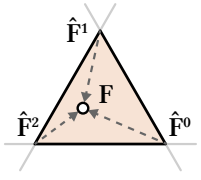
4. Method

Our neural field is built upon an underlying triangle mesh and discontinuous feature function defined locally within each vertex one-ring. We first analyze the local per-vertex feature function and then examine how feature variables are defined on the mesh (Section 4.1). This discontinuous feature function allows discontinuity across mesh edges from which we learn the target discontinuous edge set Γ. We approximate the target image by initializing a triangle mesh (Section 4.2.1), fitting a neural field with all edges potentially discontinuous (Section 4.2.2), and refining to enforce continuity on almost-continuous edges (Section 4.2.3).

4.1. Neural Field with Discontinuous Features

Given a triangle mesh in $\mathbb{R}^2$, we define a 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 continuous variables with the standard autodiff-gradient-based approach.

In a 2D domain Ω of 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

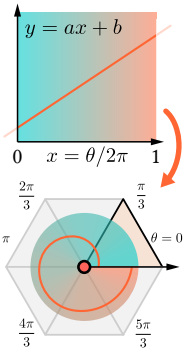


$$\mathbf{F}(\mathbf{x}) = (1 - \lambda_1 - \lambda_2)\hat{\mathbf{F}}_i^0(\mathbf{x}) + \lambda_1 \hat{\mathbf{F}}_i^1(\mathbf{x}) + \lambda_2 \hat{\mathbf{F}}_i^2(\mathbf{x}), \lambda_1, \lambda_2 \in [0, 1], \tag{1}$$

where $\hat{\mathbf{F}}_i$s are defined in its corresponding vertex one-ring $\mathcal{N}(\mathbf{v}_i)$. We parameterize the vertex one-ring using the polar coordinate system with angular coordinate normalized to $[0, 1)$. We introduce our discontinuous per-vertex feature in this local coordinate (Section 4.1.1) then return to the target 2D domain (Section 4.1.2).

4.1.1. Discontinuous feature in one-ring

Consider a simple 1D linear function $g(x) = ax + b$. We can restrict its domain to $[0, 1)$ and define $x = \theta/2\pi$ as the normalized angular coordinate in a vertex one-ring, where θ is the counterclockwise angle to an arbitrary reference edge (see the inset below).



In this way, we construct a discontinuity at $x = 0$ across the reference angle in the one-ring when $a \neq 0$. This construction enables standard autodiff to calculate the left and right derivatives of $g(x)$ at the discontinuity location. To construct the discontinuity at edges other than the reference one, we introduce an additional local coordinate $x_i = t_i(x) = \text{fmod}((\theta - \theta_i)/2\pi + 1, 1)$ centered at each *edge* e_i with angle θ_i in the one-ring. For a toy example, consider edge e_1 with angle $\theta_1 = \pi/3$. Along the reference edge e_0, we have $t_1(x) = \text{fmod}((0 - \pi/3)/2\pi + 1, 1) = 5/6$. We further define a linear function $\varphi_i(x_i) = a_i x_i + b_i$ per local coordinate. These functions form a basis for piecewise linear functions

$$g(x) = \sum_{e_i \in \mathcal{N}(\mathbf{v}_i)} \varphi_i(t_i(x)) = \sum_{e_i \in \mathcal{N}(\mathbf{v}_i)} (a_i t_i(x) + b_i), \tag{2}$$

which are discontinuous at the edge e_i if $a_i \neq 0$. Note that its discontinuous pieces all share the same slope of $\sum_{e_i \in \mathcal{N}(\mathbf{v}_i)} a_i$ (see Fig. 4a).

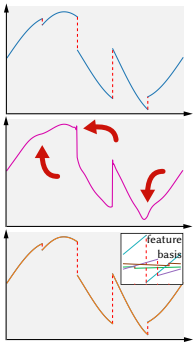
To improve the expressiveness, we extend features to $\mathbb{R}^k$

$$\begin{aligned}\hat{\mathbf{F}}(x) &= \mathbf{b} + \sum_{e_i \in \mathcal{N}(\mathbf{v}_i)} \mathbf{G}_i(t_i(x)) = \mathbf{b} + \sum_{e_i \in \mathcal{N}(\mathbf{v}_i)} w_i (t_i \cdot \mathbf{l}_i + (1 - t_i) \cdot \mathbf{r}_i) \\ &= \mathbf{b} + \sum_{e_i \in \mathcal{N}(\mathbf{v}_i)} (w_i(\mathbf{l}_i - \mathbf{r}_i)t_i + w_i \mathbf{r}_i), \end{aligned} \tag{3}$$

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 an edge e_i (Fig. 4b); 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. 4c).

We conduct a simple 1D fitting experiment to verify that our per-vertex feature can represent a discontinuous signal (inset). To fit the test non-linear signal, we pass $\hat{\mathbf{F}}(x)$ to a shallow MLP with two hidden layers of 64 neurons. The target function (blue) is a function of translated sine pieces and discontinuities at four points. For reference, we construct a 1D feature field (feature dimension

$k = 5$) with features defined at these four locations (equivalent to four edges) and linearly interpolated between them (magenta). We compare this reference to our 1D per-vertex feature ($k = 2$) with potential discontinuities at the matching locations (orange, feature basis functions are $\mathbf{G}_i$ in Eq. 3). In contrast to our faithful discontinuous 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.



4.1.2. Discontinuous field in 2D

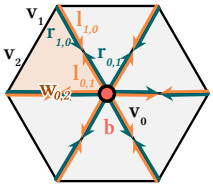
We return to the barycentric interpolation (Eq. 1) of per-vertex local features $\hat{\mathbf{F}}_i$ (Eq. 3) and describe how feature variables are defined on mesh. Given a vertex $\mathbf{v}_i$, the local feature is

$$\hat{\mathbf{F}}_i(\mathbf{x}) = \mathbf{b}_i + \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), i \neq j} \mathbf{F}_{i,j}(t_i^j(\mathbf{x})) \tag{4}$$

$$= \mathbf{b}_i + \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), i \neq j} w_{i,j} \left(t_i^j(\mathbf{x}) \cdot \mathbf{l}_{i,j} + (1 - t_i^j(\mathbf{x})) \cdot \mathbf{r}_{i,j} \right) \tag{5}$$

$$= \mathbf{b}_i + \sum_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i), i \neq j} \left(w_{i,j}(\mathbf{l}_{i,j} - \mathbf{r}_{i,j}) t_i^j(\mathbf{x}) + w_{i,j}\mathbf{r}_{i,j} \right). \tag{6}$$

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 $\mathbf{x} \in \Omega$ 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. 6 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 [BGF*23] when all adjacent edges are continuous. The feature $\mathbf{l}_{i,j}, \mathbf{r}_{i,j} \in \mathbb{R}^k$ and the discontinuity weight $w_{i,j} \in \mathbb{R}$ function similarly as the case of vertex one-ring. 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 adjacent vertices to have local "colors" while the discontinuity weight $w_{i,j} = w_{j,i}$ is shared between two vertices for consistent continuity behavior along an edge.



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

$$f(x) = \mathrm{MLP}(\mathbf{F}(x)). \tag{7}$$

Our neural fields use a shallow MLP: two hidden layers of 128 neurons with tanh activations. The inference algorithm is in Algorithm 1 where AngleCCW computes the counter-clockwise angle between two vectors.

4.1.3. Comparison to feature space in Belhe et al. [BGF*23]

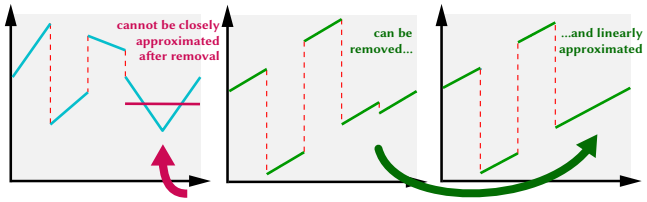
Belhe et al.'s per-vertex features are also defined in local polar coordinate

$$\hat{\mathbf{F}}_i = \mathbf{F}_i^{\mathrm{cw}} \frac{\theta_i^{\mathrm{ccw}}}{\theta_i^{\mathrm{cw}} + \theta_i^{\mathrm{ccw}}} + \mathbf{F}_i^{\mathrm{ccw}} \frac{\theta_i^{\mathrm{cw}}}{\theta_i^{\mathrm{cw}} + \theta_i^{\mathrm{ccw}}}. \tag{8}$$

This interpolation scheme (Eq. 2 [BGF*23]) finds the closest discontinuous edges clockwise ("cw") and counter-clockwise ("ccw"), then radially (based on θs) interpolates the half-edge features $\mathbf{F}_i^{\mathrm{cw}}, \mathbf{F}_i^{\mathrm{ccw}}$ (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 feature function has 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 $\mathbf{G}_i$ 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 to 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. In our ablations, our method manages to decrease the number of edges with features by 1/2 to 1/68 (Sec. A).

Algorithm 1: Field Inference

Function ***infer*** **(x, *M*)**
- $T, \lambda_1, \lambda_2 \leftarrow \mathrm{PointInFace}(\mathbf{x}, M)$;
- **forall** $\mathbf{v}_i$ *in* $T = (\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)$ **do**
  - **forall** e_i^j *in* $\mathit{AdjacentHalfEdge}(\mathbf{v}_i)$ **do**
    - $\theta_i^j \leftarrow \mathrm{AngleCCW}((\mathbf{x} - \mathbf{v}_i), e_i^j)$;
    - $t_i^j \leftarrow \mathrm{fmod}(\frac{\theta - \theta_i^j}{2\pi} + 1, 1)$;
  - **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. 6;
  - **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. 1 ;
  - **return** $\mathrm{MLP}(\mathbf{F}(\mathbf{x}))$;
- **end**



(a) [Belhe et al. 2023] (b) our feature (c) after feature removal

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

The disadvantage of having constant piecewise slopes is mitigated by the MLP as experimentally demonstrated in Section 6. Furthermore, our feature and interpolation scheme satisfy the continuity criteria proposed by Belhe et al. (Section 4). We show this by drawing a connection between our edges and the ones of Belhe et al.'s: since our w is shared per edge between the two adjacent vertices, the continuity of our field is consistent along edges, making our edges equivalent to Belhe et al.'s in the continuity criteria proof. 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 de-

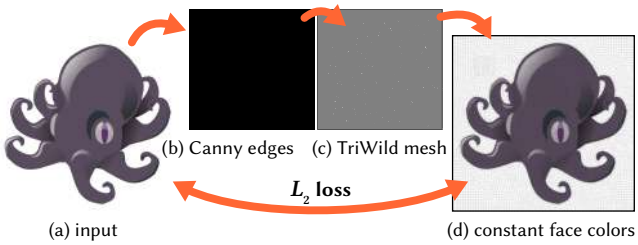


Figure 6: We initialize by triangulating Canny edges [Can86] with TriWild [HSG*19], then deforming and remeshing interleavedly. The deformation is posed as per-face constant color approximation.

fine 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.

4.2. Learning Discontinuous 2D Neural Field

With our discontinuous 2D neural field defined, we now describe the procedure for fitting it to a raster input. The procedure first initializes a triangle mesh that is approximately aligned with the target discontinuities (Section 4.2.1). Our method then optimizes the appearance of the field while more accurately aligning the discontinuities (Section 4.2.2). Finally, almost-continuous edges are discarded to enforce continuous areas and improve representation efficiency (Section 4.2.3).

4.2.1. 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 4.2.2). 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 [Can86]. We then connect positive pixels to their corresponding 8-neighbors and apply TriWild [HSG*19] 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

$$\min_M \int_\Omega \|f_0(\mathbf{x};M) - I(\mathbf{x})\|^2 \, d\mathbf{x} + \lambda_{\text{boundary}} \frac{1}{|\partial M|} \sum_{\mathbf{v}\in\partial M} \left\|\mathbf{v} - \mathbf{v}^0\right\|^2, \tag{9}$$

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 the first term by stratified sampling triangles and compute gradients using SoftRasterizer [LLCL19], which can be replaced by other differentiable renderers. We interleave the deformation with optional remeshing steps. See App. B for details.

4.2.2. Field optimization

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

$$L = \int_\Omega \|f(\mathbf{x};V,\Theta) - I(\mathbf{x})\|^2 \, d\mathbf{x} + \lambda_{\text{discont}} \int_E \|w(\mathbf{l}-\mathbf{r})\|_1 \, d\mathbf{x}, \tag{10}$$

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 $\sum_{e_{i,j}\in H} \left\|e_{i,j}\right\| \cdot \left\|w_{i,j}(\mathbf{l}_{i,j} - \mathbf{r}_{i,j})\right\|_1$, where $e_{i,j} \in H$ is a half-edge with length $\left\|e_{i,j}\right\|$, 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 [LADL18]. We observe that the discontinuity indicator $D = \max(w(\mathbf{l}-\mathbf{r}))$ 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. Therefore, we extend the important edge set to edges with $D > \beta$ and their adjacent edges. We sample these important edges with $5\times$ probabilities than other edges. For detailed gradient formulas, see Appendix B.

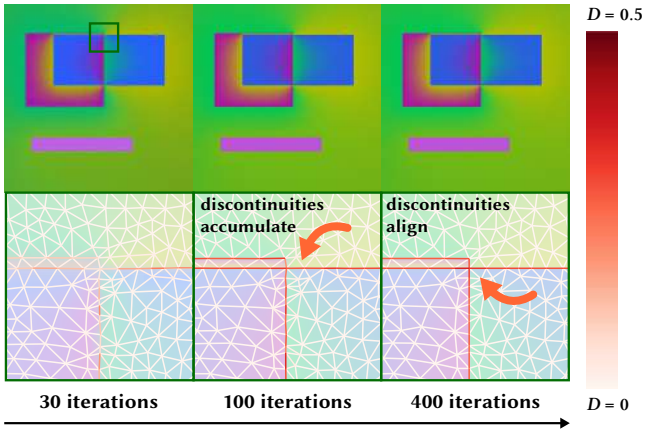


Figure 7: We jointly optimize our field and its underlying mesh. (a) In the early epochs, our indicated discontinuities begin to accumulate around the target discontinuities. (b) As optimization progresses, our method produces close interior color approximation and simultaneously aligns discontinuities.

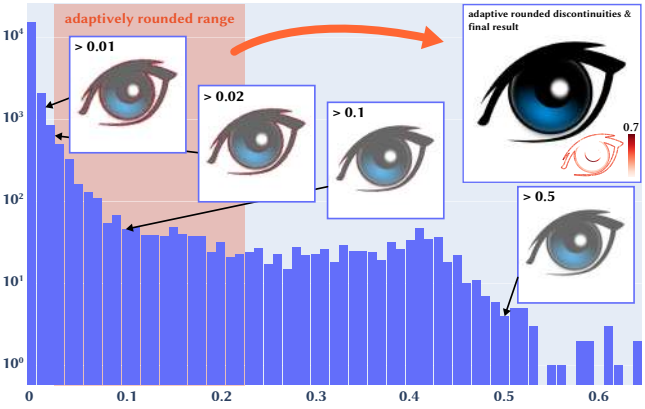


Figure 8: *Example log histogram of discontinuity indicator D.* The majority of edges are identified as continuous, allowing for efficient optimization and storage of our neural field. We apply an adaptive greedily rounding strategy to flexibly determine the final discontinuous edges.

4.2.3. Rounding

Once our neural field sufficiently approximates the input, we retain almost-continuous edges but remove their associated features. We refer to this step as "rounding", a term borrowed from integer programming [CCZ*14]. After rounding, our neural field is enforced to be continuous across the affected edges. 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 similarly to classic mesh simplification. 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 $0.5 \cdot w(\mathbf{l}-\mathbf{r}) + w\mathbf{r}$ to the center vertex bias. Thanks to the locality of our features, this greedy rounding can be done efficiently. After the rounding step, we continue refining this new rounded neural field and the mesh with more iterations.

5. Implementation

We implemented our method in PyTorch [PGM*19]. To avoid flipped triangles caused by sparse gradients, we apply the large-step precondition [NJJ21] for mesh initialization and field optimization, with weights of 1, 0.5 respectively. We use ADAM [KB15] 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 $\text{spp} = 2^2$, 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 affects the 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.

Table 1: Quantitative measures for denoising tasks.

Methods	Denoising		Denoising + Super-Resolution (2×)		Denoising JPEG Vector (Fig. 10, 15b)	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
Per-Vertex	30.758	0.0593	30.344	0.0747	25.321	0.0508
InstantNGP	39.016	0.0431	32.715	**0.0261**	33.027	0.0159
[BGF*23]+Canny	37.027	0.0316	35.453	0.0484	27.352	0.0367
MS Pixel	35.563	0.0319	-	-	26.568	0.0696
MS Mesh	36.159	0.0321	36.118	0.0466	27.733	0.0151
FoJ	32.756	0.0451	32.073	0.0560	20.683	0.1157
[PHH*24]	21.076	0.0689	-	-	11.412	0.1082
Ours	**44.486**	**0.0261**	**43.913**	0.0423	**33.168**	**0.0086**

Table 2: Quantitative measures for discontinuity approximation task.

Methods	Approximating Drawings (Fig. 13, 15a)		Approximating Photos (Fig. 3, 15d)	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
[BGF*23]+Canny	38.314	0.0337	33.064	**0.0853**
MS Pixel	28.440	0.1242	24.134	0.3504
MS Mesh	32.179	0.0472	21.863	0.1649
FoJ	31.715	0.0898	25.860	0.3585
[PHH*24]	17.607	0.1542	21.221	0.3352
Ours	**42.614**	**0.0228**	**34.009**	0.1119

6. Evaluation

We evaluate our method with various applications: denoising, super-resolution, approximation, and segmentation. 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. 1), human-drawn artistic images, natural images, and spatial data (depth maps). We evaluate our method against trivially applying Belhe et al.'s method with Canny edges as discontinuities, and methods that fit discontinuous representation without priors: MS based methods [WLL22], FoJ [VZ21], and Boundary Attention [PHH*24]. Additionally, we compare to continuous neural fields, represented by a simple reference feature field with per-vertex features and InstantNGP [MESK22], demonstrating that discontinuity representation is necessary for these tasks. We show additional results in our gallery (Fig. 15). See Appendix D for setup details.

6.1. Denoising

Noise in images comes from many sources. We test on diffusion curves with noisy Monte Carlo samples rendered using walk-on-spheres (WoS) method [Mul56; SC20]. We randomly generate 40 diffusion curves containing line segments, rectangles, and circles. Half of these only contain integer-coordinate rectangles and thus

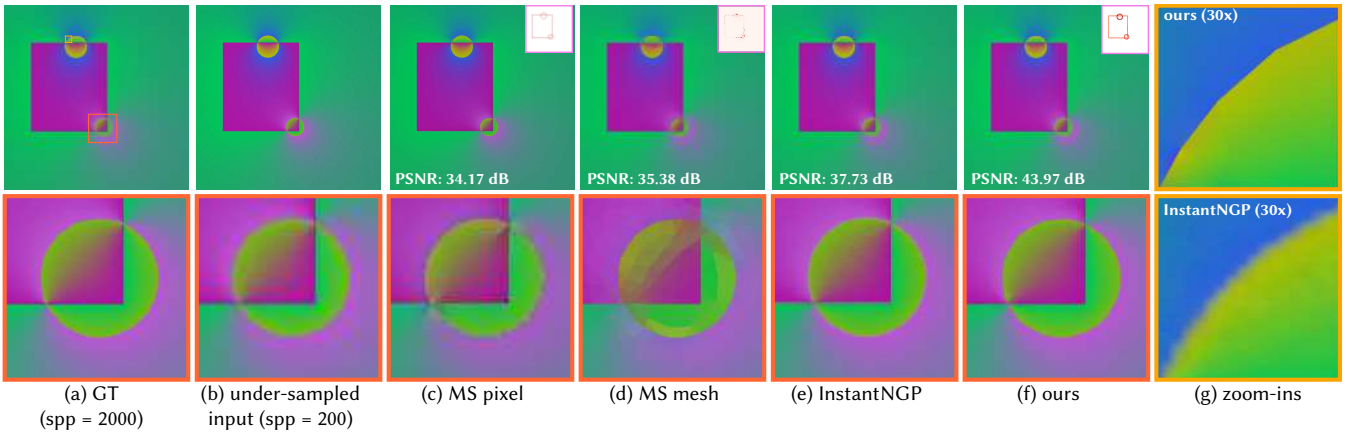


Figure 9: We evaluate our model on randomly generated diffusion curves and assess the results for denoising effects and image quality under zoom, compared against continuous neural fields, e.g., InstantNGP [MESK22], and MS functional based methods [WLL22].

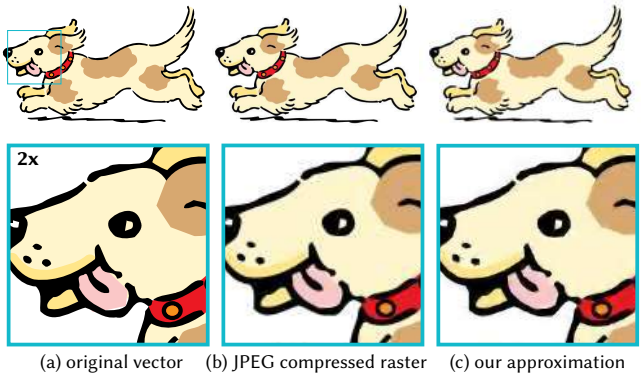


Figure 10: We verify that our neural field can approximate simple constant color fills and gradients under JPEG compression.

have 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. 2, 9, our field simultaneously approximates the target harmonic functions and denoises thanks to the smoothness bias of MLP, while the comparison methods either struggle to approximate or over-fit 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 Canny edges, discontinuous *edges* detected by mesh-based MS [WLL22], those detected by our method, and the ground truth diffusion curve *geometries*. Canny edges detected with the same parameters as in our fitting pipeline are 5.5$\times$ further than ours to the GT (Chamfer: 0.900 vs. 0.165). Mesh-based MS produces discontinuous edges that are 3.5$\times$ Chamfer distance away (0.580) from the GT compared to our results.

Additionally, we qualitatively evaluate our method with the task of denoising JPEG-compressed vector images (constant fills in Fig. 10 and human-created gradients in Fig. 15b). Our method not only approximates these two types of functions but also reduces JPEG compression artifacts without prior knowledge about the task.

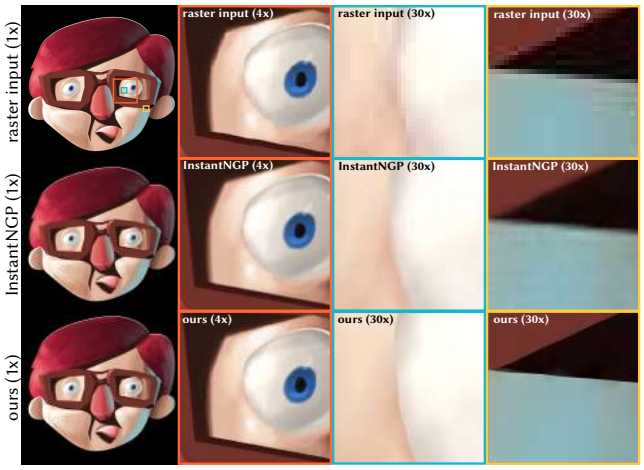


Figure 11: Our neural field supports edge-preserving super-resolution while continuous neural field, such as InstantNGP [MESK22], blurs region boundaries under close view.

6.2. Super-Resolution

We qualitatively show zoom-ins (1.5$\times$, 2$\times$, 4$\times$, 30$\times$) of the approximations across this paper (Fig. 1, 2, 3, 9, 11, 15). As presented by the previous work [BGF*23] and in Fig. 2, 9, 11, 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 inferred under 2$\times$ original resolutions (excluding Boundary Attention [PHH*24] due to implementation difficulties). We measure against the clean diffusion curve images with

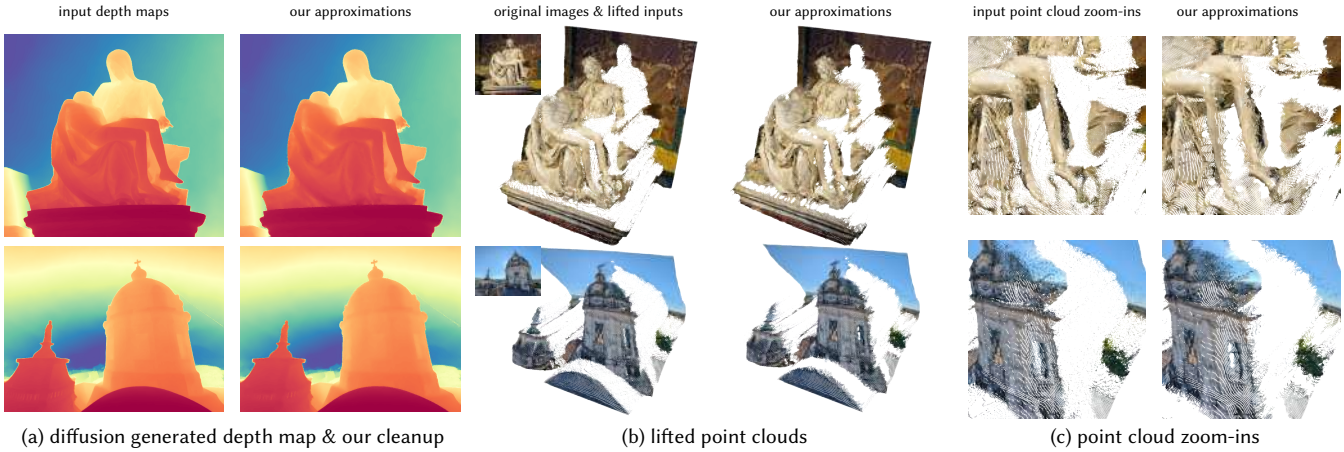


Figure 12: *Segmentation of diffusion-generated depth data.* (a) Depth map generated by diffusion model [KOH*23] is corrupted by model-induced noises, particularly around discontinuities. (b) Our neural fields closely approximate input data while reducing floaters and yielding clean cuts between depth discontinuities. See (c) for point cloud zoom-ins.

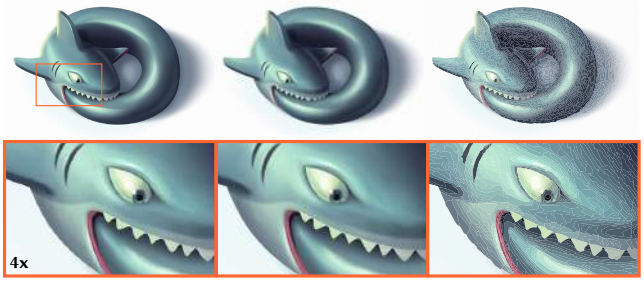


Figure 13: *Approximation of artistic images.* Our method fits resolution-independent fields to artistic images and qualitatively outperforms common vectorization methods in terms of approximation and representation. We manually select the Adobe Illustrator Image Trace parameters to achieve the best approximation.

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.

6.3. Approximation

Apart from the synthetic images, we approximate artistic drawings and natural images. This approximation process can be considered comparable to vectorization, as the resulting discontinuous neural fields are resolution-independent, a key property of vector images. In Fig. 13, 15a, 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. 14) — an effect hard to achieve with vectorization tools. While our method does not generate outputs in standard vector graphics formats like SVG and lacks an editing tool, developing one for our hybrid neural field, based on triangle meshes, is an interesting and feasible direction to enhance the expressiveness of traditional vector graphics formats.

We stress test our method on natural images. We show in Fig. 3, 15d two successful approximations with mild denoising effects,

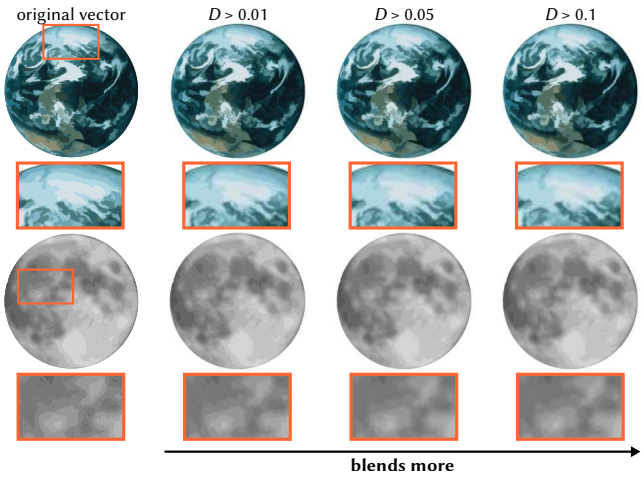
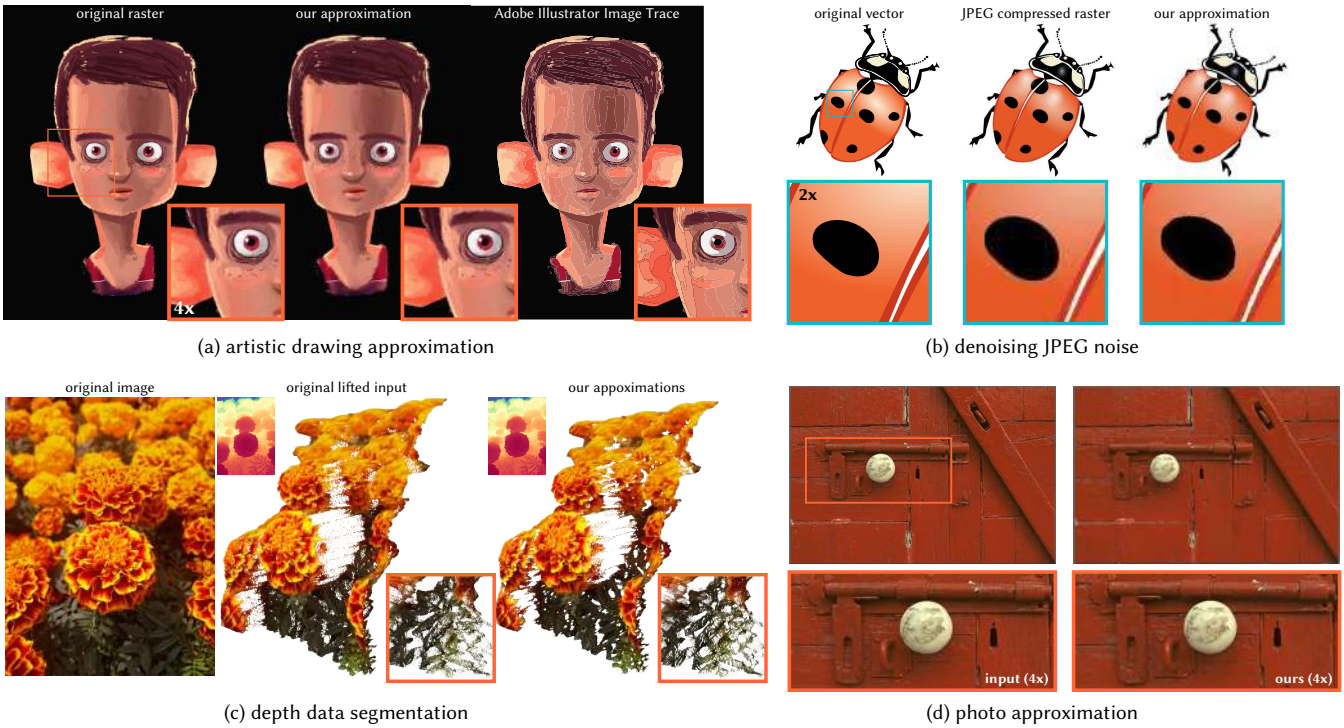


Figure 14: *Blending posterized vector images.* Vectorization using only solid color fills generates posterized vector images. Our method can be applied to blend solid fills and create a resolution-independent image with color gradients (highlighted in zoom-ins).



(a) artistic drawing approximation

(b) denoising JPEG noise

(c) depth data segmentation

(d) photo approximation

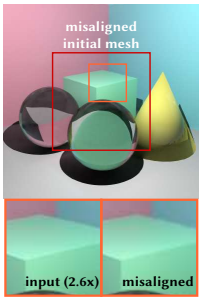
Figure 15: Additional results for all our tasks: (a,d) approximation, (b) denoising, (c) segmentation.

similar to the edge-preserving bilateral filter. Quantitative measures are given in Table 2.

6.4. 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 [RKH*21]. We evaluate on diffusion-generated depth maps [KOH*23] (Fig. 12, 15c). 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 and reducing floaters.

7. 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. 13). Additionally, our fitting procedure relies on the initial mesh to be reasonably aligned with the target discontinuities (Section 4.2.1). 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 as well as exploring more flexible underlying structures, then adapting our proposed feature functions to these structures could be interesting future directions. To analyze the importance of a reasonably aligned initial mesh, we experiment with Fig. 1 (right) by removing the Canny edges within a square of side length 0.5 times the width, centered on the canvas (inset). Although our approach can still partially recover clear discontinuities, the sharp discontinuities within the center square appear blurred in some locations, primarily because the initial edges are too distant or an insufficient density of initial edges.

The approximation constructed by our method has its limitations. While our method performs well on synthetic images or drawings with cleaner and sparser discontinuities, its performance on natural images is less accurate, resembling the behavior of an edge-preserving bilateral filter. This highlights a common trade-off between accurately capturing discontinuities and achieving effective denoising. Additionally, our use of 2×2 subpixel stratified sampling with nearest-neighbor interpolation during optimization is generally sufficient for most raster inputs but can introduce aliasing artifacts in some cases, such as the left example in Fig. 1. Employing a softer interpolation filter could potentially mitigate these artifacts and improve overall performance.

Our field is single-level compared to neural fields with multi-level feature grids, such as InstantNGP [MESK22]. This design provides denoising power and compressed field size — the sizes of our fields commonly match those of InstantNGPs with the 2^{14} hash table size (compared to their default 2^{24}). Despite these benefits, efficient representation of high-frequency details, which can

remain homogeneous within a continuous region, is a relevant open question.

8. 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. [BGF*23]. 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 [MESK22], 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 approximation of artistic drawings and photos and the cleanup of diffusion-generated depth data.

References

[Ale19] ALEXA, MARC. "Harmonic triangulations". *ACM Transactions on Graphics (TOG)* 38.4 (2019), 1–14 14.

[AT90] AMBROSIO, LUIGI and TORTORELLI, VINCENZO MARIA. "Approximation of functional depending on jumps by elliptic functional via Γ-convergence". *Communications on Pure and Applied Mathematics* 43.8 (1990), 999–1036 3.

[BCGL18] BONNEEL, NICOLAS, COEURJOLLY, DAVID, GUETH, PIERRE, and LACHAUD, JACQUES-OLIVIER. "Mumford-Shah Mesh Processing using the Ambrosio-Tortorelli Functional". *Computer Graphics Forum*. Vol. 37. 7. Wiley Online Library. 2018, 75–85 3.

[BGF*23] BELHE, YASH, GHARBI, MICHAËL, FISHER, MATTHEW, et al. "Discontinuity-aware 2D neural fields". *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 41.6 (2023). DOI: 10.1145/3550454.3555484 1, 3–5, 7, 8, 11.

[BLD20] BANGARU, SAI PRAVEEN, LI, TZU-MAO, and DURAND, FRÉDO. "Unbiased warped-area sampling for differentiable rendering". *ACM Transactions on Graphics (TOG)* 39.6 (2020), 1–18 3.

[BMM*21] BANGARU, SAI PRAVEEN, MICHEL, JESSE, MU, KEVIN, et al. "Systematically differentiating parametric discontinuities". *ACM Transactions on Graphics (TOG)* 40.4 (2021), 1–18 3.

[BWG03] BALA, KAVITA, WALTER, BRUCE, and GREENBERG, DONALD P. "Combining edges and points for interactive high-quality rendering". *ACM Transactions on Graphics (TOG)* 22.3 (2003), 631–640 2.

[Can86] CANNY, JOHN. "A computational approach to edge detection". *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), 679–698 6.

[CCZ*14] CONFORTI, MICHELE, CORNUÉJOLS, GÉRARD, ZAMBELLI, GIACOMO, et al. *Integer programming models*. Springer, 2014 7.

[CFGL16] COEURJOLLY, DAVID, FOARE, MARION, GUETH, PIERRE, and LACHAUD, JACQUES-OLIVIER. "Piecewise smooth reconstruction of normal vector field on digital data". *Computer Graphics Forum*. Vol. 35. 7. Wiley Online Library. 2016, 157–167 3.

[CLW21] CHEN, YINBO, LIU, SIFEI, and WANG, XIAOLONG. "Learning continuous image representation with local implicit image function". *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, 8628–8638 2.

[CXG*22] CHEN, ANPEI, XU, ZEXIANG, GEIGER, ANDREAS, et al. "Tensorf: Tensorial radiance fields". *European Conference on Computer Vision*. Springer. 2022, 333–350 2.

[DACB96] DAVOINE, FRANCK, ANTONINI, MARC, CHASSERY, J-M, and BARLAUD, MICHEL. "Fractal image compression based on Delaunay triangulation and vector quantization". *IEEE Transactions on Image Processing* 5.2 (1996), 338–346 2.

[DDI06] DEMARET, LAURENT, DYN, NIRA, and ISKE, ARMIN. "Image compression by linear splines over adaptive triangulations". *Signal Processing* 86.7 (2006), 1604–1616 2.

[DHB24] DELIOT, THOMAS, HEITZ, ERIC, and BELCOUR, LAURENT. "Transforming a Non-Differentiable Rasterizer into a Differentiable One with Stochastic Gradient Estimation". *arXiv preprint arXiv:2404.09758* (2024) 3.

[ES02] ESEDOGLU, SELIM and SHEN, JIANHONG. "Digital inpainting based on the Mumford–Shah–Euler image model". *European Journal of Applied Mathematics* 13.4 (2002), 353–370 3.

[HSG*19] HU, YIXIN, SCHNEIDER, TESEO, GAO, XIFENG, et al. "TriWild: robust triangulation with curve constraints". *ACM Transactions on Graphics (TOG)* 38.4 (2019), 1–15 6.

[Ima23] IMAGEMAGICK. *ImageMagick*. Version 7.1.1-9. May 21, 2023. URL: https://imagemagick.org 14.

[Ink21] INKSCAPE. *Inkscape*. Version 1.1.1. Sept. 22, 2021. URL: https://inkscape.org/ 14.

[KB15] KINGMA, DIEDERIK and BA, JIMMY. "Adam: A Method for Stochastic Optimization". *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA, 2015 7.

[KOH*23] KE, BINGXIN, OBUKHOV, ANTON, HUANG, SHENGYU, et al. "Repurposing diffusion-based image generators for monocular depth estimation". *arXiv preprint arXiv:2312.02145* (2023) 9, 10.

[KRWM22] KARNEWAR, ANIMESH, RITSCHEL, TOBIAS, WANG, OLIVER, and MITRA, NILOY. "ReLU fields: The little non-linearity that could". *ACM SIGGRAPH 2022 Conference Proceedings*. 2022, 1–9 3, 13.

[KUH18] KATO, HIROHARU, USHIKU, YOSHITAKA, and HARADA, TATSUYA. "Neural 3d mesh renderer". *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, 3907–3916 3.

[LADL18] LI, TZU-MAO, AITTALA, MIIKA, DURAND, FRÉDO, and LEHTINEN, JAAKKO. "Differentiable monte carlo ray tracing through edge sampling". *ACM Transactions on Graphics (TOG)* 37.6 (2018), 1–11 3, 6, 14.

[LB14] LOPER, MATTHEW M and BLACK, MICHAEL J. "OpenDR: An approximate differentiable renderer". *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*. Springer. 2014, 154–169 3.

[LFP22] LE, HOANG TRIEU VY, FOARE, MARION, and PUSTELNIK, NELLY. "Proximal based strategies for solving Discrete Mumford-Shah with Ambrosio-Tortorelli penalization on edges". *IEEE Signal Processing Letters* 29 (2022), 952–956 3.

[LHJ19] LOUBET, GUILLAUME, HOLZSCHUCH, NICOLAS, and JAKOB, WENZEL. "Reparameterizing discontinuous integrands for differentiable rendering". *ACM Transactions on Graphics (TOG)* 38.6 (2019), 1–14 3.

[LHK*20] LAINE, SAMULI, HELLSTEN, JANNE, KARRAS, TERO, et al. "Modular primitives for high-performance differentiable rendering". *ACM Transactions on Graphics (ToG)* 39.6 (2020), 1–14 3.

[LHM09] LAI, YU-KUN, HU, SHI-MIN, and MARTIN, RALPH R. "Automatic and topology-preserving gradient mesh generation for image vectorization". *ACM Transactions on Graphics (TOG)* 28.3 (2009), 1–8 2.

[LL06] LECOT, GREGORY and LEVY, BRUNO. "Ardeco: Automatic region detection and conversion". *17th Eurographics Symposium on Rendering-EGSR'06*. 2006, 349–360 2.

[LLCL19] LIU, SHICHEN, LI, TIANYE, CHEN, WEIKAI, and LI, HAO. "Soft rasterizer: A differentiable renderer for image-based 3d reasoning". *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, 7708–7717 3, 6.

[LLGR20] LI, TZU-MAO, LUKÁČ, MICHAL, GHARBI, MICHAËL, and RAGAN-KELLEY, JONATHAN. "Differentiable vector graphics rasterization for editing and learning". *ACM Transactions on Graphics (TOG)* 39.6 (2020), 1–15 3.

[MESK22] MÜLLER, THOMAS, EVANS, ALEX, SCHIED, CHRISTOPH, and KELLER, ALEXANDER. "Instant neural graphics primitives with a multiresolution hash encoding". *ACM transactions on graphics (TOG)* 41.4 (2022), 1–15 2, 7, 8, 10, 11.

[MGB*21] MEHTA, ISHIT, GHARBI, MICHAËL, BARNES, CONNELLY, et al. "Modulated periodic activations for generalizable local functional representations". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 14214–14223 2.

[MLL*21] MARTEL, JULIEN NP, LINDELL, DAVID B, LIN, CONNOR Z, et al. "Acorn: Adaptive coordinate networks for neural scene representation". *arXiv preprint arXiv:2105.02788* (2021) 2.

[MS89] MUMFORD, DAVID BRYANT and SHAH, JAYANT. "Optimal approximations by piecewise smooth functions and associated variational problems". *Communications on pure and applied mathematics* (1989) 3.

[MST*21] MILDENHALL, BEN, SRINIVASAN, PRATUL P, TANCIK, MATTHEW, et al. "NeRF: Representing scenes as neural radiance fields for view synthesis". *Communications of the ACM* 65.1 (2021), 99–106 2.

[Mül21] MÜLLER, THOMAS. *tiny-cuda-nn*. Version 1.7. Apr. 2021. URL: https://github.com/NVlabs/tiny-cuda-nn 14.

[Mul56] MULLER, MERVIN E. "Some continuous Monte Carlo methods for the Dirichlet problem". *The Annals of Mathematical Statistics* (1956), 569–589 7.

[NJJ21] NICOLET, BAPTISTE, JACOBSON, ALEC, and JAKOB, WENZEL. "Large steps in inverse rendering of geometry". *ACM Transactions on Graphics (TOG)* 40.6 (2021), 1–13 7.

[OBW*08] ORZAN, ALEXANDRINA, BOUSSEAU, ADRIEN, WINNEMÖLLER, HOLGER, et al. "Diffusion curves: a vector representation for smooth-shaded images". *ACM Transactions on Graphics (TOG)* 27.3 (2008), 1–8 2.

[Pal22] PALFINGER, WERNER. "Continuous remeshing for inverse rendering". *Computer Animation and Virtual Worlds* 33.5 (2022), e2101 14.

[PGM*19] PASZKE, ADAM, GROSS, SAM, MASSA, FRANCISCO, et al. "Pytorch: An imperative style, high-performance deep learning library". *Advances in neural information processing systems* 32 (2019) 7.

[PHH*24] POLANSKY, MIA GAIA, HERRMANN, CHARLES, HUR, JUNHWA, et al. *Boundary Attention: Learning to Localize Boundaries under High Noise*. 2024. arXiv: 2401.00935 [cs.CV] 2, 3, 7, 8.

[PK10] PAVIĆ, DARKO and KOBBELT, LEIF. "Two-Colored Pixels". *Computer Graphics Forum*. Vol. 29. 2. Wiley Online Library. 2010, 743–752 2.

[PZ08] PARILOV, EVGUENI and ZORIN, DENIS. "Real-time rendering of textures with feature curves". *ACM Transactions on Graphics (TOG)* 27.1 (2008), 1–15 2.

[RBW04] RAMANARAYANAN, GANESH, BALA, KAVITA, and WALTER, BRUCE. *Feature-based textures*. Tech. rep. Cornell University, 2004 2.

[RKH*21] RADFORD, ALEC, KIM, JONG WOOK, HALLACY, CHRIS, et al. "Learning transferable visual models from natural language supervision". *International conference on machine learning*. PMLR. 2021, 8748–8763 10.

[RL16] RESHETOV, ALEXANDER and LUEBKE, DAVID. "Infinite resolution textures." *High Performance Graphics*. 2016, 139–150 2.

[RZW*21] REDDY, PRADYUMNA, ZHANG, ZHIFEI, WANG, ZHAOWEN, et al. "A multi-implicit neural representation for fonts". *Advances in Neural Information Processing Systems* 34 (2021), 12637–12647 3.

[SC20] SAWHNEY, ROHAN and CRANE, KEENAN. "Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains". *ACM Transactions on Graphics* 39.4 (2020) 7.

[Sen04] SEN, PRADEEP. "Silhouette maps for improved texture magnification". *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. 2004, 65–73 2.

[SGY*21] SHEN, TIANCHANG, GAO, JUN, YIN, KANGXUE, et al. "Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis". *Advances in Neural Information Processing Systems* 34 (2021), 6087–6101 2.

[SLWS07] SUN, JIAN, LIANG, LIN, WEN, FANG, and SHUM, HEUNG-YEUNG. "Image vectorization using optimized gradient meshes". *ACM Transactions on Graphics (TOG)* 26.3 (2007), 11–es 2.

[SMB*20] SITZMANN, VINCENT, MARTEL, JULIEN, BERGMAN, ALEXANDER, et al. "Implicit neural representations with periodic activation functions". *Advances in neural information processing systems* 33 (2020), 7462–7473 2.

[SW04] SU, DAN and WILLIS, PHILIP. "Image interpolation by pixel-level data-dependent triangulation". *Computer graphics forum*. Vol. 23. 2. Wiley Online Library. 2004, 189–201 2.

[SWWW15] SONG, YING, WANG, JIAPING, WEI, LI-YI, and WANG, WENCHENG. "Vector regression functions for texture compression". *ACM Transactions on Graphics (TOG)* 35.1 (2015), 1–10 2.

[SZR*23] SPIELBERG, ANDREW, ZHONG, FANGCHENG, REMATAS, KONSTANTINOS, et al. "Differentiable visual computing for inverse problems and machine learning". *Nature Machine Intelligence* 5.11 (2023), 1189–1199 3.

[TA11] TU, XI and ADAMS, MICHAEL D. "Image representation using triangle meshes with explicit discontinuities". *Proceedings of 2011 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*. IEEE. 2011, 97–101 2.

[TC*05] TARINI, MARCO, CIGNONI, PAOLO, et al. "Pinchmaps: Textures with customizable discontinuities". *Computer Graphics Forum*. Vol. 24. 2005, 557–568 2.

[TC04] TUMBLIN, JACK and CHOUDHURY, PRASUN. "Bixels: Picture samples with sharp embedded boundaries." *Rendering Techniques* 255 (2004), 264 2.

[TG22] TIAN, XINGZE and GÜNTHER, TOBIAS. "A survey of smooth vector graphics: Recent advances in representation, creation, rasterization and image vectorization". *IEEE Transactions on Visualization and Computer Graphics* (2022) 2.

[TLY*21] TAKIKAWA, TOWAKI, LITALIEN, JOEY, YIN, KANGXUE, et al. "Neural geometric level of detail: Real-time rendering with implicit 3d shapes". *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, 11358–11367 2.

[TMN*23] TAKIKAWA, TOWAKI, MÜLLER, THOMAS, NIMIER-DAVID, MERLIN, et al. "Compact Neural Graphics Primitives with Learned Hash Probing". *SIGGRAPH Asia 2023 Conference Papers*. 2023, 1–10 2.

[TT16] TONG, WEIHUA and TAI, XUECHENG. "A variational approach for detecting feature lines on meshes". *Journal of Computational Mathematics* (2016), 87–112 3.

[TYW01] TSAI, ANDY, YEZZI, ANTHONY, and WILLSKY, ALAN S. "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 3.

[VC02] VESE, LUMINITA A and CHAN, TONY F. "A multiphase level set framework for image segmentation using the Mumford and Shah model". *International journal of computer vision* 50 (2002), 271–293 3.

[VZ21] VERBIN, DOR and ZICKLER, TODD. "Field of Junctions: Extracting Boundary Structure at Low SNR". *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2021, 6869–6878 2, 3, 7, 14.

[WLL22] WANG, CHUNXUE, LIU, ZHENG, and LIU, LIGANG. "Feature-preserving Mumford–Shah mesh processing via nonsmooth nonconvex regularization". *Computers & Graphics* 106 (2022), 222–236 2, 3, 7, 8, 14.

[XLY09] XIA, TIAN, LIAO, BINBIN, and YU, YIZHOU. "Patch-based image vectorization with automatic curvilinear feature alignment". *ACM Transactions on Graphics (TOG)* 28.5 (2009), 1–10 2.

[XTS*22] XIE, YIHENG, TAKIKAWA, TOWAKI, SAITO, SHUNSUKE, et al. "Neural fields in visual computing and beyond". *Computer Graphics Forum*. Vol. 41. 2. Wiley Online Library. 2022, 641–676 2.

[YBAF22] YANG, YUTING, BARNES, CONNELLY, ADAMS, ANDREW, and FINKELSTEIN, ADAM. "A δ: autodiff for discontinuous programs-applied to shaders". *ACM Transactions on Graphics (TOG)* 41.4 (2022), 1–24 3.

[YLT*21] YU, ALEX, LI, RUILONG, TANCIK, MATTHEW, et al. "Plenoctrees for real-time rendering of neural radiance fields". *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, 5752–5761 2.

[ZDZ17] ZHAO, SHUANG, DURAND, FRÉDO, and ZHENG, CHANGXI. "Inverse diffusion curves using shape optimization". *IEEE transactions on visualization and computer graphics* 24.7 (2017), 2153–2166 2.

[ZJL20] ZHAO, SHUANG, JAKOB, WENZEL, and LI, TZU-MAO. "Physics-based differentiable rendering: a comprehensive introduction". *ACM SIGGRAPH 2020 Courses* 14 (2020), 1–14 3.

[ZRJ23] ZHANG, ZIYI, ROUSSEL, NICOLAS, and JAKOB, WENZEL. "Projective Sampling for Differentiable Rendering of Geometry". *ACM Transactions on Graphics (TOG)* 42.6 (2023), 1–14 3.

Appendix A: Ablations

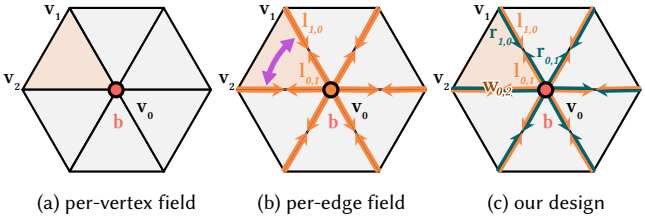
Feature design We compare against two simpler versions of our neural field: the per-vertex field and the per-edge field (Fig. 16). The per-vertex field is a hybrid feature field with features stored at the vertex. Additionally, we ablate our design of discontinuous edges by introducing a per-edge field. This field interpolates the features stored on the two consecutive half-edges of the query face (purple arrows in Fig. 16). This interpolation is similar to our method but the feature of this configuration is continuous across edges. Both versions are significantly worse than our model as reported in Table 3a 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.

λ_{discont} sensitivity We analyze the sensitivity of the parameter λ_{discont} (5×10^{-3} by default) in Eq. 10 by repeating the ablation experiments on our diffusion curve test set. When $\lambda_{\text{discont}}=0$, the loss function lacks the sparsity inducing term for discontinuous edges, resulting in a slight increase in their number due to small-scale discontinuous edges in almost continuous areas. As λ_{discont} increases, the discontinuous edges become cleaner, but start to lose detail near shape intersection regions. Quantitatively, Table 3b shows that the alternative values of λ_{discont} yield PSNR results with negligible improvements ($<1\%$ on average).

Activation We choose tanh for activations since we observe that ReLU competes with discontinuous edges when approximating discontinuities. As utilized by ReLU fields [KRWM22], ReLU activations tend to create steep slopes that approximate true discontinuities, serving an overlap role as our discontinuous features. In Fig. 17, our neural field with ReLU activations is less smooth compared to the version with tanh activations. As we rely on the smoothness of MLP, we decide on tanh for activations.

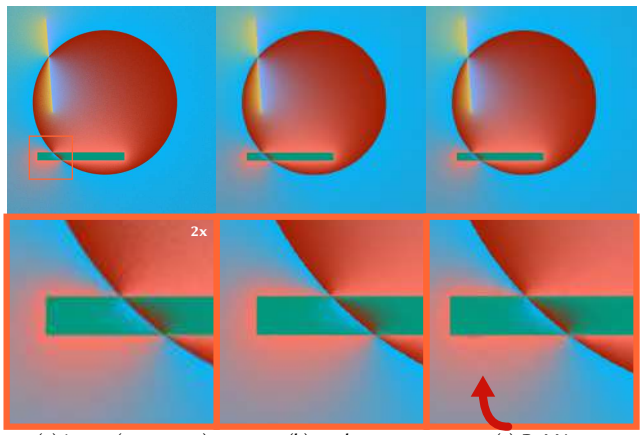
Importance sampling We randomly sample edges using the same number of samples without importance sampling on the diffusion curve test set. As the sampling edge number increases by roughly $8\times$, the Monte-Carlo edge sampling converges slower, leading to inaccurate discontinuity locations (blurs) and worse PSNR (by 0.6%) and Chamfer (by 35.2%).

Rounding We compare the edge counts and the parameter numbers before and after the rounding step. Without the rounding, all



(a) per-vertex field (b) per-edge field (c) our design

Figure 16: Different designs of mesh-based feature fields.



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

Figure 17: ReLU activations generate redundant discontinuities.

edges are considered potentially discontinuous and thus present in the resulting field — a similar setting as trivially applying Belhe et al.'s to our problem setting. After rounding, the number of edges with features decreases to $1/68$ on the diffusion curve test set and the number of parameters reduces to $1/5$. On more complicated inputs, the decreases are as follows: Fig. 10, 15b shows $1/4$ for edge counts and $1/3$ for parameters; Fig. 13, 15a shows $1/27$ for edge counts and $1/5$ for parameters; Fig. 2, 15d shows $1/2$ for both.

Appendix B: 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

$$\sum_{e\in F_i,F_j}\left\|f_0(x;F_i,F_j\in M)-I(x)\right\|^2+\lambda_{\text{Delaunay}}\text{trace}(\mathbf{L}),\qquad(11)$$

where F_i,F_j are the two faces defined by an edge; $\mathbf{L}$ is the positive

Table 3: *Ablation studies.* (a) Feature design and (b) effect of varying λ_{discont} (5×10^{-3} by default).

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) Feature design.

	Default	0	10^{-3}	10^{-2}	10^{-1}
PSNR ↑	44.486	44.416	44.539	44.708	43.519
Ratio ↑	100.0%	99.8%	100.1%	100.5%	97.8%

(b) Effect of varying λ_{discont}.

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 the cotan Laplacian is decreased by Delaunay flips and reaches a global minimum when the 2D triangulation is Delaunay [Ale19].

Field optimization To jointly optimize our neural field in triangle interiors and the triangle mesh, we compute the gradient of the rendering integral with respect to moving triangle area. This is achieved via the common differentiable rendering technique of edge-sampling Monte Carlo estimation [LADL18]

$$\nabla \int_\Omega \|f(\mathbf{x};V,\Theta) - I(x)\|^2 \, d\mathbf{x} \tag{12}$$

$$= \left(\nabla^T \int_\Omega f(\mathbf{x};V,\Theta) d\mathbf{x}\right) \cdot \int_\Omega (f(\mathbf{x};V,\Theta) - I(\mathbf{x})) d\mathbf{x}. \tag{13}$$

This gradient can be estimated as

$$\nabla \int_\Omega f \, dx \tag{14}$$

$$= \int_{\Omega \setminus E} \frac{\partial}{\partial \Theta} f \, d\mathbf{x} + \sum_{e_i \in E} \int_{\alpha_i(\mathbf{x})=0} \frac{\nabla_V \alpha_i(\mathbf{x})}{\|\nabla_V \alpha_i(\mathbf{x})\|} f(\mathbf{x}) \, dp(\mathbf{x}), \tag{15}$$

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 are defined by implicit functions $\alpha(x)$.

Appendix C: 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 a Gaussian kernel of size 3 before Canny edge detection. We call TriWild with a 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 [Pal22]. 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 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 the standard Xavier normal distribution and all biases as zero.

Appendix D: 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 an 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 a quality of 50 via IMAGEMAGICK.

InstantNGP We use the tiny-cuda-nn [Mül21] 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 [WLL22] and run it on the resulting aligned mesh from our method. The pixel-based MS denoising algorithm is implemented by replacing WANG, LIU, and LIU'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 [VZ21]. 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, a 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.