



MDNF: Multi-Diffusion-Nets for Neural Fields on Meshes

Avigail Cohen Rimón¹ , Tal Shnitzer² , Mirela Ben Chen¹ 

¹Technion - Israel Institute of Technology

²Broad Institute of MIT and Harvard

Abstract

We propose a novel framework for representing neural fields on triangle meshes that is multi-resolution across both spatial and frequency domains. Inspired by the Neural Fourier Filter Bank (NFFB), our architecture decomposes the spatial and frequency domains by associating finer spatial resolution levels with higher frequency bands, while coarser resolutions are mapped to lower frequencies. To achieve geometry-aware spatial decomposition we leverage multiple DiffusionNet components, each associated with a different spatial resolution level. Subsequently, we apply a Fourier feature mapping to encourage finer resolution levels to be associated with higher frequencies. The final signal is composed in a wavelet-inspired manner using a sine-activated MLP, aggregating higher-frequency signals on top of lower-frequency ones. Our architecture attains high accuracy in learning complex neural fields and is robust to discontinuities, exponential scale variations of the target field, and mesh modification. We demonstrate the effectiveness of our approach through its application to diverse neural fields, such as synthetic RGB functions, UV texture coordinates, and vertex normals, illustrating different challenges. To validate our method, we compare its performance against two alternatives, showcasing the advantages of our multi-resolution architecture.

CCS Concepts

• **Computing methodologies** → *Machine Learning, Shape Analysis*;

1. Introduction

Recent advancements in machine learning have led to a surge of interest in solving visual computing problems using coordinate-based neural networks, known as *Neural fields*. These networks parameterize the physical properties of scenes or objects across spatial and temporal dimensions. Neural fields have gained widespread adoption due to their ability to encode continuous signals over arbitrary dimensions at high resolutions, enabling accurate, high-fidelity, and expressive solutions [XTS*22]. They have demonstrated remarkable success in a variety of tasks, including animation of human bodies [HXS*21], mesh smoothing and deformations [YBHK21, DLJ*20], novel view synthesis [MST*21], mesh geometry and texture editing [YBZ*22], 3D reconstruction [DLJ*20, SGR*24], textured 3D reconstruction from images [OMN*19, KGM*22], shape representation and completion [PFS*19], and neural stylization of meshes [MBOL*22].

Despite their widespread success, these coordinate-based neural architectures remain vulnerable to spectral bias [JGH18] and demand significant computational resources. Among other generalizations, these shortcomings have been addressed through spatial decomposition strategies using grids [CXG*22, MESK22, TLY*21], which support rapid training and level of detail control. Additionally, techniques that encode input data using high-dimensional features through frequency transformations, such as

sinusoidal representations [MST*21, SMB*20, TSM*20], help mitigate the inherent low-frequency bias of neural fields [TSM*20].

Wu et al. [WJY23] propose an architecture that bridges these two approaches. They demonstrate that employing different grid resolutions focused on distinct frequency components, combined with proper localization, achieves state-of-the-art performance in terms of model compactness and convergence speed across multiple tasks. However, the proposed grid-based methods, including [WJY23], are designed for Euclidean spaces and do not account for the unique properties of non-Euclidean, irregular geometric domains like triangle meshes. Although adapting such methods to fit such structures via data modification has shown efficacy, it often overlooks the inherent characteristics of mesh data. Notably, meshes typically represent smooth manifolds with defined geometry, offering potential for enhanced understanding and representation. Furthermore, we aim to enhance the architecture's invariance to the multi-representational nature of mesh geometry, accommodating different resolutions and many equivalent vertex connectivities.

Despite significant advancements in learning on meshes, most existing neural architectures primarily prioritize improving generalization capabilities required for classic tasks such as segmentation and classification. Consequently, we find that these approaches struggle when tasked with capturing high-fidelity signals on individual meshes, limiting their effectiveness for neural fields on

meshes. Addressing this fundamental gap, our work puts the spotlight on accurately capturing high-resolution signals, setting the foundation for future improvements in intricate signal representation necessary for advanced mesh applications.

In this work, we introduce a novel geometry-aware framework for representing neural fields on triangle meshes that are multi-resolution across both *spatial* and *frequency* domains. Inspired by the Neural Fourier Filter Bank (NFFB) [WJY23] and leveraging the geometry-aware DiffusionNet architecture [SACO22], our approach decomposes the spatial and frequency domains using multiple DiffusionNet components representing different spatial resolutions and controlling frequency bands using Fourier feature mappings at different scales. We associate finer spatial resolution levels with higher frequency bands, while coarser resolutions are mapped to lower frequencies. This wavelet-inspired decomposition, combined with a carefully designed network architecture, enables our method to effectively learn and represent complex neural fields, accurately capturing intricate details and frequency variations. We demonstrate the efficacy of our approach through its application to diverse neural fields, including synthetic RGB functions, UV texture coordinates, and vertex normals, showcasing its robustness to discontinuities, exponential scale variations, and mesh modification. Additionally, we illustrate its practical utility by integrating it into a texture generation model, demonstrating its ability to preserve geometric fidelity and detailed features.

1.1. Related Work

We highlight relevant work that is related to the key components of our method: architectures for learning on meshes that leverage mesh geometry and neural fields on non-Euclidean domains.

Learning on Meshes Several works have proposed unique architectures to leverage mesh geometry and other structural properties for learning on meshes [HHF*19, MLR*20, CWKW19, WEH20, YLP*20, MKK21, SS21, SACO22]. Hanocka et al. [HHF*19] defined MeshCNN, a convolutional layer on meshes by learning edge features and defining pooling operations through edge collapse. Milano et al. [MLR*20] captures triangle adjacency in meshes through graphs of mesh edges and dual edges. Cohen et al. [CWKW19] and Wiersma et al. [WEH20] defined equivariant architectures to overcome challenges arising from the mesh geometry. Cohen et al. [CWKW19] developed gauge equivariant CNNs on manifolds that depend only on the intrinsic geometry, focusing on signals defined on the surface of the icosahedron, and Wiersma et al. [WEH20] proposed to overcome the rotational ambiguity of filter kernel transportation by defining rotation-equivariant features for CNNs. Another approach by Yang et al. [YLP*20], maps surface mesh patches onto flat tangent planes and aligns them to form a flat Euclidean structure, thereby mimicking standard convolutions. Mitchel et al. [MKK21] define surface convolutions through a scattering operation, which is more resilient to noise due to its aggregation of information from multiple coordinate systems. Other works use spectral geometry to facilitate learning on meshes. Smirnov et al. [SS21], learns spectral geometry elements to construct custom mesh features, and DiffusionNet [SACO22] leverages the heat equation and learns multiscale diffusion operations to propagate in-

formation across the manifold. These architectures commonly focus on segmentation, classification and correspondence learning tasks. While they form the basic architecture of our work, they are typically incapable of capturing subtle differences in multiple resolutions, as demonstrated in the experimental results, Section 4.

Neural Fields Neural fields have been increasingly used for learning functional representations in arbitrary resolutions, most commonly for Euclidean domains, e.g. [MST*21, XTS*22, KAL*21, PFS*19]. The foundational work, NeRF [MST*21], a coordinate-based neural network for view synthesis, demonstrated the importance of positional encodings to facilitate learning of high frequency data by neural networks. Subsequent works have used periodic activation functions [SMB*20], wavelet-like multi-resolution decomposition [WJY23], and a decomposition to a cascade of band-limited neural fields [SGR*24]. These works focus on Euclidean spaces, encoding non-Euclidean 2D manifolds as volumes, resulting in higher computational costs or failure to capture the manifold structure.

Neural Fields on Manifolds Recently, a few works have proposed methods for learning neural fields on non-Euclidean domains [KGM*22, XBM*23, BRGK23, XXH*21]. Bensaïd et al. [BRGK23] leverages neural fields for learning partial matching of nonrigid shapes. They use intrinsic positional encodings and a neural representation in the spectral domain to interpolate between matched sparse landmarks of partial shapes. NeuTex [XXH*21] represents meshes as 3D volume in a Euclidean space, but encodes texture with a 2D network. To enable texture representation and editing, they train mapping networks between the two spaces, which can be seen as learning a representation of the 2D surface. Koestler et al. [KGM*22] takes into account the manifold structure by using the eigenfunctions of the Laplace-Beltrami Operator as positional encodings, serving as point embeddings in the input of the trained neural network. This approach is conceptually similar to the concatenation of DiffusionNet [SACO22] and a Multi-Layer Perceptron (MLP), and we compare our method to such an architecture in Section 4. Sharing similarities with this approach, Grattarola and Vandergheynst [GV22] propose implicit neural representations based on intrinsic spectral embeddings from the graph Laplacian, generalizing coordinate-based representations to non-Euclidean domains. [XBM*23] further extends this concept and learns a continuous field, independent of the manifold discretization, by mapping a series of mesh poses to an implicit canonical representation and learning surface deformations fields for each pose. This approach requires a series of related inputs, such as a human mesh in different poses. Edavamadathil et al. [ESLR24] propose Neural Geometry Fields for Meshes, based on partitioning the mesh into parameterizable patches. However, their method is primarily designed for mesh compression, focusing on obtaining a compact neural representation of discrete surface geometry. A recent work, MeshFeat [MHC24], proposes a multi-resolution framework for learning features for neural fields on meshes. Their approach replaces feature learning on Euclidean hash grids with feature learning at mesh vertices across different resolutions, effectively implementing spatial decomposition. However, it aggregates features across resolution levels without explicitly controlling

their frequency content, and the learned features remain sensitive to mesh connectivity.

1.2. Contribution

To summarize, our contributions are as follows:

- We propose a novel geometry-aware framework for neural fields representation on triangle meshes that is multi-resolution across both *spatial* and *frequency* domains.
- We show that our method attains high precision in learning diverse neural fields, such as synthetic RGB functions, UV texture coordinates, and vertex normals, illustrating different challenges.
- We show that our method outperforms DiffusionNet in learning high-detail functions, and the NFFB model in handling discontinuous neural fields and mesh modifications.

2. Background

Our architecture draws inspiration from the architecture proposed by Wu et al. [WJY23], and builds upon two main works: DiffusionNet [SACO22] and Fourier feature mapping [TSM*20]. For completeness, we provide here a brief review of these works.

2.1. DiffusionNet

DiffusionNet [SACO22] is a discretization agnostic architecture for deep learning on surfaces. The architecture consists of successive identical DiffusionNet *blocks*. A central feature of each DiffusionNet block is the use of learned diffusion based on the heat equation to propagate information across the surface. This diffusion is discretized via the Laplacian \mathbf{L} and mass \mathbf{M} matrices of the surface. In our work, we use the cotangent-Laplace matrix, which is ubiquitous in geometry processing applications [CDGDS13, Mac49, PP93].

For efficient diffusion computation, the authors propose to use a spectral method that utilizes the k smallest eigenpairs ϕ_i, λ_i of the generalized eigenvalue problem $\mathbf{L}\phi_i = \lambda_i \mathbf{M}\phi_i$. The diffusion layer $h_t(u)$ corresponding to time t is implemented by projecting the input feature channel u onto this truncated basis, exponentially scaling the coefficients by $-\lambda_i t$, and projecting back:

$$h_t(u) := \Phi \begin{bmatrix} e^{-\lambda_1 t} \\ e^{-\lambda_2 t} \\ \vdots \end{bmatrix} \odot (\Phi^T \mathbf{M} u)$$

where \odot denotes the Hadamard (elementwise) product and Φ, Λ are the matrices of generalized eigenvectors and eigenvalues, respectively. The learned diffusion times, optimized per feature channel, control the spatial support ranging from purely local to totally global. Here, we have briefly reviewed only the aspects critical for understanding our method; see [SACO22] for further details.

2.2. Fourier Feature Mapping

The work by Tancik et al. [TSM*20] addresses the problem of "spectral bias" in coordinate-based multi-layer perceptrons (MLPs), which refers to their inherent limitation in accurately modeling high-frequency components due to the rapid decay of eigenvalues in their neural tangent kernels (NTKs) [JGH18]. The authors

propose using a Fourier feature mapping that applies a non-linear transformation to the input coordinates before passing them to the MLP. They report a Gaussian mapping as most effective, where input coordinates are multiplied by random Gaussian matrices to produce high-dimensional Fourier features. Theoretically, they show that this mapping transforms the NTK into a stationary kernel with a tunable bandwidth. This bandwidth, which determines the width of the kernel's effective frequency spectrum, is controlled by the scale (standard deviation) of the Gaussian matrices. A larger scale allows representing higher frequencies, overcoming spectral bias.

2.3. Neural Fourier Filter Bank (NFFB)

Wu et al. [WJY23] introduce a novel neural field framework named "Neural Fourier Filter Bank" (NFFB) that decomposes the target signal jointly in the spatial and frequency domains, inspired by wavelet decomposition [Sha49]. The core idea is to utilize multi-layer perceptrons (MLPs) to implement a low-pass filter by leveraging their inherent frequency bias, and to employ grid features at varying spatial resolutions alongside Fourier feature mappings [TSM*20] at different scales to create a high-pass filter. A novel aspect of this framework is the correlation of finer spatial resolutions with higher frequency bands, whereas coarser resolutions correspond to lower frequencies. Fourier feature mappings are applied at scales that match the respective spatial resolutions of each grid feature. The proposed architecture feeds the high-frequency components into sine-activated MLP layers at appropriate depths, mimicking the sequential accumulation of higher frequencies on top of lower frequencies in wavelet filter banks. This wavelet-inspired decomposition into spatial and frequency components, coupled with the association of specific resolutions with corresponding frequencies, allows for efficient learning of detailed signals while maintaining model compactness and fast convergence.

3. Method

We propose a multi-resolution framework that facilitates representing neural fields on meshes across both *spatial* and *frequency* domains. As illustrated in Figure. 1, our pipeline comprises three key stages: (1) Diffusing features across mesh vertices via multiple DiffusionNet components (Section 3.1) to capture spatial variations, (2) Transforming these diffused features through Fourier feature mapping (Section 3.2) to associate different frequency bands with the respective resolution levels, and (3) Composing the multi-resolution, multi-frequency signal representation using a sine-activated MLP in a wavelet-inspired manner (Section 3.3). We delve into the details of each stage in the following sections.

The input to the network is a matrix $X \in \mathbb{R}^{n \times 3}$, representing the 3D mesh vertex coordinates. Demonstrating the representational capability of our framework rather than the quality of loss function, the experiments in Section 4 employ supervised learning, where the optimization target is incorporated into the loss function. We explicitly specify the exact loss formulation used in each experiment. In Section 5, we adopt the configuration from [MBOL*22], which is unsupervised; see [MBOL*22] for further details.

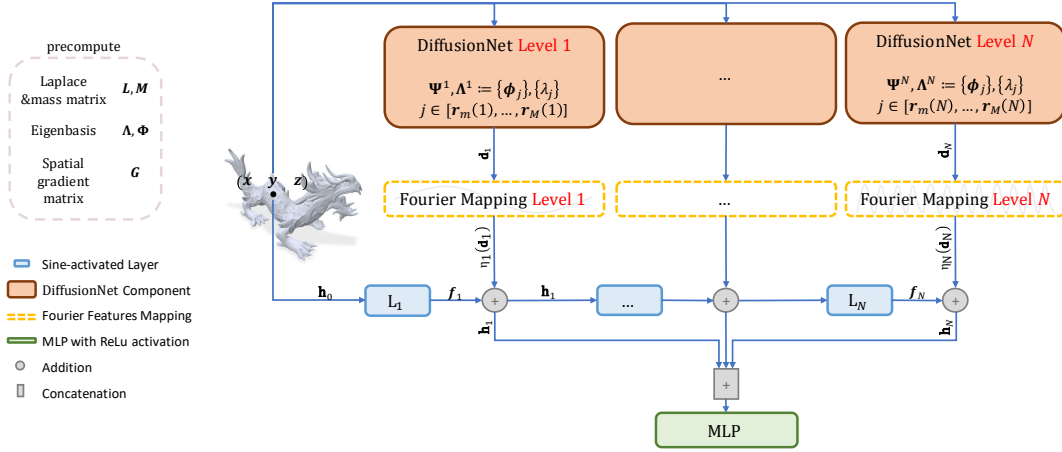


Figure 1: Framework overview. The backbone of our architecture is a sine-activated Multi-Layer Perceptron (MLP), which receives two signals at each linear layer: the output from the previous layer (h_{i-1}), representing a low-frequency signal, and another signal produced by the i -th resolution level, $\eta_i(d_i)$, representing a higher-frequency signal. The initial input to the first layer L_1 is $h_0 := X \in \mathbb{R}^{n \times 3}$. To generate the features $\eta_i(d_i)$, we construct N DiffusionNet components that take X as input and produce $d_i \in \mathbb{R}^{n \times F}$. Fourier Feature mapping layers are then applied to encode these features into appropriate frequencies, resulting in $\eta_i(d_i) \in \mathbb{R}^{n \times m}$. These features are subsequently fed into the linear layer L_i as the higher-frequency components within the sine-activated MLP. To construct the final output, we concatenate the intermediate outputs $h_i \in \mathbb{R}^{n \times m}$ and feed them into a regular MLP with ReLU activation. The definition of G referenced in the "precompute" rectangle can be found in Supplemental Material Section 1.1.

To clarify our architecture's pipeline, we use a synthetic example featuring the Chinese dragon mesh with 125K vertices and 250K faces. The mesh is divided into three groups—Red, Yellowish, and Blue—each linked to a distinct function depicted in the inset figure. These groups represent increasing frequencies: Red corresponds to ϕ_1 , Yellowish to ϕ_{125} , and Blue to ϕ_{500} , where $\phi_j \in \mathbb{R}^n$ is the j -th eigenfunction of the Laplace-Beltrami operator on the mesh, and n is the number of vertices. We generate the target neural field by mapping the patchwork function to an RGB using the HSV colormap. More details can be found in Subsection 4.1.



Illustrative example. Partitioning of mesh vertices.

3.1. DiffusionNet Layers

Motivation Aligned with the strategy in [WJY23], our pipeline's first stage inputs features into a multi-component "layer," each component associated with a different resolution band, representing varying spatial resolutions of mesh features. Unlike NFFB [WJY23], our architecture replaces each hash grid with a DiffusionNet component. As discussed in Section 2.1, DiffusionNet utilizes diffusion layers to facilitate spatial communication and optimizes diffusion support for each feature channel.

The choice of adopting the DiffusionNet architecture is based on two main reasons. The first stems from its inherent compatibility with irregular data structures, specifically triangle meshes, as opposed to hash grid structures suited for regular formats like images. Despite previous adaptations of triangle meshes to regular-structured architectures yielding favorable results, a geometry-aware approach like DiffusionNet has proven more accurate and

efficient in various applications. Due to that, substituting the grid representation with DiffusionNet is particularly effective for mesh data. Additionally, DiffusionNet facilitates discretization-agnostic learning, enhancing generalization capabilities of the overall architecture.

Second, DiffusionNet intrinsically facilitates a methodology akin to the multi-resolution hash-grid paradigm of NFFB. The diffusion time parameter can be utilized to adjust spatial resolutions via the initial values assigned to each component. Furthermore, employing the "spectral method" for the diffusion process enables each component to be associated with a distinct set of eigenvectors, enhancing their spatial resolutions as well.

Formally, the DiffusionNet component at the i -th level, δ_i , maps the per vertex input 3D coordinate $X \in \mathbb{R}^{n \times 3}$ to an F -dimensional feature space: $\delta_i: \mathbb{R}^{n \times 3} \rightarrow \mathbb{R}^{n \times F}$. Let N denote the number of DiffusionNet components.

Splitting the spectrum Considering the total number of eigenvectors k_{eig} used for diffusion, we distribute the eigenvectors evenly across the levels, associating the eigenvectors corresponding to the lowest eigenvalues with level 1 and highest to level N .

For each level $i \in [1, N]$, we define the range of eigenvector indices used for diffusion in the i -th DiffusionNet component as $[r_m(i), r_M(i)]$ where

$$\begin{aligned} r &:= \text{linspace}(0, k_{\text{eig}}, N + 1) \\ r_m(i) &:= r(i) \quad r_M(i) := r(i + 1) \end{aligned} \quad (1)$$

where $\text{linspace}(\text{start}, \text{end}, \text{steps})$ is a one-dimensional vector of size steps whose values are evenly spaced from start to end , inclusive.

The corresponding sets of eigenvectors Ψ_i and eigenvalues Λ_i

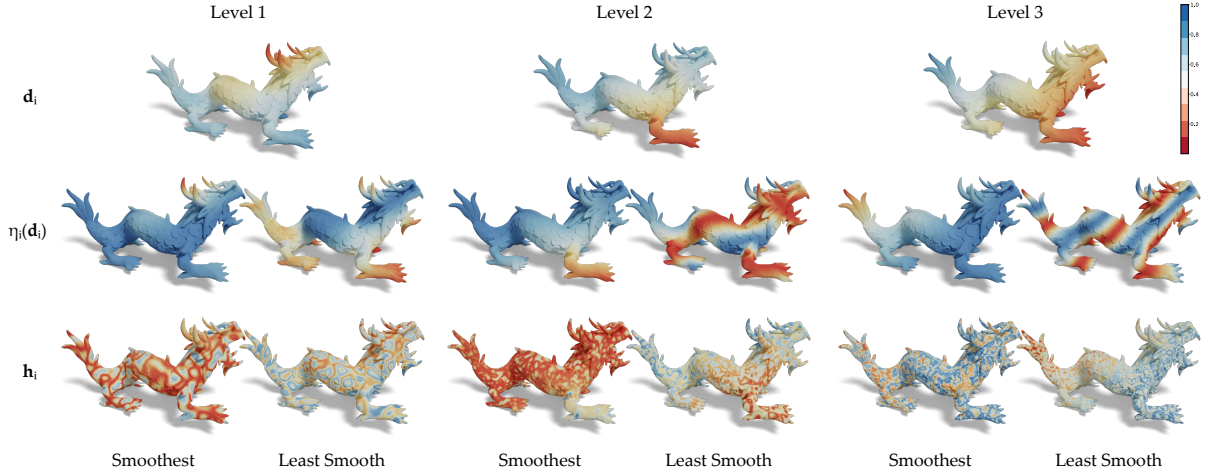


Figure 2: Analysis of the illustrative example. The figure displays the output features at each resolution level across the three key stages of the pipeline. Except for the first stage (d_i), where each level contains only one feature ($F = 1$), both the smoothest and least smooth features are showcased for each level. Note that the features become less smooth as the stage and level increase.

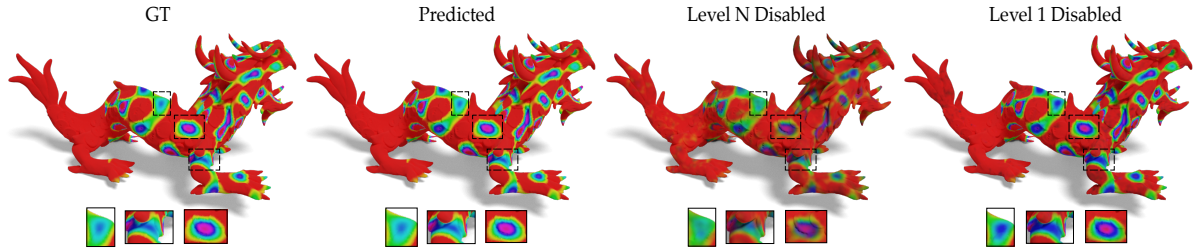


Figure 3: Analysis of the illustrative example. From left to right: (1) Ground truth (GT) RGB function (2) RGB function predicted by our model. (3) Output function with the N -th level features h_N disabled. (4) Output function with the first level features h_1 disabled. Below each mesh, three zoomed-in areas of the function are presented. Compared to the GT function, the output in (3) appears significantly blurrier, roughly capturing outlines, while the output in (4) presents accentuated contrast, occasionally with an overstated effect.

used for diffusion at the i -th level are:

$$\Psi_i := \{\phi_j\}_{j=r_m(i)}^{j=r_M(i)} \quad \Lambda_i := \{\lambda_j\}_{j=r_m(i)}^{j=r_M(i)} \quad (2)$$

Splitting diffusion time Recall the diffusion time parameter in DiffusionNet controls the spatial resolution of diffusion, theoretically ranging from local to global scales. However, in practice, such range isn't fully realized, as shown in Sec 4. The diffusion process, as implemented by the "spectral method", acts as a low-pass filter due to the exponentiation $e^{-\lambda_j t}$, where t is the diffusion time and λ_j the j -th Laplacian eigenvalue. By creating multiple Diffusion-Nets, each with distinct eigenvalue ranges, we achieve a refined representation of high-frequency components.

Following the initialization scheme considered in Wu et al [WJY23] for the Gaussian distribution variance (as in our Equation (5)), we initialize the diffusion times of the i -th DiffusionNet component by $t(i)$ defined as

$$t(i) := t_{\text{base}} \cdot (t_{\text{exp}})^i \quad (3)$$

Typically t_{base} is set to the squared mean edge length of the mesh and $t_{\text{exp}} < 1$.

Figure 2 illustrates the output features of each level at the key pipeline stages, displaying the smoothest and least smooth feature channels per level and stage. See Supplemental Material Section 2 for measuring function smoothness. The first row shows $d_i := \delta_i(x)$ for $i \in [1, 2, 3]$. Since we set $F = 1$ for simplification, only one feature is output at this stage. We observe that the functions at the three levels in this stage exhibit smooth behavior.

3.2. Fourier Feature mapping

As in NFFB, the Fourier feature mapping stage serves to associate each level of the multi-resolution representation with a distinct frequency band. Inspired by the Fourier feature mapping approach of [TSM*20], we apply a sinusoidal transformation to the output features from the previous DiffusionNet stage.

In more details, the Fourier feature at the i -th level is defined as a mapping from the DiffusionNet output features at the i -th level $d_i \in \mathbb{R}^{n \times F}$ to an m -dimensional feature space:

$$\eta_i(d_i) := [\sin(2\pi \cdot d_i \cdot B_{i,1}), \dots, \sin(2\pi \cdot d_i \cdot B_{i,m})] \quad (4)$$

where $B_{i,1}, B_{i,2}, \dots, B_{i,m}$ are trainable parameters in \mathbb{R}^F forming

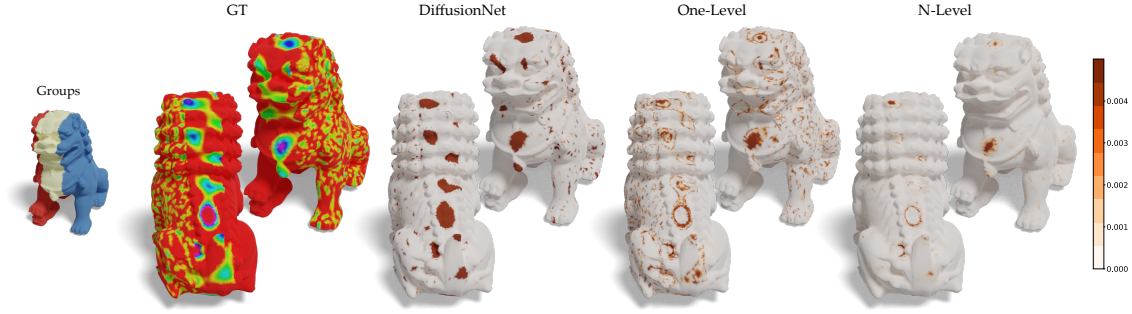


Figure 4: Synthetic example results. From left to right: (1) Partitioning of mesh vertices into three groups, colored Red, Yellowish, and Blue. (2) Two poses of the ground truth RGB function y_{rgb} , with each group assigned a distinct function. (3) Error distribution of the field learned by DiffusionNet. (4) Error distribution of the field learned by the One-Level model. (5) Error distribution of the field learned by the N-Level model. Note that our N-Level model demonstrates superior performance relative to the other models, with an error distribution that shows significantly fewer artifacts.

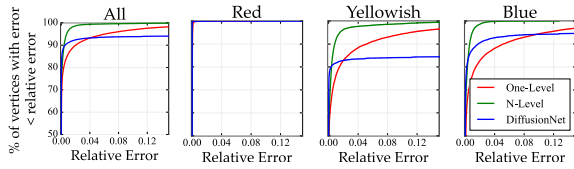


Figure 5: Synthetic example results. Cumulative Distribution Functions (CDFs) of vertex errors for each model across four vertex subsets. We note that for all subsets except the "Red", our N-Level model demonstrates superior performance. For the "Red", the function considered is a constant trivial function which is easily handled by all models. It is noteworthy that although the One-level model requires a longer duration, it ultimately manages to improve, whereas DiffusionNet alone stagnates.

the frequency transform coefficients on the i -th level, and m is a hyper-parameter. The frequency ranges for each level are defined by the initialization of the $B_{i,j}$ coefficients. Drawing on the Gaussian random Fourier feature mapping by Tancik et al. [TSM*20], we set these coefficients using a Gaussian distribution with a mean of 0 and a level-specific variance σ_i (Equation (5)). Finer resolutions levels, associated with higher frequencies, are initialized with greater variance, biasing them towards encoding higher frequency signal components. This adaptive initialization approach allows each resolution level to naturally associate with specific frequency bands without pre-setting fixed ranges.

Practically, let $\sigma_{base}, \sigma_{exp} \in \mathbb{R}$, we initialize the i -th level coefficients with variance $\sigma_i \in \mathbb{R}$ defined by

$$\sigma_i := \sigma_{base} \cdot (\sigma_{exp})^i \quad (5)$$

where $\sigma_{base}, \sigma_{exp}$ are hyper-parameters, and $\sigma_{exp} \geq 1$.

Referring again to Figure 2, the second row depicts the output features $\eta_i(d_i)$ for $i \in [1, 2, 3]$. We observe that the frequency of the least smooth feature increases as the level increases.

3.3. Composing the final output

The next stage composes the final output using the Fourier transformed features $\eta_i(d_i) \in \mathbb{R}^{n \times m}$ across levels $i \in [1, N]$. Two critical observations from [WJY23] inform this process: First, features across levels are not necessarily orthogonal, calling for learned layers for optimal combination and mitigation of non-orthogonality. Second, implementing residual connections helps aggregate and joint update of the multi-resolution features, maintaining consistent processing depth across all levels in the network.

We thus start by applying a sine-activated MLP [SMB*20] that takes in the Fourier features $\eta_i(d_i)$ in a manner that sequentially accumulates higher-frequency components on top of lower-frequency components.

More formally, let us denote the i -th layer as L_i where $i \in [1, N]$, using f_i to represent the output of L_i , and h_i to denote the combination of the i -th layer output with the next level's features:

$$f_i[v]^T := \sin(\alpha_i \cdot \mathbf{W}_i \cdot h_{i-1}[v]^T + \mathbf{b}_i), \quad h_i := f_i + \eta_i(\mathbf{d}_i) \quad (6)$$

where for ease of notation we define $h_0 := \mathbf{X} \in \mathbb{R}^{n \times 3}$. Since the MLP is weight-shared across vertices, we denote by $f_i[v], h_i[v]$ the entries of f_i, h_i corresponding to vertex v , respectively. Here, $\mathbf{W}_i \in \mathbb{R}^{m \times m}$ (with $\mathbf{W}_0 \in \mathbb{R}^{m \times 3}$) and $\mathbf{b}_i \in \mathbb{R}^m$ are the trainable weight and bias parameters in layer L_i , and α_i is analogous to the w_0 hyperparameter in SIREN [SMB*20], acting as a frequency scaling factor that allows controlling the frequency band that this level focuses on representing.

Next, as illustrated in Figure 1, we establish residual connections by concatenating the outputs $h_i \in \mathbb{R}^{n \times m}$ from each level and passing them through an additional MLP with ReLU activations, while also transferring them to the subsequent layer L_{i+1} as described earlier. Alternatively, as suggested in [WJY23], instead of concatenating $\{h_i\}_{i=1}^N$, one could pass each feature through a per-level linear layer O_i and sum the outputs to obtain the final feature representation. We refer to [WJY23] for further details.

The third row in Figure 2, representing the output features h_i for $i \in [1, 2, 3]$, exhibits features that are significantly less smooth than those in previous stages. Further, in this stage we can see that for

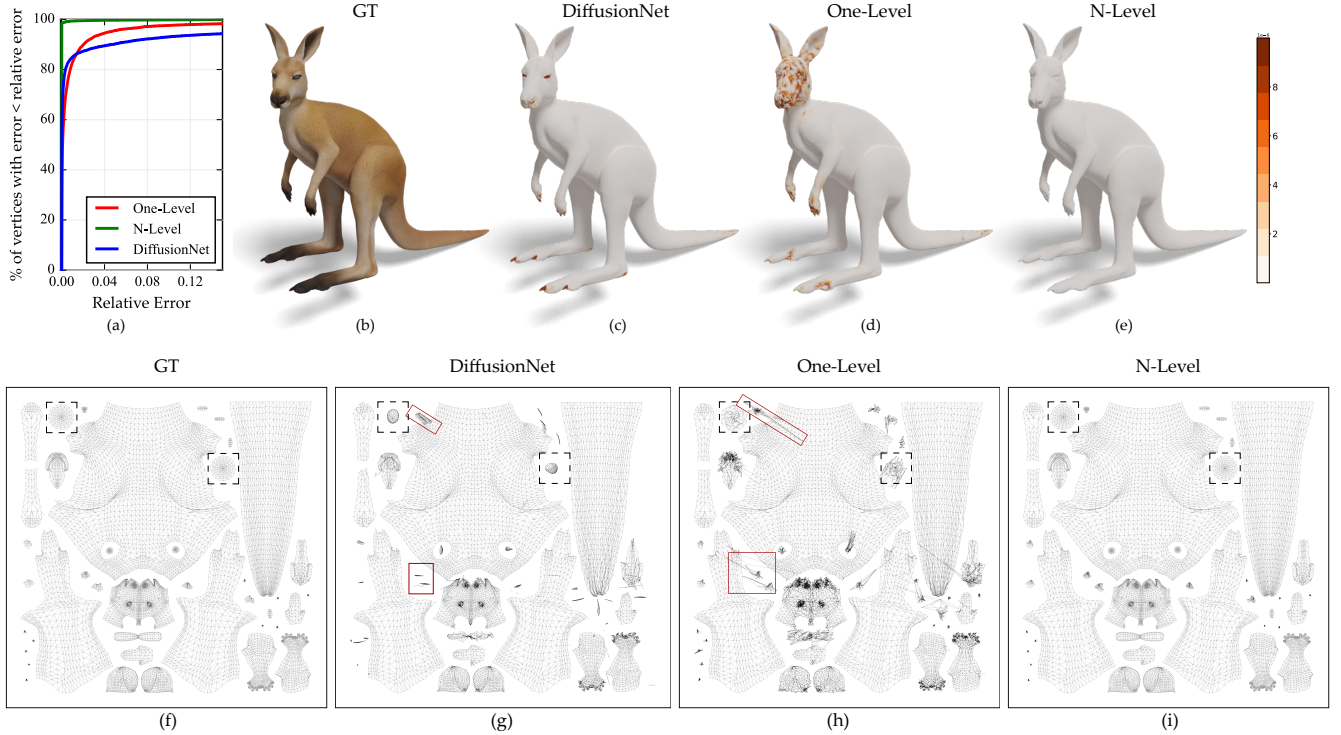


Figure 6: Discontinuity of mesh and UV coordinates. Upper row: (a) Cumulative Distribution Functions (CDFs) of vertex errors for each of the three models. (b) Ground truth textured mesh. (c) Error distribution of the UV function learned by the DiffusionNet model. (d) Error distribution of the UV function learned by our One-Level model. (e) Error distribution of the UV function learned by our N-Level model. Bottom row: the corresponding learned 2D UV coordinates for each model. Note that our N-Level model (e, i) is free from artifacts at the presented error level and exhibits the best results, also quantitatively, as shown in (a). Conversely, the DiffusionNet model (c, g) exhibits high errors at the eyes (dashed rectangles in (f, g, h, i)) and other pointed areas, and tends to squash small areas in the UV coordinates. Our One-Level model shows significant errors in the head region (d), and creates undesirable overlaps in the UV coordinates (h). Examples for high error areas in the UV are marked by red rectangles in (g, h).

both the smoothest and least smooth features, higher levels correspond to increasingly noisier features.

To gain insight into the resolutions levels learned by our trained network, Figure 3 depicts the output neural field representing the RGB function during evaluation, with either the first or last level disabled. Disabling level N generally results in a blurry output, reflecting lower frequency components, whereas disabling level 1 produces a high-contrast function tied to higher frequencies. We analyze this by zooming in on three areas across the ground truth (GT), our model's predicted function, and outputs with level N or 1 disabled. The output from disabling level N is notably blurrier, capturing only basic texture outlines compared to the GT. Conversely, disabling level 1 enhances contrast, often exaggerating transitions. For instance, areas that are light blue in the GT and surrounded by similar light green regions appear as a more distinct blue, despite their subtle difference in hue in the GT.

3.4. Implementation Details

The implementation details such as the loss function, number of training epochs, and network size are adjusted for each experi-

ment. Outside of these customized components, the overall training setup remains consistent across all experiments. We implement our method in PyTorch [PGM*19], and utilize the Adam optimizer [KB14] with the default settings of $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The learning rate is set to 10^{-4} , and it is reduced by a factor of 0.7 every 700 iterations. We set the output dimension of DiffusionNet components as $F = 2$, and the maximal index of the Laplacian eigenpair considered in the diffusion process, k_{eig} , is set to 500. We run all experiments on a single NVIDIA A40 GPU. For brevity, only the essential details are presented here; for a detailed description of the hyperparameters, see the Supplemental Material.

4. Experimental Results

We evaluate our method on three neural fields: synthetic RGB function (Section 4.1), UV texture coordinates (Section 4.2), and vertex normals (Section 4.3). Focusing on demonstrating our architecture, all models were trained using a supervised approach.

We compare against two baselines: a single DiffusionNet component, and our method with $N = 1$, denoted as the One-Level model. We refer to our method as the N-Level model where $N > 1$,

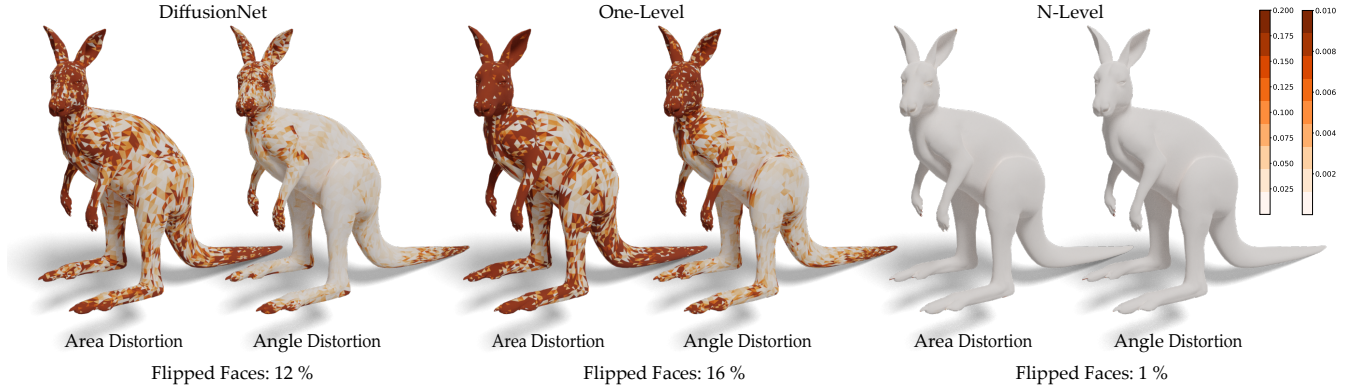


Figure 7: Discontinuity of mesh and UV coordinates. Three UV distortion metrics are presented for each model: area distortion, angle distortion, and the percentage of flipped faces. The area and angle distortion are visualized as mesh functions for each model, while the percentages of flipped faces are noted in text in the bottom row. We observe that our N-level model significantly outperforms the two other models in terms of these metrics.

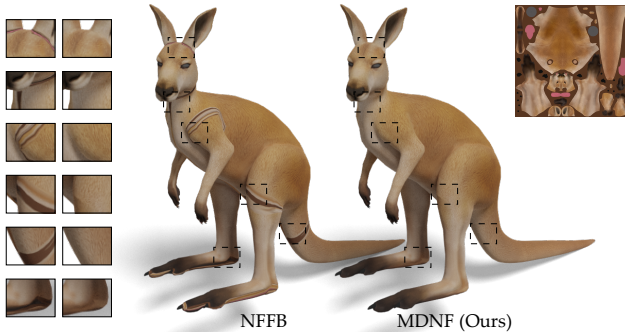


Figure 8: Discontinuity of mesh and UV coordinates. Textured mesh visualization using UV coordinates learned by NFFB (left) and our N-Level model (right). Areas of texture mapping discontinuities reveal significant artifacts in the NFFB results, while our method maintains mapping accuracy. Zoomed insets (left) highlight key areas where the differences are most prominent. The texture image is shown in the top right for reference.

with N determined empirically for each experiment. For all models, we report the results of the best-performing configuration.

In Sections 4.2 and 4.3, we also compare the performance of the NFFB architecture with our N-level model to further motivate our architecture.

4.1. Synthetic Example

To illustrate the effectiveness of our method, we start with a synthetic example, resembling the one in the Section 3. In this experiment we define the target field to be a 3-channel RGB function $y_{rgb} \in \mathbb{R}^{n \times 3}$ defined on a mesh, where n denotes the number of mesh vertices. We train the model by minimizing the mean square error (MSE), hence our loss function for this task is

$$\mathcal{L}_{rgb}(y) := \frac{1}{n} \|y - y_{rgb}\|_2^2 \quad (7)$$

Data. We demonstrate this example using the Chinese lion mesh, composed of 50K vertices.

Neural Field Generation To generate the function y_{rgb} , we first partition the mesh vertices into three groups based on their x coordinates, as visualized in Figure 4. We denote the Red, Yellowish, and Blue, groups by group_1 , group_2 , and group_3 , respectively. We assign to each group a scalar function: $g_1 := \phi_1 \in \mathbb{R}^n$ (constant), $g_2 := \phi_{125} \in \mathbb{R}^n$, and $g_3 = g_p \in \mathbb{R}^n$ generated as Perlin noise on the mesh [Per85, Vig22]. Note that the frequency of the functions g_i increases with i . We define a patchwork function $q \in \mathbb{R}^n$ on the mesh such that $q[\text{group}_i] = g_i[\text{group}_i]$. y_{rgb} is then derived by mapping q , normalized to $[0, 1]$, to a HSV colormap by defining the Hue parameter. See y_{rgb} in the GT figures in Figure 4.

Results Figure 4 shows the error distributions for the fields learned by the three models, clipping errors above 5×10^{-4} for clearer visualization. Our N-Level model outperforms the others, exhibiting fewer artifacts. To provide quantitative results as well, Figure 5 presents the Cumulative Distribution Functions (CDFs) of vertex errors across four groups: all vertices, group_1 (Red), group_2 (Yellowish), and group_3 (Blue), quantifying the percentage of vertices at each error level. For each vertex v , error is measured as MSE: $\frac{1}{3} \|y(v) - y_{rgb}(v)\|_2^2$. The x -axis represents relative error, calculated by dividing vertex errors by the maximal error across all models for each subset. We note that the N-Level model shows superior performance for all groups except for the Red group, which corresponds to a constant function and is considered a trivial region.

4.2. Discontinuities and Exponential Scale Variations

We further evaluate our method on neural fields representing UV coordinates of textured meshes, which are typically non-continuous and exhibit exponential scale variations when generated by conformal parametrization.

As in Section 4.1, we train the model by minimizing the mean

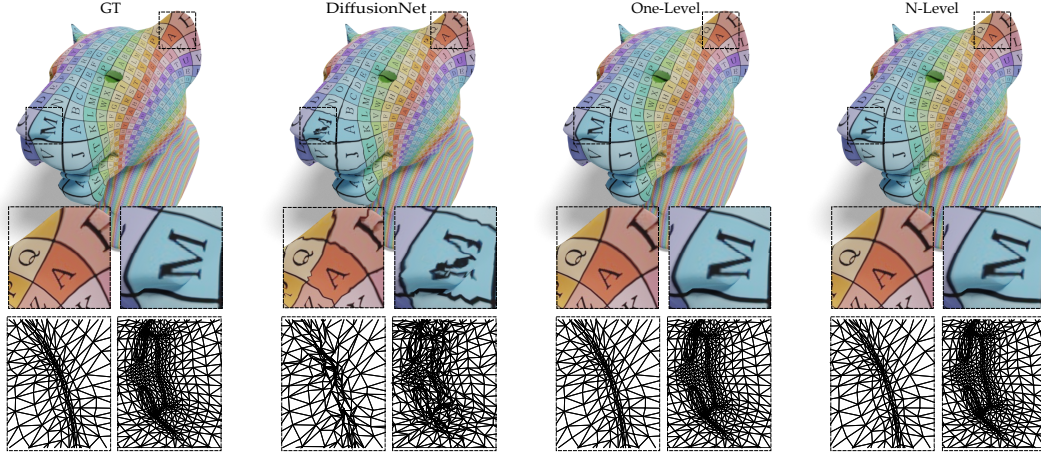


Figure 9: Exponential scale variations. From left to right: (1) GT texture, (2) Texture learned by the DiffusionNet model, (3) Texture learned by our One-Level model, (4) Texture learned by our N-Level model. The middle row zooms in on the texture in the nose and ear areas. The bottom row zooms in on the UV coordinates in the nose and ear areas. We observe that the DiffusionNet model exhibits significant texture distortions, while the textures learned by our One-Level and N-Level models closely resemble the GT texture.

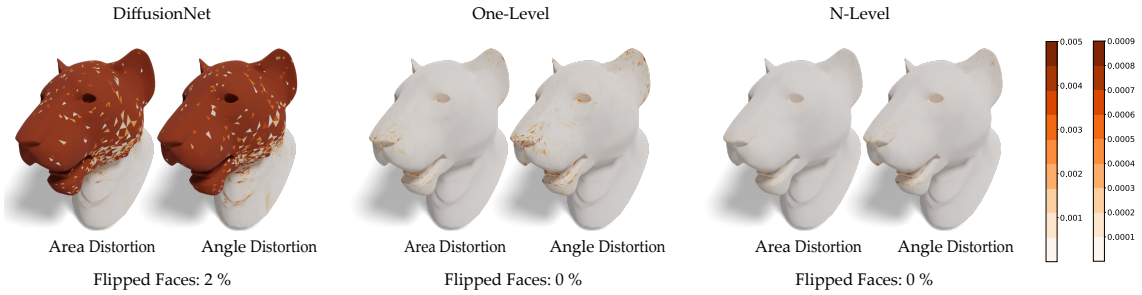


Figure 10: Exponential scale variations. Three UV distortion metrics are presented for each model: area distortion, angle distortion, and the percentage of flipped faces. The area and angle distortion are visualized as mesh functions for each model, while the percentages of flipped faces are noted in text in the bottom row. Note that the DiffusionNet exhibits poor performance at the displayed error level, and that our N-level model attains the best results in terms of area and angle distortions.

square error (MSE), hence our loss function for this task is

$$\mathcal{L}_{uv}(y) := \frac{1}{n} \|y - y_{uv}\|_2^2 \quad (8)$$

where $y_{uv} \in \mathbb{R}^{n \times 2}$ defines the UV texture coordinates of vertices.

4.2.1. Discontinuity of Mesh and UV Coordinates

Data In this example, we use a Kangaroo mesh with texture, with a total number of 10K vertices. The geometry of this mesh is composed of multiple connected components.

Results Figure 6 displays the ground truth texture mesh (b) and error distributions for the three models (c, d, e), with errors above 1×10^{-5} clipped for visualization. The bottom row shows the 2D UV coordinates for each model. On the left (a), the CDFs of vertex errors are shown. Notably, DiffusionNet exhibits high errors at distinct features like eyes, nails, and tail tips. In its UV coordinates (g), DiffusionNet tends to squash and distort smaller regions. The One-Level model, while showing significant errors at

the head area, has less squashing than DiffusionNet but has distortions that cause overlaps with other texture components in UV (h). Conversely, the N-Level model outperforms the others both qualitatively and quantitatively, with minimal distortions in its 2D UV coordinates, closely resembling the ground truth and with the best CDF results.

Figure 7 compares three UV distortion metrics for each model: area distortion, angle distortion, and the percentage of flipped faces. Area distortion is measured by the absolute difference from 1 of the ratio between ground truth and predicted triangle areas, with values over 0.2 clipped. Angle distortion involves the absolute difference from 1 of the mean ratio between ground truth and predicted triangle angles, clipping values exceeding 0.01. The bottom row notes in text the flipped faces percentages: 12% for DiffusionNet, 16% for One-Level, and 1% for the N-Level model. Overall, the N-Level model outperforms the others across all metrics.

Figure 8 shows a comparison between the UV learned by our N-Level model and NFFB. We observe that NFFB struggles particu-

larly in regions where the texture mapping exhibits discontinuities, producing visible artifacts in these challenging areas. In contrast, our N-Level model generates a clean UV map without distortions. Figure 2 in the supplemental material demonstrates this on an additional example, where in addition to difficulties with discontinuities the NFFB UV map exhibits noisy behavior in some regions.

4.2.2. Exponential Scale Variations

Data In this example, we utilize a truncated lion head mesh with 8K vertices and UV coordinates computed using conformal mapping [BCGB08], which leads to exponential scale variations notably between the head and neck areas, see Figure 9.

Results Figure 9 displays the GT texture alongside the results of the three models. The top row features the textured mesh, while the middle and bottom rows show zoomed-in regions of the texture and UV coordinates, respectively. The DiffusionNet texture reveals high distortion areas, but both the One-Level and N-Level models accurately capture the UV field.

Figure 10 evaluates three UV distortion metrics for each model. Area distortion values above 5×10^{-3} and angle distortion values above 9×10^{-4} are clipped. The bottom row notes flipped face percentages: 2% for DiffusionNet and 0% for both the One-Level and N-Level models. Although the One-Level and N-Level models show similar performance, the N-Level model still demonstrates the highest accuracy in area and angle distortions. We note that NFFB achieves results comparable to ours on this smoother mapping.

4.3. Mesh Generalization

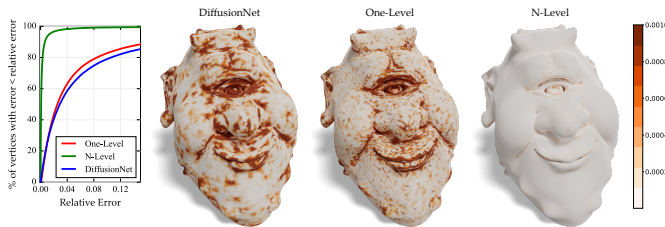


Figure 11: Mesh Generalization. From left to right: (1) CDFs of vertex errors for each of the three models. (2) Error distribution of the field learned by DiffusionNet. (3) Error distribution of the field learned by our One-Level model. (4) Error distribution of the field learned by our N-Level model. Note that from both a quantitative and qualitative perspective, our N-Level model markedly surpasses the performance of the other two models.

In this experiment, we demonstrate our architecture’s ability to generalize across multiple versions of a single mesh. Starting with a base mesh, we generate several subdivided versions via a variant of Loop subdivision, as implemented by MeshLab [C*22], which avoids adding new vertices if triangle edges are below a specified threshold. We apply subdivision iterations until additional triangles are negligible. Since our base mesh is not overly coarse, not all triangles are subdivided in each iteration. However, before being

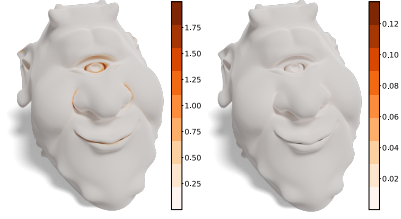


Figure 12: Mesh Generalization. Error comparison between our N-Level model and NFFB on test mesh. The NFFB was trained on the mesh that is one resolution level lower than the test. We observe that our model outperforms the NFFB. Note the different colorbars.

fed into the network, each mesh is centered and normalized, making triangle additions significantly change the mesh embedding.

We aim to learn the neural field defined by mesh vertex normals, $y_n \in \mathbb{R}^{n \times 3}$. Focusing on the direction of these normals, we train the model by minimizing the mean cosine distance error [SWY75, HPT22]. Thus, our loss function is given by:

$$\mathcal{L}_n(y) := \frac{1}{n} \sum_v \left(1 - \frac{\langle y(v), y_n(v) \rangle}{\|y(v)\|_2 \cdot \|y_n(v)\|_2} \right) \quad (9)$$

where $y_n(v) \in \mathbb{R}^3$ is the normal vector at vertex v .

Data The base mesh used for the subdivision iterations is the smiling ogre mesh, which comprises of 20K vertices. We then generated five additional subdivided versions, with the largest containing 33K triangles and 65K faces. The dataset was split into training and testing sets; the training set comprises meshes subdivided through [0, 1, 2, 3, 4] iterations, and the test set contains only the mesh generated by the 5-th subdivision iteration.

Results Figure 11 displays the CDFs of vertex errors on the left, and the error distributions of the three models on the right, where clipping values exceeding 0.001 for visualization. Both quantitatively and qualitatively, our N-Level model significantly outperforms the other two models.

Figure 12 compares the error values on the test mesh between NFFB and our N-Level model. The visualization is shown without error clipping; note the different colorbar scales for each method. It can be observed that our model performs better. The NFFB was trained on a mesh that is one resolution level lower than the test mesh (4 subdivision iterations).

5. Illustrative Application

To demonstrate the practical utility of our architecture, we integrate it into the text2mesh framework [MBOL*22], a CLIP-based [RKH*21] model that generates textured meshes from text prompts. This architecture employs three MLPs, where a "base" MLP’s output feeds into two subsequent networks that learn color and displacement values for each vertex of a base mesh.

In our experiment, we replace the base MLP with our architecture and compare the output of the original text2mesh implementation with our modified version output. We reproduce two examples



Figure 13: Illustrative Application. Comparison between the text2mesh architecture without MDNF (left) and our modified version with MDNF (right) for the given prompts. See the experiments details in supplemental. While both models successfully generate meshes corresponding to the given prompts, our architecture achieves finer geometric resolution and enhanced detail preservation. Note the sharper features and more defined geometric details in our result, demonstrating the benefit of replacing the base MLP with our multi-resolution architecture.

given in the paper, both using the same base human mesh. The first example corresponds to the prompt "A 3D rendering of a ninja in unreal engine" and the second example corresponds to the prompt "a 3D rendering of the Hulk in unreal engine". We provide the details for both experiments in the supplemental material.

For consistency with the original implementation, we preserve text2mesh's zero initialization of weights and biases in the final layer of both color and displacement MLPs, which promotes neu-

tral initial colors and displacements. We note that the final output is sensitive to this initialization.

Figure 13a and Figure 13b present results from both the original architecture and our modified version. Across different parameter settings of our model, we observe a consistent trend: our architecture prioritizes producing meshes with finer geometric resolution, successfully capturing both high-frequency geometric details and visual features. This enhanced detail preservation demonstrates our model's capacity to effectively represent multi-resolution signals.

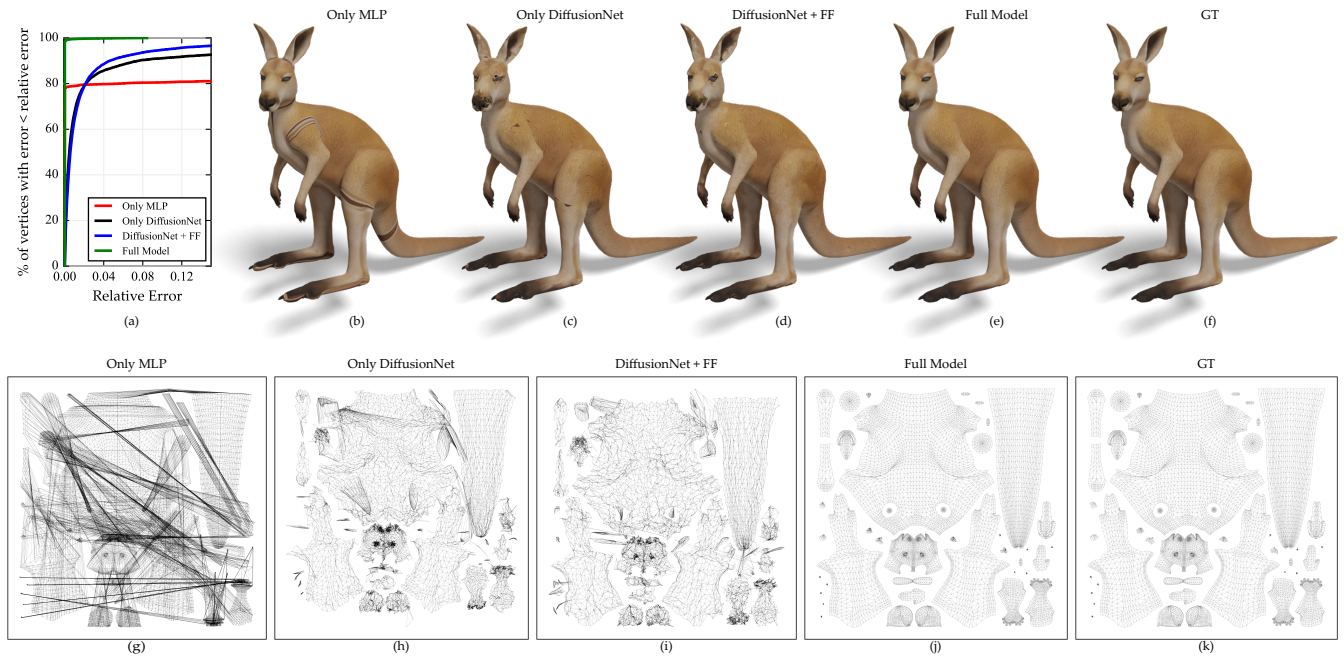


Figure 14: Ablation Study. Upper row: (a) Cumulative Distribution Functions (CDFs) of vertex errors for each of the three model variants and our full model. Texture learned by: (b) the "Only MLP" variant, (c) the "Only DiffusionNet" variant, (d) the "DiffusionNet + FF" variant, (e) our full model. (f) Ground truth textured mesh. Bottom row: Corresponding learned 2D UV coordinates for each texture. Note that our full model (e, j) is visually free from artifacts and achieves the best results both qualitatively and quantitatively, as shown in (a).

6. Ablation Study

To validate the design choices of our method, we conduct an ablation study. Inspired by the original NFFB paper, we examine three variants: our method using only the DiffusionNet output, referred to as "Only DiffusionNet"; our method incorporating both DiffusionNet and Fourier feature encoding, denoted as "DiffusionNet + FF"; and a variant using only the sine-activated MLP, referred to as "Only MLP." For the "Only MLP" variant, we experimented with different network sizes to ensure a fair comparison in terms of model complexity. Figure 14 presents the results for the UV learning experiment with the kangaroo mesh model. The upper row illustrates the textured mesh learned by each variant (b, c, d) alongside our full model (e) and the ground truth (f). On the left (a), we show the cumulative distribution functions (CDFs) of vertex errors, and the bottom row shows the corresponding 2D UV coordinates. We observe that our full model achieves superior results both qualitatively and quantitatively.

7. Conclusions

Our multi-resolution framework shows strong capability in representing neural fields on triangle meshes, achieving high precision across various domains and functions. Its detailed capture of fine features makes it ideal for high-precision tasks in computer graphics, such as UV learning, where a generally low error that suffices for applications like segmentation is not enough, and one needs to achieve close to machine precision. This framework can potentially be integrated into architectures addressing applications such as tex-

ture reconstruction from images, mesh stylization, and (as in the preliminary results that we have shown) texture and displacement generation. It also has the potential to serve as an effective feature extractor for other high-precision tasks in geometry processing.

References

- [BCGB08] BEN-CHEN M., GOTSMAN C., BUNIN G.: Conformal flattening by curvature prescription and metric scaling. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 449–458. [10](#)
- [BRGK23] BENSÂID D., ROTSTEIN N., GOLDENSTEIN N., KIMMEL R.: Partial matching of nonrigid shapes by learning piecewise smooth functions. In *Computer Graphics Forum* (2023), vol. 42, Wiley Online Library, p. e14913. [2](#)
- [C*22] CIGNONI P., ET AL.: Meshlab, 2022. [Software]. URL: <https://www.meshlab.net/>. [10](#)
- [CDGDS13] CRANE K., DE GOES F., DESBRUN M., SCHRÖDER P.: Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses*. 2013, pp. 1–126. [3](#)
- [CWKW19] COHEN T., WEILER M., KICANOGLU B., WELLING M.: Gauge equivariant convolutional networks and the icosahedral cnn. In *International conference on Machine learning* (2019), PMLR, pp. 1321–1330. [2](#)
- [CXG*22] CHEN A., XU Z., GEIGER A., YU J., SU H.: Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision* (2022), Springer, pp. 333–350. [1](#)
- [DLJ*20] DENG B., LEWIS J. P., JERUZALSKI T., PONS-MOLL G., HINTON G., NOROUZI M., TAGLIASACCHI A.: Nasa neural articulated shape approximation. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16* (2020), Springer, pp. 612–628. [1](#)

- [ESLR24] EDVAMADATHIL SIVARAM V., LI T.-M., RAMAMOORTHY R.: Neural geometry fields for meshes. In *ACM SIGGRAPH 2024 Conference Papers* (2024), pp. 1–11. [2](#)
- [GV22] GRATTAROLA D., VANDERGHEYNST P.: Generalised implicit neural representations. *Advances in Neural Information Processing Systems* 35 (2022), 30446–30458. [2](#)
- [HHF*19] HANOCKA R., HERTZ A., FISH N., GIRYES R., FLEISHMAN S., COHEN-OR D.: Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)* 38, 4 (2019), 1–12. [2](#)
- [HPT22] HAN J., PEI J., TONG H.: *Data mining: concepts and techniques*. Morgan kaufmann, 2022. [10](#)
- [HXS*21] HE T., XU Y., SAITO S., SOATTO S., TUNG T.: Arch++: Animation-ready clothed human reconstruction revisited. In *Proceedings of the IEEE/CVF international conference on computer vision* (2021), pp. 11046–11056. [1](#)
- [JGH18] JACOT A., GABRIEL F., HONGLER C.: Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems* 31 (2018). [1](#), [3](#)
- [KAL*21] KARRAS T., AITTALA M., LAINE S., HÄRKÖNEN E., HELLSTEN J., LEHTINEN J., AILA T.: Alias-free generative adversarial networks. *Advances in neural information processing systems* 34 (2021), 852–863. [2](#)
- [KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). [7](#)
- [KGM*22] KOESTLER L., GRITNER D., MOELLER M., CREMERS D., LÄHNER Z.: Intrinsic neural fields: Learning functions on manifolds. In *European Conference on Computer Vision* (2022), Springer, pp. 622–639. [1](#), [2](#)
- [Mac49] MACNEAL R. H.: *The solution of partial differential equations by means of electrical networks*. PhD thesis, California Institute of Technology, 1949. [3](#)
- [MBOL*22] MICHEL O., BAR-ON R., LIU R., BENAÏM S., HANOCKA R.: Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 13492–13502. [1](#), [3](#), [10](#)
- [MESK22] MÜLLER T., EVANS A., SCHIED C., KELLER A.: Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15. [1](#)
- [MHC24] MAHAJAN M., HOFHERR F., CREMERS D.: Meshfeat: Multi-resolution features for neural fields on meshes. In *European Conference on Computer Vision* (2024), Springer, pp. 268–285. [2](#)
- [MKK21] MITCHEL T. W., KIM V. G., KAZHDAN M.: Field convolutions for surface cnns. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 10001–10011. [2](#)
- [MLR*20] MILANO F., LOQUERCIO A., ROSINOL A., SCARAMUZZA D., CARLONE L.: Primal-dual mesh convolutional neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 952–963. [2](#)
- [MST*21] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHY R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM* 65, 1 (2021), 99–106. [1](#), [2](#)
- [OMN*19] OECHSLE M., MESCHERER L., NIEMEYER M., STRAUSS T., GEIGER A.: Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 4531–4540. [1](#)
- [Per85] PERLIN K.: An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296. [8](#)
- [PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 165–174. [1](#), [2](#)
- [PGM*19] PASZKE A., GROSS S., MASSA F., LERER A., BRADBURY J., CHANAN G., KILLEEN T., LIN Z., GIMELSHEIN N., ANTIGA L., ET AL.: Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019). [7](#)
- [PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental mathematics* 2, 1 (1993), 15–36. [3](#)
- [RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., ET AL.: Learning transferable visual models from natural language supervision. In *International conference on machine learning* (2021), PMLR, pp. 8748–8763. [10](#)
- [SACO22] SHARP N., ATTAIKI S., CRANE K., OVSIANIKOV M.: Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)* 41, 3 (2022), 1–16. [2](#), [3](#)
- [SGR*24] SHABANOV A., GOVINDARAJAN S., READING C., GOLI L., REBAIN D., YI K. M., TAGLIASACCHI A.: Banf: Band-limited neural fields for levels of detail reconstruction. *arXiv preprint arXiv:2404.13024* (2024). [1](#), [2](#)
- [Sha49] SHANNON C. E.: Communication in the presence of noise. *Proceedings of the IRE* 37, 1 (1949), 10–21. [3](#)
- [SMB*20] SITZMAN V., MARTEL J., BERGMAN A., LINDELL D., WETZSTEIN G.: Implicit neural representations with periodic activation functions. *Advances in neural information processing systems* 33 (2020), 7462–7473. [1](#), [2](#), [6](#)
- [SS21] SMIRNOV D., SOLOMON J.: Hodgenet: Learning spectral geometry on triangle meshes. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–11. [2](#)
- [SWY75] SALTON G., WONG A., YANG C.-S.: A vector space model for automatic indexing. *Communications of the ACM* 18, 11 (1975), 613–620. [10](#)
- [TLY*21] TAKIKAWA T., LITALIEN J., YIN K., KREIS K., LOOP C., NOWROUZEZAHRAI D., JACOBSON A., MCGUIRE M., FIDLER S.: 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* (2021), pp. 11358–11367. [1](#)
- [TSM*20] TANCIK M., SRINIVASAN P., MILDENHALL B., FRIDOVICH-KEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHY R., BARRON J., NG R.: Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems* 33 (2020), 7537–7547. [1](#), [3](#), [5](#), [6](#)
- [Vig22] VIGIER P.: perlin-numpy: A small library to generate perlin noise with numpy. <https://github.com/pvigier/perlin-numpy/tree/master>, 2022. Accessed: 2024-05-13. [8](#)
- [WEH20] WIERSMA R., EISEMANN E., HILDEBRANDT K.: Cnns on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (ToG)* 39, 4 (2020), 92–1. [2](#)
- [WJY23] WU Z., JIN Y., YI K. M.: Neural fourier filter bank. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 14153–14163. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#)
- [XBM*23] XUE Y., BHATNAGAR B. L., MARIN R., SARAFIANOS N., XU Y., PONS-MOLL G., TUNG T.: Nsf: Neural surface fields for human modeling from monocular depth. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 15049–15060. [2](#)
- [XTS*22] XIE Y., TAKIKAWA T., SAITO S., LITANY O., YAN S., KHAN N., TOMBARI F., TOMPKIN J., SITZMAN V., SRIDHAR S.: Neural fields in visual computing and beyond. In *Computer Graphics Forum* (2022), vol. 41, Wiley Online Library, pp. 641–676. [1](#), [2](#)
- [XXH*21] XIANG F., XU Z., HASAN M., HOLD-GEOFFROY Y., SUNKAVALLI K., SU H.: Neutex: Neural texture mapping for volumetric neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 7119–7128. [2](#)

- [YBHK21] YANG G., BELONGIE S., HARIHARAN B., KOLTUN V.: Geometry processing with neural fields. *Advances in Neural Information Processing Systems 34* (2021), 22483–22497. [1](#)
- [YBZ*22] YANG B., BAO C., ZENG J., BAO H., ZHANG Y., CUI Z., ZHANG G.: Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *European Conference on Computer Vision* (2022), Springer, pp. 597–614. [1](#)
- [YLP*20] YANG Y., LIU S., PAN H., LIU Y., TONG X.: Pfcnn: Convolutional neural networks on 3d surfaces using parallel frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 13578–13587. [2](#)