M-plicits: Neural Implicit Surfaces via Nested Multiscale Residuals

Vinícius da Silva[1]* **Isabelle Melo**[1] **Matheus Bessa**[1] **Guilherme Schardong**[2]
Luiz Schirmer[2] **André Araújo**[3] **Nuno Gonçalves**[4,6] **Hélio Lopes**[1]
Alberto Raposo[1] **Luiz Velho**[5] **Tiago Novello**[5]

[1]PUC-Rio [2]Universidade Federal de Santa Maria [3]Google DeepMind
[4]University of Coimbra [5]IMPA [6]Imprensa Nacional-Casa da Moeda (INCM), Portugal

Abstract

Encoding input coordinates with sinusoidal functions into multi-layer perceptrons (MLPs) has proven effective for implicit neural representations (INRs) of surfaces defined as zero-level sets. However, existing methods often struggle to balance training efficiency, rendering speed, and noise robustness: single-MLP approaches are expensive at inference, grid-based representations are fast but can limit surface smoothness and overfit input noise, and previous multiscale approaches frequently capture noise and produce artifacts due to hard spectral truncation. To address these limitations, we propose M-plicits, a multiscale framework that models surfaces as a residual sum of MLPs trained via a sequence of nested neighborhoods. Unlike existing residual approaches that rely on standard domain-wide sampling and require costly mesh extraction for visualization, our method strictly localizes supervision to narrow bands around the previous zero-level sets. This nested design naturally provides robustness against noisy input data: the coarse network acts as a low-pass filter that establishes a clean geometric prior, while subsequent residuals progressively refine the geometry without fitting to high-frequency artifacts. We further introduce a multiscale sphere-tracing algorithm and a GEMM-based analytical normal computation that bypasses auto-differentiation entirely, yielding high-fidelity real-time rendering. On Stanford and Thingi32, M-plicits achieves the best mean Chamfer distance in the coarse configuration and the best median Chamfer distance and IoU in the fine configuration, with substantially better noise robustness than iNGP, BACON, and IDF, while using an order of magnitude fewer parameters than grid-based baselines. Code, models, and data are available at `https://github.com/dsilvavinicius/m-plicits`.

1 Introduction

Reconstructing surfaces from point clouds is a long-standing problem in vision and graphics Kazhdan et al. [2006], with applications in augmented/virtual reality Tkach et al. [2016], digital twins Sun et al. [2005], cultural heritage preservation Scopigno et al. [2011], and autonomous robotics Whelan et al. [2016]—where high-quality 3D geometry is essential for perception, interaction, and decision-making. The emergence of high-resolution depth sensors has further motivated the development of accurate and efficient surface reconstruction methods.

Representing the surfaces as zero-level sets of multi-layer perceptrons (MLPs) has become a prominent approach, due to their strong representational capacity and the ability to incorporate geometric regularizations Wang et al. [2021]. To enhance model expressiveness, input coordinates are projected

*Corresponding author: `viniciuss@tecgraf.puc-rio.br`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

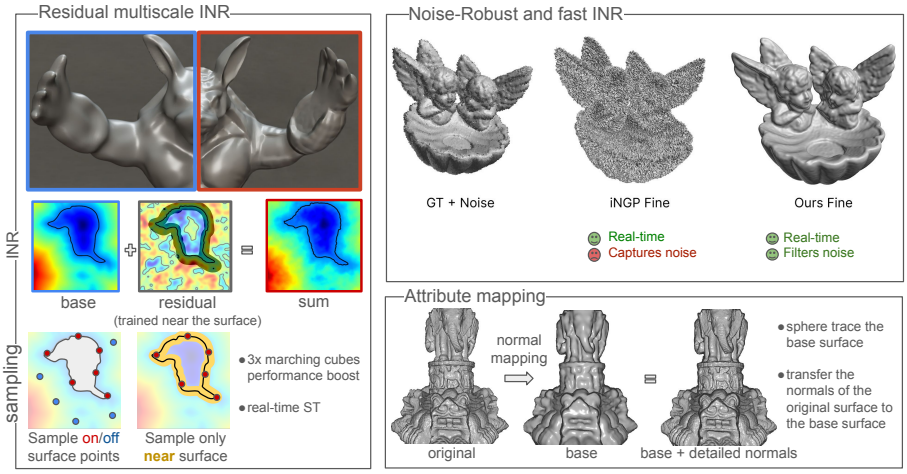


Figure 1: We introduce M-plicits, a multiscale INR framework for SDFs, based on a nested neighborhood scheme. (Left) Geometry is decomposed into base and residual SIRENs, trained at progressively finer scales with localized, near-surface sampling. (Top-right) M-plicits is noise-robust while supporting real-time sphere tracing (ST), unlike grid-based methods such as iNGP that capture noise. (Bottom-right) M-plicits supports downstream tasks such as normal mapping.

into a set of sinusoidal functions Tancik et al. [2020], Novello et al. [2025], allowing the network's bandlimit to be controlled through appropriate frequency initialization. For geometric regularization, it is common to assume that the underlying function represents a signed distance function (SDF) of the ground-truth surface Schirmer et al. [2024], which satisfies the Eikonal equation. Incorporating this constraint into the implicit neural representation (INR) loss function acts as geometric regularization, enforcing the SDF property away from the data.

Most INRs rely on a single large MLP modeling the SDF across the whole domain. Scaling up captures finer details but produces large models with prohibitive inference costs, especially for level-set extraction methods—marching cubes Lorensen and Cline [1987] or sphere tracing Hart et al. [1989]—which require dense off-surface evaluations. SIREN Sitzmann et al. [2020] requires full MLP evaluation at every query point, regardless of proximity to the surface; grid-based MLPs Müller et al. [2022] enable localized evaluations but limit surface smoothness and overfit input noise; and previous multiscale methods frequently capture noise and introduce artifacts due to hard frequency truncation.

To address these limitations, we introduce M-plicits (**M**ultiscale Im**plicit** Neural **S**urfaces), which models the SDF as a residual sum of MLPs trained within a sequence of nested neighborhoods around the previous zero-level sets. The coarse network acts as a low-pass filter that establishes a clean, noise-resilient prior, and finer residuals progressively refine geometry without fitting high-frequency artifacts. Surface attributes (normals, textures) are modeled within the same neighborhood structure; for real-time rendering we introduce multiscale sphere tracing and a GEMM-based Dongarra et al. [1990] analytical normal computation. Fig. 1 illustrates M-plicits's outputs across fitting, rendering, and downstream tasks. In summary, our contributions are:

- A compact and efficient multiscale INR model for accurate SDF representation, formulated as a residual sum of MLPs, each capturing a specific frequency band—achieving high representational capacity with fast inference for detailed surface modeling.
- A nested neighborhood training strategy that refines each residual component by supervising only near the previous zero-level set. This localized approach acts as a natural noise filter, enhancing geometric fidelity and improving data efficiency for oriented point clouds, even under noisy conditions.
- M-plicits enables fast inference through real-time multiscale sphere tracing, a GEMM-based technique for normal computation, and efficient mesh extraction with support for normal and texture mapping, delivering real-time rendering and up to a $5\times$ mesh-extraction speedup over a SIREN baseline.

2 Related Work

Implicit functions are central to graphics and vision Velho et al. [2007], Macêdo et al. [2009], Mescheder et al. [2019], with SDFs serving as a fundamental tool for modeling and manipulating surfaces Bloomenthal and Wyvill [1990], Sang et al. [2025]. MLPs have been shown effective as INRs to model SDFs Park et al. [2019], Gropp et al. [2020], including SIRENs Sitzmann et al. [2020], which use periodic activations to capture high-frequency details.

Various methods have investigated multiscale or frequency-aware training to enhance the expressiveness of INRs. BACON Lindell et al. [2021] employs multiplicative filter networks (MFNs) Fathony et al. [2020] to band-limit the spectrum of INRs. However, this approach introduces artifacts and is susceptible to capturing input noise due to hard spectral truncation, and the lack of non-linear activations limits the capacity to represent fine details with small networks. Dou et al. [2023] improves the use of MFNs by integrating a feature grid and small architectural changes. BANF Shabanov et al. [2024], MINER Saragadam et al. [2022], and MRNet Paz et al. [2023] follow a multiscale design using a Laplacian pyramid. BANF uses grid-based MLPs, which increases memory usage and introduces a dependency on spatial grids. Also, these methods supervise off-surface regions at all scales, limiting their efficiency and making real-time inference more challenging. Residual and detail decompositions also appear in the multi-view setting: HF-NeuS Wang et al. [2022b] adapts the IDF displacement composition to a NeuS pipeline, and PR-NeuS Xu et al. [2023] learns a residual on top of a data prior to accelerate convergence. Both supervise the full domain, with no nesting guarantee between levels. A complementary line suppresses spurious off-surface geometry through second-order regularizers loss-level curvature penalties developed largely for unoriented inputs, orthogonal to our representation and combinable with our per-level losses (DiGS Ben-Shabat et al. [2022], StEik Yang et al. [2023], and Neural-Singular-Hessian Wang et al. [2023]). In contrast, our approach constructs a residual sum of SIRENs, with each component supervised to capture a distinct frequency band improving geometric fidelity and data efficiency by concentrating learning near the surface. Unlike all of these prior methods, M-plicits combines a continuous (non-grid) residual representation with strictly localized nested-band supervision and a multiscale tracing / mesh-extraction inference path that exploits this nesting; Tab. 1 contrasts these four axes against the published methods used as baselines in our experiments.

Table 1: Differentiation of M-plicits from the methods compared in the experiments. M-plicits uniquely combines a continuous (non-grid) residual representation with strictly localized nested-band supervision and a multiscale tracing/extraction inference path that exploits this nesting. Citations in surrounding text.

Method	Representation	Supervision domain	Inference / tracing	Noise mechanism
BACON	Multiplicative filter network	Full domain, multiscale via architecture	Multiscale MC	Hard spectral truncation
BANF	Regular grid + interpolation kernel	Grid sample locations, full domain	Standard MC	Output kernel filter (anti-aliasing)
IDF	Coarse base MLP + displacement MLP	Full domain	Mesh extraction only	Coarse-base smoothness; off-surface blobs persist
iNGP	Multiresolution hash grid	Full domain	Hash-grid ST	None (grid-based)
M-plicits (ours)	**Coarse MLP + multi-level residual MLPs**	**Nested narrow δ_i-band per residual**	**Multiscale ST + adaptive MC**	**Coarse low-pass + nested band excludes off-surface blobs**

Traditional visualization of SDFs relies on marching cubes Lorensen and Cline [1987] or sphere tracing (ST) Hart [1996]. Performance-focused approaches such as Davies et al. [2020] leverage ST to enable real-time rendering of INR level sets. NGLOD, in particular, interpolates hierarchical features stored in a sparse voxel octree and decodes them with shallow MLPs. Although efficient, the discretized octree structure introduces gradient discontinuities across voxel boundaries, which affect normal estimation and surface smoothness. In addition, it is not designed for real-time rendering of detailed level sets. Instant-NGP Müller et al. [2022] can be viewed as an evolution of

NGLOD. It replaces the sparse octree with a multiresolution hash-grid encoding, improving memory efficiency and enabling fully GPU-parallelized training and inference. This results in significantly faster optimization and real-time rendering, making Instant-NGP a strong and relevant baseline for comparison. However, its grid-based capacity makes it prone to overfitting, often struggling to recover smooth, coherent surfaces from noisy input point clouds. In contrast, M-plicits adopts a continuous representation by modeling each frequency band explicitly with SIREN components. This avoids discretization artifacts typical of grid-based encodings, provides smooth spatial gradients by construction, and naturally filters noise through its coarse-to-fine nested architecture, improving normal estimation and level-set fidelity.

Finally, classical attribute mapping techniques, such as normal mapping Cohen et al. [1998], enhance surface detail but require explicit parameterizations and are sensitive to geometric distortions. Recent neural approaches Wang et al. [2022a] extend these ideas by propagating learned features off-surface using convolutional modules. Our method simplifies this process by avoiding interpolation entirely: inspired by variational inpainting techniques Bertalmıo et al. [2001], we regularize attribute fields to remain smooth along normals near the surface. For texture mapping, traditional parameterized approaches Catmull [1974] and neural texture fields Oechsle et al. [2019], Gao et al. [2022] require known meshes or image-depth pairs and often involve complex training. In contrast, we define texture fields directly from colored point clouds using compact MLPs, regularized along the surface, enabling fast, texture-aware rendering without dense supervision or UV mapping.

3 M-plicits: Multiscale Implicit Neural Surfaces

Our goal is to model SDFs using a multiscale INR based on a residual sum of SIRENs, where each component captures a distinct level of detail. We train these networks using a nested neighborhood strategy: each residual is supervised only near the current zero-level set, concentrating learning near the surface. This improves geometric details and enables fast inference. Additionally, we introduce an attribute mapping scheme that leverages the SDF structure to support textures and normals without relying on mesh parameterizations or interpolation. Figure 2 gives an overview of our method.

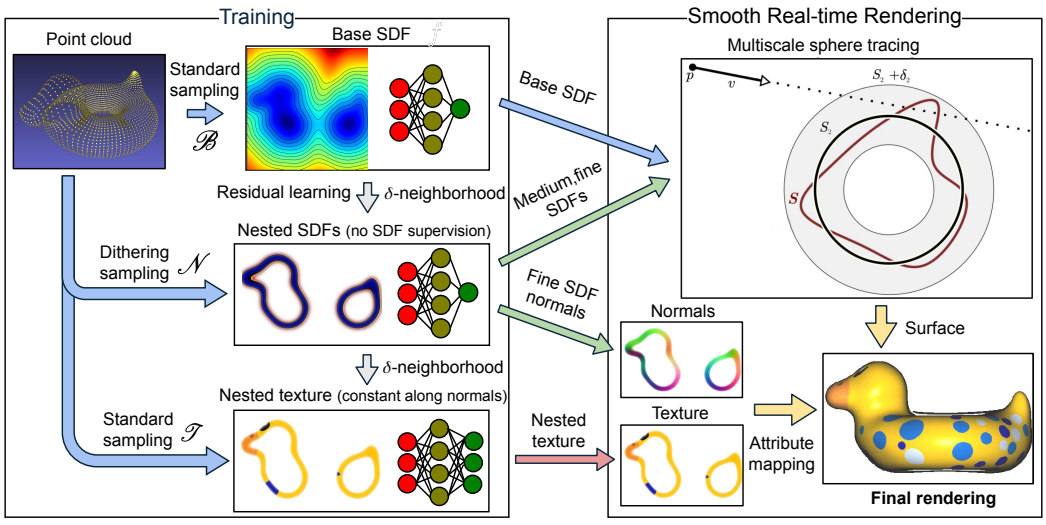


Figure 2: Overview of M-plicits. Starting from an oriented point cloud with colors, we combine sampling techniques and loss regularizations ($\mathscr{B}$ and $\mathscr{N}$) to create a base, a medium , and a fine SDF to implicitly represent the SDF in multiscale. The base SDF is defined for the entire domain, while the others are residuals, defined in (nested) neighborhoods of the surface. The colors are also trained in a neighborhood, regularized (by $\mathscr{T}$) to be constant along normals. The resulting multiscale representation can be rendered using novel sphere tracing and attribute mapping algorithms.

3.1 Preliminaries

Given an oriented point cloud $\{x_j, N_j\}_{j=1}^n$, consisting of surface points x_j and their normals N_j, our goal is to reconstruct the underlying surface S as the zero level set of a signed distance function (SDF) $f : \mathbb{R}^3 \to \mathbb{R}$, i.e., $S = f^{-1}(0) = \{x \mid f(x) = 0\}$, with the additional condition that $f(x_j) \approx 0$ and $\nabla f(x_j) \approx N_j$. To regularize the solution away from the input data, we also recall that true SDFs satisfy the Eikonal equation: $\|\nabla f(x)\| = 1$ for all $x \in \Omega$ in the training domain Ω. Enforcing this condition during training improves generalization in unsupervised regions. Finally, combining the data constraints with the Eikonal regularization leads to a loss function Gropp et al. [2020]:

$$\mathcal{L}(f) = \frac{1}{n} \sum_j \left[f(x_j)^2 + (1 - \nabla f(x_j) \cdot N_j) \right] + \int_\Omega \left(\|\nabla f(x)\| - 1 \right)^2 \, dx. \tag{1}$$

The first two terms ensure that the network fits the input points and aligns the gradient with the ground-truth normals. The third term enforces the Eikonal constraint, providing geometric regularization over Ω. To capture fine geometric detail, it is common to parameterize the SDF f using a sinusoidal MLP (SIREN). A SIREN with $n-1$ hidden layers is defined as:

$$f(x) = W_n \circ h_{n-1} \circ \cdots \circ h_0(x) + b_n, \quad h_i(x) = \sin(\omega_0(W_i x + b_i)) \,. \tag{2}$$

Here, ω_0 is a frequency parameter controlling the network's capacity to model high-frequency details Sitzmann et al. [2020], and each W_i, b_i are the learnable weight matrices and biases. This architecture enables the network to fit both surface constraints $f(x_j) \approx 0$ and normal alignment $\nabla f(x_j) \approx N_j$, while providing high representational capacity for complex geometries. However, relying on a single large MLP to represent the entire domain is inefficient, especially for high-resolution surfaces, as it leads to high computational cost during inference.

3.2 Multiscale modeling of neural SDFs

Standard neural SDFs typically rely on large MLPs to capture fine surface details, leading to fitting inefficiencies and high inference costs. To address this, we propose a multiscale SDF representation by recursively refining a coarse base network. Specifically, we define a sequence of neural SDFs $\{f_i\}_{i=1}^n$ via a sum of residual MLPs: $f_{i+1} = f_i + r_i$, for $i = 1, 2$, where f_1 is a compact base MLP trained over the full domain, and each residual r_i refines f_i by capturing higher-frequency components, modulated by the SIREN frequency parameter ω_0. We refer to the sequence $\{f_i\}$ as multiscale SDFs, each approximating the SDF of S at an increasing level of detail. While this formulation naturally extends to an arbitrary number of residuals, we focus on the case of three MLPs for simplicity and because this configuration performs well in our experiments.

To enable efficient learning and real-time inference, we impose a nesting condition between successive SDF levels. Specifically, each refined surface $S_i = f_i^{-1}(0)$, for $i = 2, 3$ is constrained to lie within a narrow band around the previous level set S_{i-1}, that is,

$$S_3 \subset \left[|f_2| < \delta_2 \right] \subset \left[|f_1| < \delta_1 \right] \tag{3}$$

Throughout, we use the bracket shorthand $\left[|f_i| < \delta_i \right]$ to denote the δ_i-band $\{x \in \Omega : |f_i(x)| < \delta_i\}$ around S_i. The inclusions in (3) hold by construction (up to marching-cubes interpolation at extraction): each residual r_{i+1} is supervised only on this set, and δ_i is set adaptively (Eq. 5) so that all input points lie within it. This is enforced during training by supervising the residual MLP r_i only within this δ_i-neighborhood, a strategy we call nested neighborhood training. This condition ensures that each residual captures localized corrections, promotes coarse-to-fine refinement, and supports efficient applications such as progressive sphere tracing and surface-aware attribute mapping (see supplementary material). The choice of δ_i is tied to the training regime and is discussed in next.

Training with nested neighborhoods. In standard SIREN-based SDF fitting, the loss in (1) is applied across the entire domain Ω. In contrast, our multiscale framework leverages the nesting condition to localize supervision. The hierarchy begins with a coarse-level SDF f_1, modeled by a compact SIREN. Finer levels $f_2, f_3, \ldots$ are added as residual SIRENs with progressively higher-frequency capacity, controlled by increasing values of the sinusoidal parameter ω_0.

Training proceeds in stages: first, the base SDF f_1 is trained over the full domain $\Omega_1 := \Omega$; then, each subsequent level f_{i+1} (for $i = 1, 2$) is trained within the restricted band $\Omega_{i+1} := \{x \in \Omega_i : |f_i(x)| < \delta_i\}$, so that each field f_i is trained on its own domain Ω_i. Each SDF f_i is optimized using a combination of data and Eikonal losses:

$$\mathcal{L}_i(f_i) = \frac{1}{n}\sum_j \left[f_i(x_j)^2 + (1 - \nabla f_i(x_j)\cdot N_j)\right] + \int_{\Omega_i} \left(\|\nabla f_i(x)\| - 1\right)^2 \, dx. \tag{4}$$

To ensure that $\{x_j\}$ lies within Ω_i at each stage i, we set the band threshold δ_i adaptively following:

$$\delta_i = (1+\varepsilon)\cdot \max_{j=1,\dots,n} |f_i(x_j)|, \text{ with } \varepsilon > 0 \text{ a small threshold.} \tag{5}$$

The maximum runs over the input surface points $\{x_j\}$: $\max_j |f_i(x_j)|$ is level i's residual fitting error at the data (small by construction) not the field's maximum over the domain. Measured band widths are reported in suppl. Sec. S3.4.

Sampling. To discretize the Eikonal term over Ω_i, we employ dithering-based sampling around the input points $\{x_j\}$, perturbing each coordinate by a random value in the interval $(-2\delta_{i-1}, 2\delta_{i-1})$. Samples falling outside Ω_i are rejected using the condition $|f_{i-1}(x)| < \delta_{i-1}$; see Fig. 3(a).

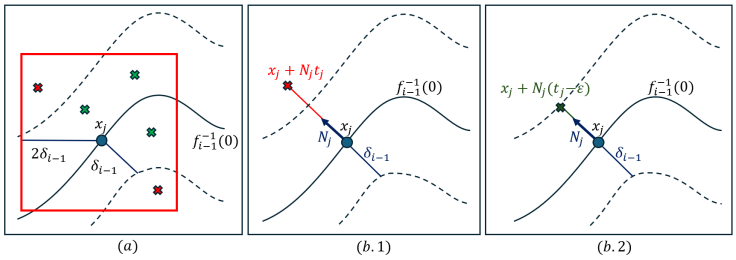


Figure 3: **(a)** Dithering-based sampling around each input point x_j, where each coordinate is perturbed by a random value in $(-2\delta_{i-1}, 2\delta_{i-1})$, followed by filtering to retain only points in the valid band Ω_i (green points denote those that are kept and red those rejected). **(b.1)** Computation of the displacement vector $t_j N_j$, where the offset point $x_j + t_j N_j$ lies just outside the narrow band Ω_i. **(b.2)** Final accepted sample $x_j + N_j(t_j - \varepsilon)$, located inside Ω_i, used to supervise the data term.

Adding extra samples to accelerate SDF training is a common strategy Novello et al. [2022], but it often requires evaluating a large MLP over the entire domain, leading to expensive inference. In contrast, M-plicits uses residual MLPs that are trained only within a narrow band around the input data, allowing for efficient sampling. To enrich supervision of the data term in (4), we propose sampling along the normal N_j of each point x_j; see Fig. 3(b). Specifically, we compute a scalar $t_j \leq \delta_{i-1}$ such that for all $t \in [0, t_j]$, the offset point $x_j + tN_j$ lies at a distance t from the surface. This allows us to supervise both the SDF value and its gradient: $f_i(x_j + tN_j) = t$ and $\nabla f_i(x_j + tN_j) = N_j$, providing richer sampling near the surface. We determine each t_j via an iterative scheme: starting from $t_j = \delta_{i-1}$, we reduce it by a small step until the distance from $x_j + t_j N_j$ to the point cloud $\{x_j\}$ equals t_j, ensuring that the offset point lies inside the band.

Rendering and mesh extraction. To render the zero level set $f^{-1}(0)$ of an SDF f, it is common to use either sphere tracing (ST) or marching cubes for mesh extraction followed by standard mesh rendering. When the SDF is represented using M-plicits, both strategies become more efficient.

We first introduce a **multiscale ST**. Given a view ray $\gamma(t) = p_0 + tv$, with origin at a point p_0 and direction v, intersecting $f^{-1}(0)$, standard ST approximates the first intersection point by iterating $p_{i+1} = p_i + vf(p_i)$ along γ. However, querying a large MLP at each step may be expensive. To reduce this cost, we exploit the multiscale SDF hierarchy $\{f_i\}$, using coarser networks to guide early steps. Thanks to the nesting condition in 3, coarse levels can be used to trace offset surfaces before switching to finer levels near the surface. The ray starts tracing $f_1^{-1}(\delta_1)$ with f_1, then proceeds to $f_2^{-1}(\delta_2)$ using f_2, and finally reaches the target surface $f_3^{-1}(0)$ with f_3. Each coarser level uses offset tracing via $p_{i+1} = p_i + v\,(f_j(p_i) - \delta_j)$, ensuring convergence avoiding high-cost evaluations. The values δ_i (5) play a crucial role in rendering. Using distinct values at each level helps prevent issues such as missed ray-surface intersections: if δ_1 is too small, parts of $f_2^{-1}(0)$ might lie outside the region bounded by $f_1^{-1}(\delta_1)$; Fig. 4 shows some of these issues.

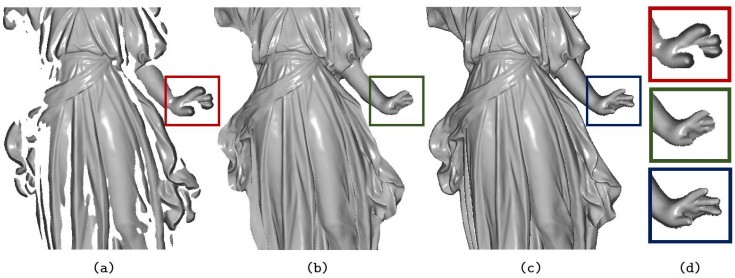


Figure 4: (a) Using too large $\delta_1 = \delta_2$ may result in holes (the ray does not reach the surface). More iterations would be needed using the finer SDF to fill those holes, defeating the idea of minimizing iterations. (b) Conversely, reducing the deltas $\delta_1 = \delta_2$ may miss parts of the silhouette since the target surface may not be inside the previous neighborhood (the hand). (c) Using δ_1 and δ_2 suited for the nesting condition implies no holes and better silhouette capture.

M-plicits also accelerates mesh extraction using **marching cubes**. We adopt an adaptive grid inference strategy by first evaluating the coarse SDF f_1 to cull grid vertices, and querying finer SDFs only for vertices inside the δ_1-neighborhood. This reduces the number of voxel evaluations, thereby accelerating mesh extraction (Fig. 5).

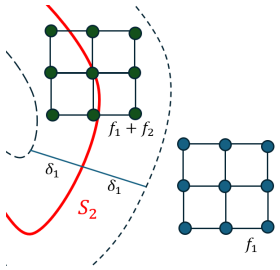


Figure 5: Adaptive marching cubes. For grid vertices outside the δ_1 neighborhood (blue), only the coarse SDF f_1 is evaluated. For points in the neighborhood (green) the residual f_2 is added.

Normal and texture mapping. Let S be a surface nested within a δ-neighborhood of the zero-level set of a neural SDF f. Assuming f is a finer-level neural SDF, we define the **neural normal mapping** by assigning to each point $p \in S$ the attribute $g(p) := \nabla f(p)$. If S corresponds to the zero-level set of a coarser neural SDF, this mapping allows us to bypass additional sphere tracing iterations, reducing computational overhead.

Similarly, we define a network $g\colon \mathbb{R}^3 \to \mathcal{C}$ to encode a **texture** within the δ-neighborhood of f, where $\mathcal{C}$ is the RGB color space. We refer to the attribute mapping defined by the triple $\{S, f, g\}$ as a **neural texture mapping**. To train the parameters ϕ of g, we optimize $\mathcal{T}(\phi) = \int_{f^{-1}(0)} (g - \cancel{g})^2 \, dx + \int_{[|f| \leq \delta]} \langle \nabla g, \nabla f \rangle^2 \, dx$, where the first term ensures that g fits the ground-truth texture $\cancel{g}$, while the second term regularizes g to remain constant along the gradient flow of f, effectively propagating texture information throughout the δ-neighborhood.

GEMM-based normal calculation. To ensure real-time rendering performance, we compute normals without the need of auto-differentiation nor computational graphs. It works as a forward pass on the MLP and is implemented on the GPU using only a GEMM library, resulting in a 2X performance improvement over `torch.autograd`. Details are in the supplementary material.

4 Experiments

This section presents experiments to evaluate the proposed method comprehensively, both in comparison to representative multiscale and grid-based neural SDF methods and with ablations designed to understand the importance of its different components. We also demonstrate several applications of our method, showcasing its versatility.

Implementation details. All experiments are conducted on an NVIDIA RTX 5090 with 32 GB. For sphere tracing, we fix the number of iterations to 20 for coarse and 5 for each residual level, for better control of the parallelism. The ω_0 SIREN parameters are 30 for coarse, 45 for medium, and 100 for fine. We use PyTorch's Adam optimizer for training Paszke et al. [2019].

Evaluation protocol. All input point clouds were centered and normalized to the unit sphere. For evaluation, we extracted the zero-level set of each Neural SDF using marching cubes at a resolution of 512^3. We then uniformly sampled 500K points on the reconstructed meshes and computed the L2 Chamfer Distance (CD), and the Intersection over Union (IoU) against the input point cloud using PyTorch3D's implementation Ravi et al. [2020].

Table 2: Surface reconstruction on Stanford + Thingi32 datasets. Best values are **bold**, second best are underlined, and third best are *italic*. Chamfer Distance (CD) considers 500K samples. Sampling time is measured on a 512^3 grid; for our method this uses the adaptive multiscale marching cubes (Sec. 3). Renderer FPS considers 512^2 images and uses multiscale sphere tracing on geometry only (without normal mapping unless explicitly noted in Tab. 4).

Method	mean CD ↓	median CD ↓	mean IoU ↑	median IoU ↑	# params ↓	avg Training (min) ↓	avg Sampling (s) ↓	Renderer FPS ↑
iNGP coarse	*6.87E-05*	6.43E-05	4.12E-01	4.05E-01	2,040,864	8.22	0.40	*86*
iNGP fine	6.44E-05	*6.04E-05*	*4.20E-01*	*3.90E-01*	9,113,760	10.59	*0.89*	88
IDF	6.13E-02	7.74E-04	4.09E-01	9.20E-02	1,191,943	*8.26*	42.38	N/A
BACON	8.12E-03	2.18E-05	5.15E-01	7.32E-01	*530,953*	8.43	**0.24**	N/A
Ours coarse	**5.36E-05**	3.80E-05	4.48E-01	4.59E-01	**17,153**	**4.07**	1.51	**180**
Ours fine	6.57E-05	**1.87E-05**	**5.86E-01**	**8.67E-01**	246,627	8.53	5.29	43

MLP notation. (N, d) refers to a MLP with d hidden layers of the form $\mathbb{R}^N \to \mathbb{R}^N$. Additionally, $(128, 2) \triangleright (256, 2) \triangleright (400, 2)$ refers to a multiscale SDF with coarse, medium, and fine MLPs with two $\mathbb{R}^{128} \to \mathbb{R}^{128}$, $\mathbb{R}^{256} \to \mathbb{R}^{256}$, and $\mathbb{R}^{400} \to \mathbb{R}^{400}$ hidden layers, respectively. This is our setup for all experiments unless otherwise stated.

4.1 Main results

Surface reconstruction. We compare our neighborhood nesting approach against representative multiscale and grid-based neural SDF methods for surface representation on the Stanford and Thingi32 datasets Curless and Levoy [1996], Zhou and Jacobson [2016], Takikawa et al. [2021]. We include Instant-NGP (iNGP) Müller et al. [2022], focusing on real-time rendering performance, Implicit Displacement Fields (IDF) Wang et al. [2022a], which disentangles shape and detail, and BACON Lindell et al. [2021], a multiscale representation where we report results from its 8th hidden layer. We do not include NGLOD Takikawa et al. [2021] as a separate baseline because Instant-NGP supersedes it on this task: iNGP replaces NGLOD's sparse-voxel octree with a multiresolution hash-grid encoding that improves memory efficiency, training speed, and rendering quality, so the iNGP comparison already covers the strongest representative of this lineage. For BANF Shabanov et al. [2024], the public implementation does not provide a pipeline for surface reconstruction from point clouds (the released code targets image fitting, NeRFing, and multi-view 3D reconstruction); we therefore report a comparison against our own faithful re-implementation separately, in suppl. Sec. S2.4, with its full configuration disclosed. Its numbers reflect a custom implementation rather than the authors' code. We additionally compare against classical Screened Poisson reconstruction Kazhdan and Hoppe [2013] in suppl. Sec. S2.3.

Tab. 2 summarizes the results. We evaluate geometric fidelity using Chamfer Distance (CD), volumetric consistency using Intersection over Union (IoU), computational cost, and rendering throughput. While CD measures pointwise surface deviation, IoU reflects global occupancy agreement and structural correctness.

In the coarse configuration, our method achieves the best mean CD while also improving IoU over both iNGP variants, indicating better global geometric alignment rather than merely local surface fitting (Fig. 6). In the fine configuration, mean CD remains comparable to iNGP fine, while our method achieves the best median CD and significantly higher IoU (Fig. 7). Most reconstructions closely match the ground truth, with only a few challenging Thingi32 cases (non-watertight surfaces, multiple components, thin structures, topological defects; see suppl. Fig. S11) affecting the mean.

Qualitatively, iNGP recovers slightly sharper high-frequency details but often introduces a geometric offset relative to the ground truth (suppl. Fig. S12). We acknowledge this trade-off explicitly: the smoothness prior in the coarse SDF favors faithful global geometry over the highest-frequency micro-detail, and our neural normal mapping (Sec. 4.2, Fig. 9) partially mitigates it by transferring fine-frequency normals from a higher-capacity SDF onto the coarse geometry without rendering-cost penalty. IDF frequently produces unstable surfaces with blob-like artifacts (suppl. Fig. S13) and requires explicit mesh extraction, preventing direct real-time rendering.

Trade-offs and design space. No single method in Tab. 2 dominates all axes. iNGP fine attains the second-best mean CD and real-time FPS but is highly noise-sensitive (Tab. 3: mean CD 2.80×10^{-4}

vs. ours 3.36×10^{-5}). BACON has the fastest sampling and competitive median CD/IoU but is also noise-sensitive (mean CD 1.68×10^{-3}) and provides no real-time rendering. M-plicits is the only method in the (noise-robust, real-time) region, achieving the best mean CD in coarse, the best median CD/IoU in fine, and the best noise robustness across configurations, while remaining roughly an order of magnitude smaller in parameter count than grid-based baselines and supporting native real-time rendering (180 FPS coarse, 35–43 FPS fine, at 512^2; Tabs. 2 and 4). Per-mesh CD distributions in suppl. Figs. S14, S15, S17 and S18 are tightly concentrated around the mean for M-plicits and substantially wider for BACON and IDF, indicating more stable reconstruction across shapes.



Figure 6: Coarse reconstruction of the Armadillo (Stanford). Ours vs. iNGP.



Figure 7: Qualitative comparison on a Thingi32 sample. (a) GT; (b) ours (fine); (c) iNGP fine; (d) BACON, showing blob-like artifacts; (e) IDF. All with their respectives CD values.

Robustness to Noise. We evaluate noise robustness on a perturbed version of Stanford and Thingi32 in which vertices are randomly displaced along the ground-truth normal by up to 1.0% of the bounding box. The coarse level of M-plicits acts as a low-pass filter, removing high-frequency noise and providing a clean geometric prior for subsequent residual levels (Fig. 8). In contrast, iNGP and BACON absorb noise into the reconstructed surface; IDF avoids surface noise but produces unstable blobs around it, lowering its mean CD (Tab. 3).

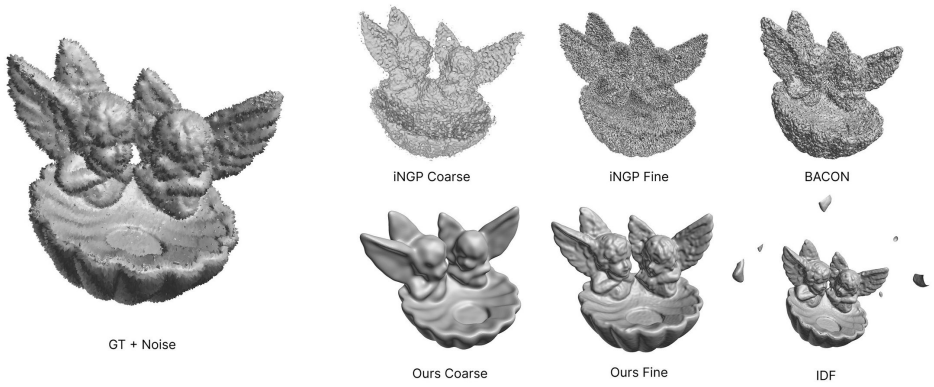


Figure 8: Qualitative comparison of surface reconstruction on sample 354371 with 1% noise (Thingi32). Left: GT with noise. Right: reconstructions. IDF is the only baseline that does not retain noise on the surface, but it generates blobs around it.

Table 3: Quantitative robustness to noise. CD (↓) and IoU (↑) for Stanford and Thingi32 under vertex noise. **Bold**, *italic*, underlined = best, second, third.

Method	mean CD↓	median CD↓	mean IoU↑	median IoU↑
Ours coarse	7.42E-05	5.33E-05	3.81E-01	4.03E-01
Ours medium	*4.62E-05*	3.03E-05	*5.37E-01*	**6.38E-01**
Ours fine	**3.36E-05**	**1.62E-05**	**5.82E-01**	*5.91E-01*
iNGP coarse	6.70E-04	3.76E-04	1.90E-01	1.88E-01
iNGP fine	2.80E-04	1.09E-04	3.09E-01	3.05E-01
IDF	8.37E-02	*2.49E-05*	3.22E-01	3.08E-01
BACON	1.68E-03	1.51E-04	1.76E-01	1.65E-01

4.2 Ablations and additional experiments

Isolating residuals and nested bands. Suppl. Sec. S3.1 adds an ablation isolating our two design axes (the residual composition and the nested-band supervision) against variants trained with full-domain residuals and with independent per-level SDFs, together with robustness studies of coarse-stage under-training (suppl. Sec. S3.2) and outlier contamination (suppl. Sec. S3.3).

Neural normal mapping and multiscale ST. Fig. 9 shows that neural normal mapping adds high-frequency detail to a coarse SDF without rendering-cost penalty (left), and that adding multiscale ST iterations on a medium-level SDF further refines the silhouette (right). A broader evaluation across other models is in the appendix.

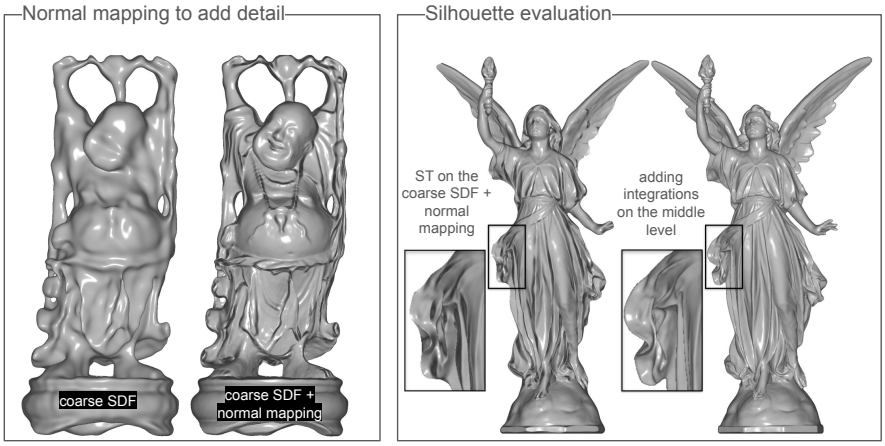


Figure 9: Left: neural normal mapping from a finer SDF adds high-frequency detail to a coarse $(64, 1)$ SDF (Buddha). Right: silhouette evaluation (Lucy); compared to sphere tracing the coarse SDF alone with normal mapping, adding ST iterations on a medium $(256, 2)$ level refines the silhouette.

Table 4: Real-time CUDA renderer ablation (multiscale ST, GEMM normals, neural normal mapping). 20 ST iterations on the first SDF, 5 on each subsequent level. (NM) denotes normal mapping without additional ST on the last SDF. Images 512^2; size in KB.

Model	FPS	Speedup	Size
$(400, 4)$ (SIREN baseline)	19	1.0×	1899
$(128, 2)$ (coarse)	180	9.5×	68
$(128, 2) \triangleright (256, 2)$ (NM)	85	4.5×	331
$(128, 2) \triangleright (256, 2)$	70	3.7×	331
$(128, 2) \triangleright (400, 2)$ (NM)	47	2.5×	704
$(128, 2) \triangleright (400, 2)$	38	2.0×	704
$(128, 2) \triangleright (256, 2) \triangleright (400, 2)$ (NM)	43	2.3×	967
$(128, 2) \triangleright (256, 2) \triangleright (400, 2)$	35	1.8×	967

Table 5: Marching cubes runtime (s) on Stanford. Baseline: SIREN $(256, 4)$. Multiscale: $(64, 1) \triangleright (128, 2) \triangleright (400, 2)$. Up to $\mathbf{5}\times$ speedup.

Model	Baseline	No cull.	Cull.	Speedup
Arm.	4.87	4.10	0.99	4.92×
Lucy	4.87	7.49	1.07	4.56×
Dragon	4.88	7.28	1.71	2.85×
Thai	4.89	7.28	1.73	2.82×

Real-time renderer and mesh extraction. A CUDA renderer using multiscale ST and GEMM-based normals (via CUTLASS) achieves real-time performance, with neural normal mapping yielding substantial speedups over the SIREN baseline (Tab. 4). For mesh extraction (Tab. 5), our adaptive grid evaluation skips inference at finer levels far from the level set, achieving up to $5\times$ speedup; smaller surfaces benefit more, since their nesting neighborhoods contain fewer vertices.

5 Conclusion

We propose M-plicits, an INR framework modeling neural SDFs as residual SIRENs trained on nested neighborhoods, with multiscale sphere tracing, neural attribute mapping, and GEMM-based analytical normals enabling real-time rendering and accelerated marching-cubes mesh extraction. The nested-band design acts as an implicit regularizer yielding strong noise robustness and extends naturally to differentiable and time-dependent pipelines. *Limitations:* smoothness prevents very sharp edges (addressable with local features), and the maximum in Eq. 5 makes the band width sensitive to isolated outlier points. A quantile-based δ is a natural robustification (suppl. Sec. S3.3), while the adaptive δ keeps the hierarchy robust to the coarse stage's training budget (suppl. Sec. S3.2). Future directions include inverse rendering, unsigned distance functions, and fully fused GEMMs Müller [2021].

References

Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. Digs: Divergence guided shape implicit neural representation for unoriented point clouds. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19301–19310. IEEE, 2022.

Marcelo Bertalmıo, Li-Tien Cheng, Stanley Osher, and Guillermo Sapiro. Variational problems and partial differential equations on implicit surfaces. *Journal of Computational Physics*, 174(2): 759–780, 2001.

Jules Bloomenthal and Brian Wyvill. Interactive techniques for implicit modeling. *ACM Siggraph Computer Graphics*, 24(2):109–116, 1990.

Edwin Earl Catmull. *A subdivision algorithm for computer display of curved surfaces*. The University of Utah, 1974.

Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 115–122, 1998.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes. *arXiv preprint arXiv:2009.09808*, 2020. URL `http://arxiv.org/abs/2009.09808`.

Jack J Dongarra, Jeremy Du Croz, Sven Hammarling, and Iain S Duff. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software (TOMS)*, 16(1):1–17, 1990.

Yishun Dou, Zhong Zheng, Qiaoqiao Jin, and Bingbing Ni. Multiplicative fourier level of detail. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1808–1817, 2023.

Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International Conference on Learning Representations*, 2020.

Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

John C Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

John C Hart, Daniel J Sandin, and Louis H Kauffman. Ray tracing deterministic 3-D fractals. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, pages 289–296, 1989.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanaes. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, 2013.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

David B. Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. BACON: Band-limited Coordinate Networks for Multiscale Scene Representation. *arXiv preprint arXiv:2112.04645*, 2021. URL `http://arxiv.org/abs/2112.04645`.

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

Ives Macêdo, João Paulo Gois, and Luiz Velho. Hermite interpolation of implicit surfaces with radial basis functions. In *2009 XXII Brazilian Symposium on Computer Graphics and Image Processing*, pages 1–8, 2009. doi: 10.1109/SIBGRAPI.2009.11.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

Thomas Müller. Tiny CUDA neural network framework, 2021. https://github.com/nvlabs/tiny-cuda-nn.

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

Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinícius da Silva, Hélio Lopes, and Luiz Velho. Exploring differential geometry in neural implicits. *Computers & Graphics*, 108, 2022. ISSN 0097-8493. doi: https://doi.org/10.1016/j.cag.2022.09.003. URL `https://dsilvavinicius.github.io/differential_geometry_in_neural_implicits/`.

Tiago Novello, Vinícius da Silva, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Neural implicit surface evolution. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

Tiago Novello, Diana Aldana, Andre Araujo, and Luiz Velho. Tuning the frequencies: Robust training for sinusoidal neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of CVPR*, pages 165–174, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL `https://arxiv.org/abs/1912.01703`.

Hallison Paz, Daniel Perazzo, Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinícius Da Silva, Daniel Yukimura, Fabio Chagas, Hélio Lopes, and Luiz Velho. Mr-net: Multiresolution sinusoidal neural networks. *Computers & Graphics*, 114:387–400, 2023.

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

Lu Sang, Zehranaz Canfes, Dongliang Cao, Florian Bernard, and Daniel Cremers. Implicit neural surface deformation with explicit velocity fields, 2025. URL `https://arxiv.org/abs/2501.14038`.

Vishwanath Saragadam, Jasper Tan, Guha Balakrishnan, Richard G Baraniuk, and Ashok Veeraraghavan. Miner: Multiscale implicit neural representation. In *European Conference on Computer Vision*, pages 318–333. Springer, 2022.

Luiz Schirmer, Tiago Novello, Vinícius da Silva, Guilherme Schardong, Daniel Perazzo, Hélio Lopes, Nuno Gonçalves, and Luiz Velho. Geometric implicit neural representations for signed distance functions. *Computers & Graphics*, 125:104085, 2024.

Roberto Scopigno, Marco Callieri, Paolo Cignoni, Massimiliano Corsini, Matteo Dellepiane, Federico Ponchio, and Guido Ranzuglia. 3d models for cultural heritage: Beyond plain visualization. *Computer*, 44(7):48–55, 2011.

Akhmedkhan Shabanov, Shrisudhan Govindarajan, Cody Reading, Lily Goli, Daniel Rebain, Kwang Moo Yi, and Andrea Tagliasacchi. Banf: Band-limited neural fields for levels of detail reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20571–20580, 2024.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

Wei Sun, Binil Starly, Jae Nam, and Andrew Darling. Bio-cad modeling and its applications in computer-aided tissue engineering. *Computer-aided design*, 37(11):1097–1114, 2005.

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 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*, pages 11353–11362. IEEE, jun 2021. ISBN 978-1-6654-4509-2. doi: 10.1109/CVPR46437.2021.01120. URL `https://ieeexplore.ieee.org/document/9578205/`.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7537–7547. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/55053683268957697aa39fba6f231c68-Paper.pdf`.

Anastasia Tkach, Mark Pauly, and Andrea Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Transactions on Graphics (ToG)*, 35(6):1–11, 2016.

Luiz Velho, Jonas Gomes, and Luiz H de Figueiredo. *Implicit objects in computer graphics*. Springer Science & Business Media, 2007.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

Yifan Wang, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields. In *The Tenth International Conference on Learning Representations*. OpenReview, 2022a.

Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Hf-neus: Improved surface reconstruction using high-frequency details. In *Advances in Neural Information Processing Systems*, volume 35, pages 1966–1978, 2022b.

Zixiong Wang, Yunxiao Zhang, Rui Xu, Fan Zhang, Peng-Shuai Wang, Shuangmin Chen, Shiqing Xin, Wenping Wang, and Changhe Tu. Neural-singular-hessian: Implicit neural representation of unoriented point clouds by enforcing singular hessian. *ACM Transactions on Graphics*, 42(6): 1–14, 2023.

Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

Jianyao Xu, Qingshan Xu, Xinyao Liao, Wanjuan Su, Chen Zhang, Yew-Soon Ong, and Wenbing Tao. Pr-neus: A prior-based residual learning paradigm for fast multi-view neural surface reconstruction. *arXiv preprint arXiv:2312.11577*, 2023.

Huizong Yang, Yuxin Sun, Ganesh Sundaramoorthi, and Anthony Yezzi. Steik: Stabilizing the optimization of neural signed distance functions and finer shape representation. In *Advances in Neural Information Processing Systems*, volume 36, pages 13993–14004, 2023.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10371–10381, June 2024.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *European Conference on Computer Vision*, 2024.

Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.

Supplementary Material

Table of Contents

S1 Additional qualitative comparisons

The following figures complement the main-paper qualitative comparisons (Figs. 6 and 7) with additional illustrative cases referenced in Sec. 4.1 of the main paper.

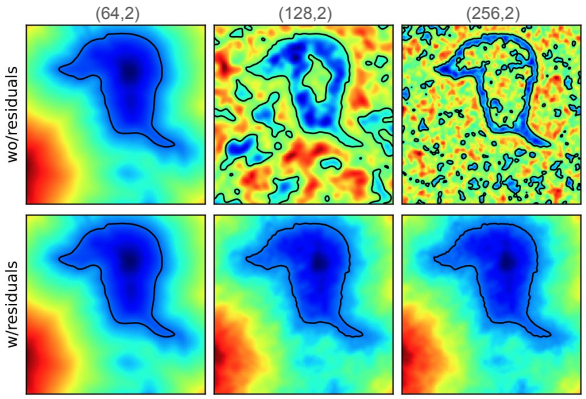


Figure S10: Evaluation of the residual approach. Training the SDFs in the neighborhoods (first row: center, right) results in spurious components outside the region. Using the residual approach eliminates those components (second row: center, right).

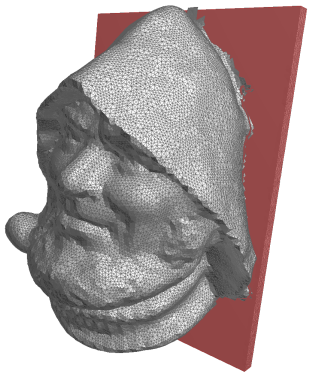


Figure S11: Sample from the Thingi32 dataset. The red region corresponds to a separate connected component from the gray surface. Such disconnected structures violate common assumptions of SDF-based representations, making them challenging for methods relying on a single continuous signed distance field.



Figure S12: Qualitative comparison on a sample from the Thingi32 dataset. (a) Ground-truth reference, (b) our method (fine), and (c) Instant-NGP (fine). iNGP recovers slightly sharper high-frequency details but often introduces a geometric offset relative to the ground truth, which becomes apparent under direct alignment with the reference mesh.

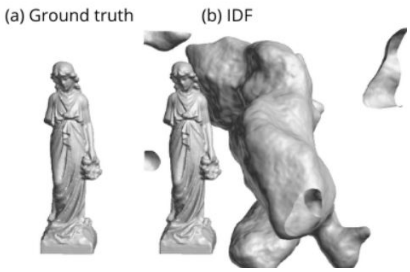


Figure S13: Instability observed in IDF reconstructions. (a) Ground-truth reference and (b) corresponding IDF reconstruction; the method generates blob-like artifacts instead of a coherent surface, indicating reduced robustness on complex geometries.

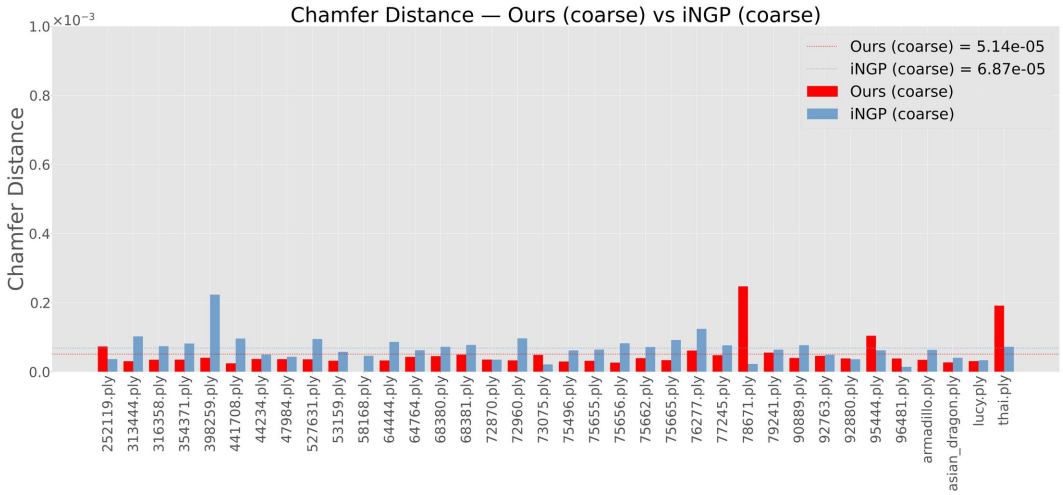


Figure S14: Comparison of Ours Coarse with iNGP Coarse.

S2 Comparison details

This section presents detailed model-level results comparisons to enable a deeper analysis of the performance for the evaluated approaches.

S2.1 Per-model metrics

In the results section of the main paper, we discussed the evaluation of the mean and median performance of coarse and fine representations across several approaches, including the proposed M-plicits. Here, we further present the per-model Chamfer Distance values obtained for each approach.

Figures S14 and S15 show the per-model Chamfer Distance comparison between M-plicits and iNGP Müller et al. [2022] for the coarse and fine representations, respectively. Figures S17 and S18 present the corresponding results for BACON Lindell et al. [2021] and IDF Wang et al. [2022a].

In Figure S14, the Chamfer Distance values obtained by M-plicits are strongly concentrated around the mean and median, with only a few clearly visible outliers. Notably, the model 78671.ply represents a clear outlier among our coarse reconstructions. As shown in Figure S16, this behavior is most likely caused by the presence of a blob-like artifact generated during reconstruction, which significantly increases the geometric deviation from the ground truth surface.

When comparing these results with those shown in Figures S17 and S18, it becomes evident that the Chamfer Distance values obtained by BACON and IDF exhibit a wider dispersion across the dataset. This larger variability indicates less consistent reconstruction quality, with several models presenting higher geometric error. In contrast, the results obtained by M-plicits remain more tightly clustered, indicating a more stable reconstruction behavior.

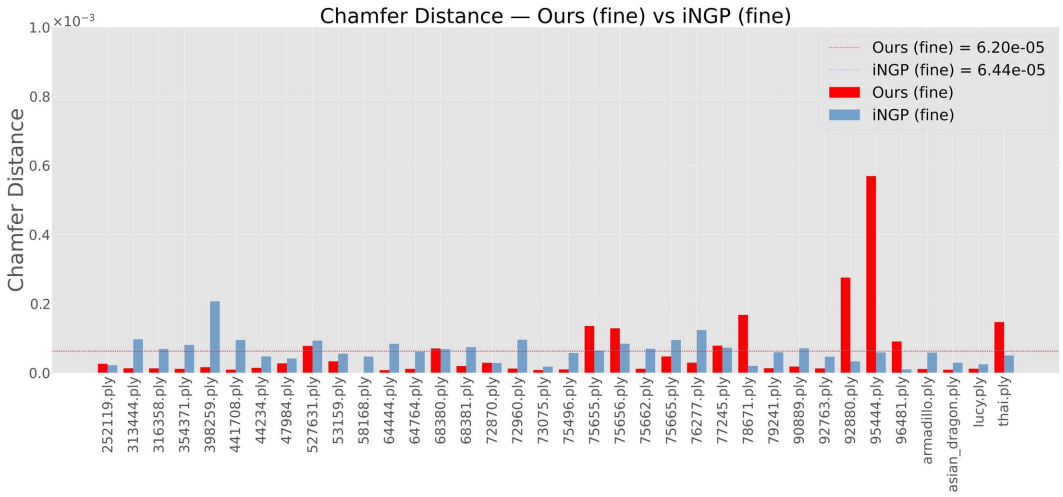


Figure S15: Comparison of Ours Fine with iNGP Fine.



Figure S16: Reconstruction for 78671.ply (ours Coarse vs iNGP Coarse).

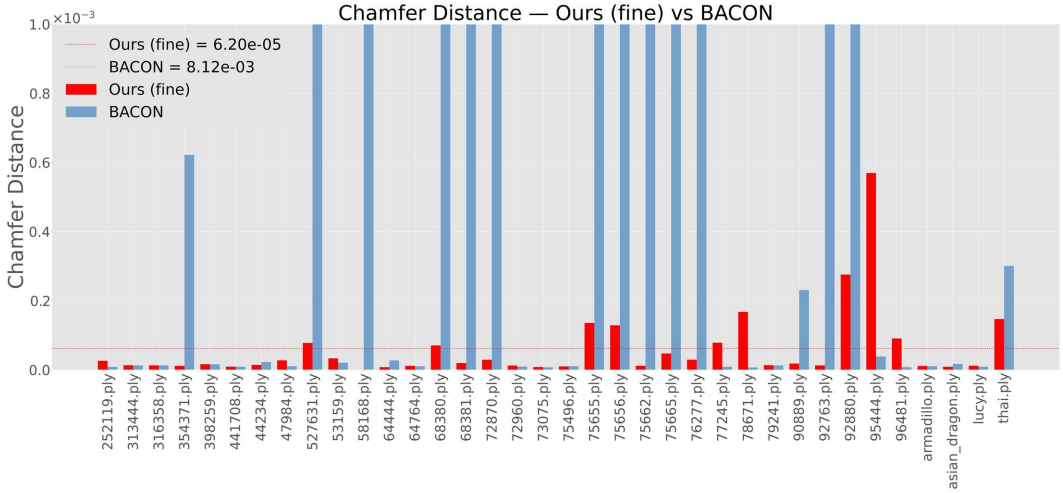


Figure S17: Comparison of Ours Fine with BACON.

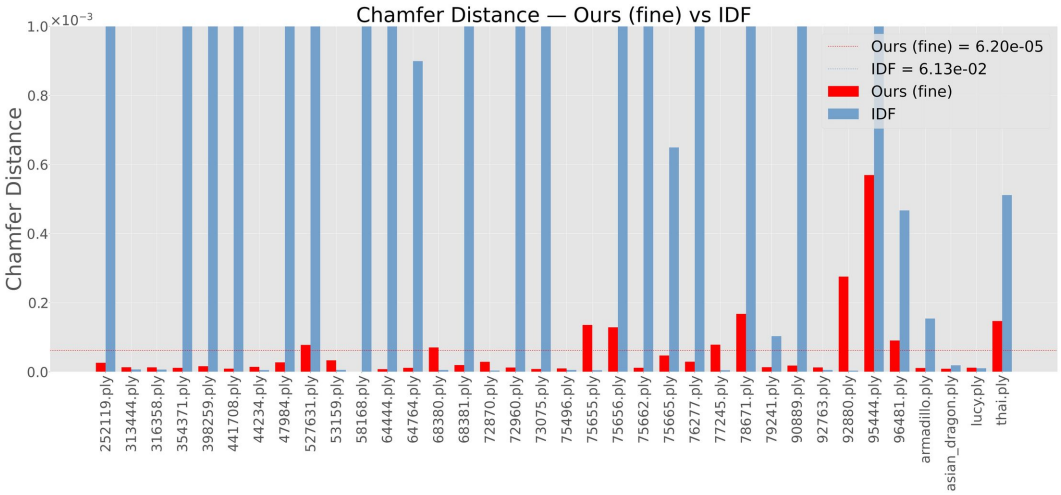


Figure S18: Comparison of Ours Fine with IDF.

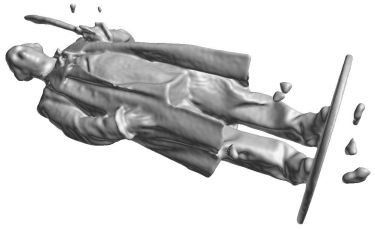


Figure S19: Reconstruction for 95444.ply (ours Fine).

Overall, these results suggest that the proposed coarse generation approach provides improved robustness, producing more consistent reconstructions with lower geometric error when compared to iNGP, BACON, and IDF.

In Figure S15, a larger variation in Chamfer Distance values can be observed when compared to the coarse case. Although most reconstructions still present relatively low Chamfer Distance values, the distribution becomes slightly wider for both methods. One factor contributing to this variability is the nature of the Thingi32 dataset itself. Several meshes in this dataset contain geometric characteristics that are not ideal for training Signed Distance Functions, such as multiple disconnected components, non-watertight surfaces, and self-intersections. These properties introduce additional difficulty for implicit reconstruction methods and can lead to localized instability during training. It is also important to note that Table 1 in the main paper shows that the fine case presents improvements in the IoU metric.

Another factor that may contribute to the observed variation is the need for a more refined adjustment of the M-plicits parameters for specific objects. In this study, the training procedure was performed using a unified set of parameters for all meshes in the dataset. While this configuration provides a fair and consistent evaluation across models, it does not necessarily yield the optimal reconstruction for every individual shape.

An example of this behavior can be observed for the model 95444, shown in Figure S19. In this case, the reconstructed mesh presents blob-like artifacts, which increase the geometric deviation from the ground truth surface. These artifacts could likely be reduced with a more careful tuning of the model parameters specifically for this object.

Despite these challenges, M-plicits achieves performance comparable to iNGP in terms of geometric robustness and spatial fidelity. The majority of reconstructions remain within the same error scale, indicating that the proposed method is capable of maintaining stable reconstructions even under imperfect training conditions. The next section also shows that our approach is robust to input noise, while iNGP absorbs it.

In contrast, the results presented for BACON and IDF continue to show a higher degree of variability in Chamfer Distance values across the dataset. This behavior suggests that these approaches are more sensitive to geometric irregularities present in the input meshes, leading to less consistent reconstruction quality when compared to the proposed method. Additionally, when compared to our fine model, the obtained Chamfer Distance values tend to be higher, indicating comparatively lower reconstruction accuracy.

S2.2 Per-Model Metrics: Noise Robustness Analysis

Expanding upon the noise robustness discussion presented in the main paper, this section provides a detailed, per-model analysis of the Chamfer Distance for each evaluated approach under noisy conditions.

In M-plicits, the initial coarse level effectively acts as a low-pass filter, significantly attenuating high-frequency noise. This establishes a clean and robust base representation for the subsequent residual levels. As illustrated in Figure S20, reconstruction accuracy improves progressively through the hierarchy. The coarse level yields a mean Chamfer Distance of 7.42e-05, which decreases to 4.62e-05 at the medium level, and further improves to 3.36e-05 at the fine level. This progression demonstrates the clear advantage of utilizing the coarse level's low-pass characteristics to guide the finer, high-frequency residual reconstructions without overfitting to the noise.

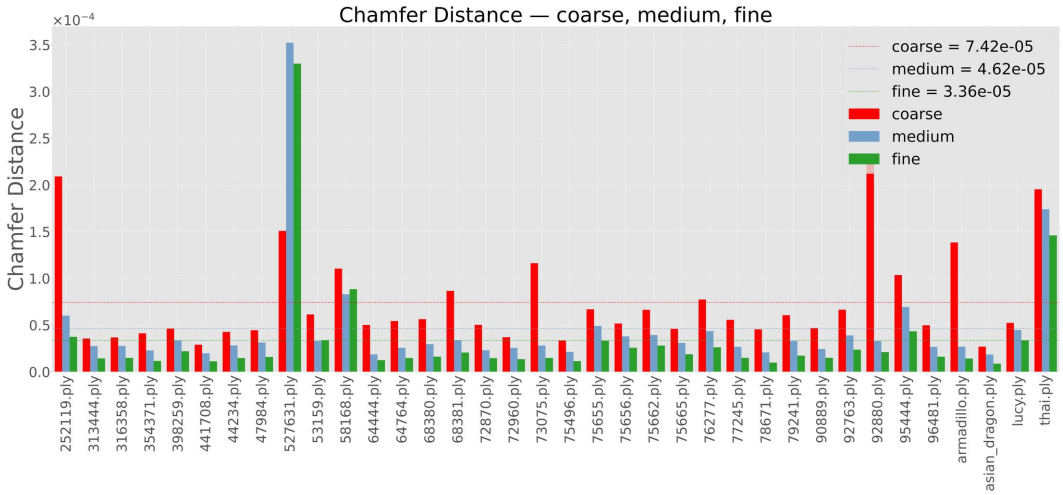


Figure S20: Per-mesh Chamfer Distance for each residual level of our method under 1% noise.

One notable outlier in our evaluation is sample `527631.ply` (see Figure S20), where the Chamfer Distance fails to decrease after the initial coarse level. While the general surface noise is successfully removed, structural artifacts remain. As detailed in Figure S21, the ground truth for this specific sample contains distant, unconnected off-surface point clusters. Consequently, our method accurately smooths the main surface but fits isolated geometry (blobs) to these spatial outliers. The subsequent fine levels preserve these structural blobs, maintaining the higher error metric.

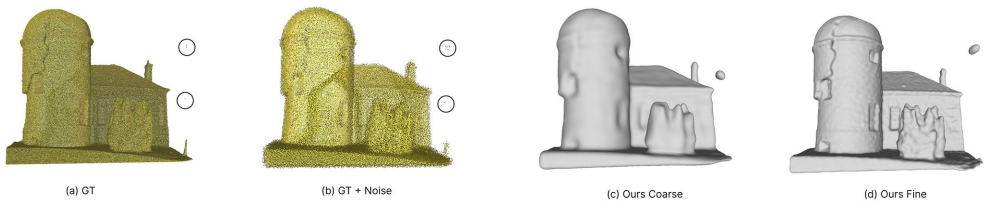


Figure S21: Analysis of outlier sample `527631.ply` from the Thingi32 Dataset. (a) Original ground truth, (b) ground truth with 1% noise perturbation, (c) our coarse reconstruction generating an isolated blob, and (d) our fine reconstruction preserving the structure.

When compared to baseline methods, our approach demonstrates superior robustness to noise. Figure S22 shows that our coarse model achieves a lower Chamfer Distance than the coarse iNGP model across all evaluated meshes. Furthermore, our fine model outperforms the fine iNGP model in all but two meshes (Figure S23).

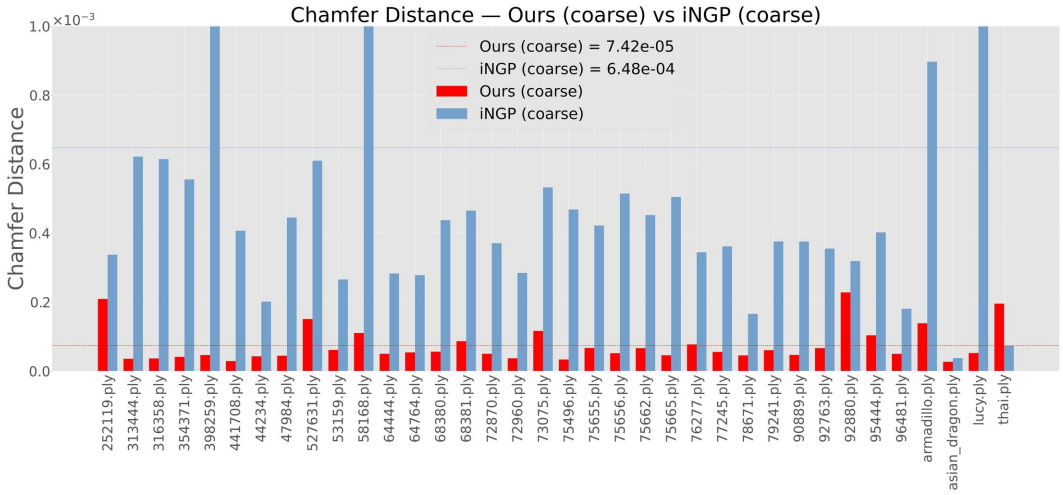


Figure S22: Comparison of our coarse level versus iNGP (coarse) with 1% noise.

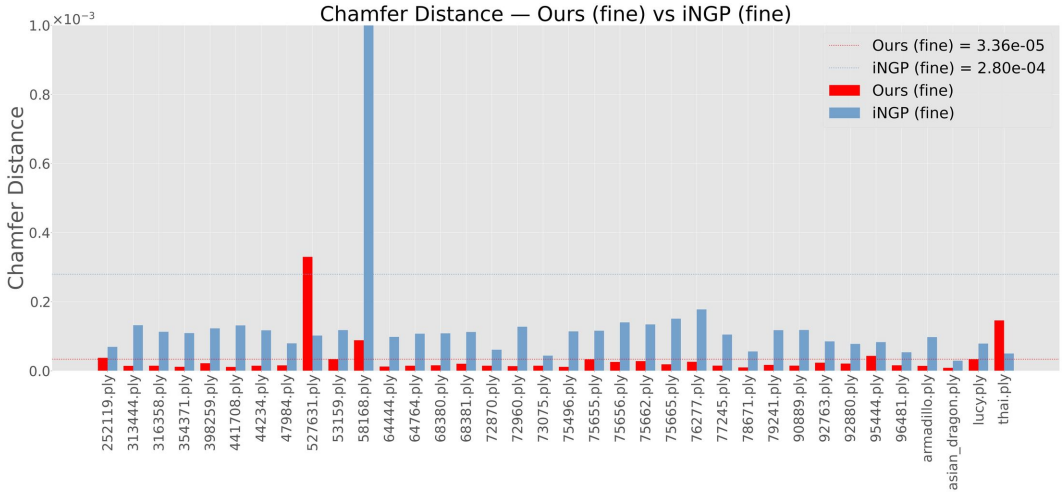


Figure S23: Comparison of our fine level versus iNGP (fine) with 1% noise.

Similarly, our fine level consistently outperforms BACON across the entire dataset (Figure S24). When compared to IDF (Figure S25), a different trend emerges. While IDF achieves lower errors on several individual meshes, it exhibits instability in the presence of noise. This instability results in extreme outliers, drastically inflating IDF's mean Chamfer Distance (8.37e-02 vs. our 3.36e-05). In contrast, our method remains stable and avoids such failures, ultimately providing a more robust and reliable reconstruction.

S2.3 Comparison with Screened Poisson reconstruction

We compare against Screened Poisson surface reconstruction (SPSR) Kazhdan and Hoppe [2013] (Open3D, depth 10, with its standard 1% low-density trim), on the 10-shape subset and evaluation protocol of Sec. S3.1, using the same clean and noisy inputs as Tab. 3 of the main paper. Table S6 reports mean CD and mean IoU.

On clean, dense, well-oriented inputs SPSR is excellent, attaining the best mean IoU of any method we evaluated. That strength is expected of a classical direct solver: handed dense samples with exact normals, SPSR fits the occupancy indicator itself (one global linear solve on a depth-10 adaptive

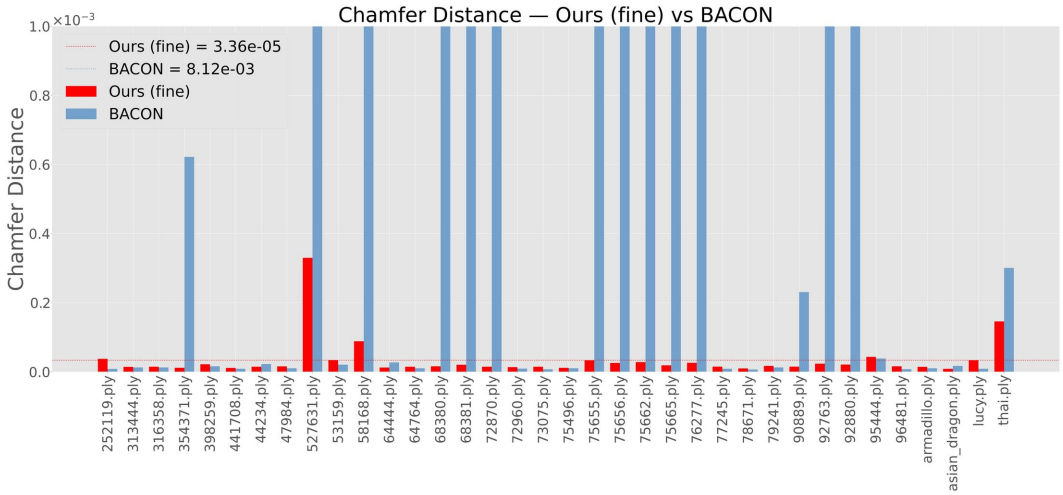


Figure S24: Comparison of our fine level versus BACON with 1% noise.

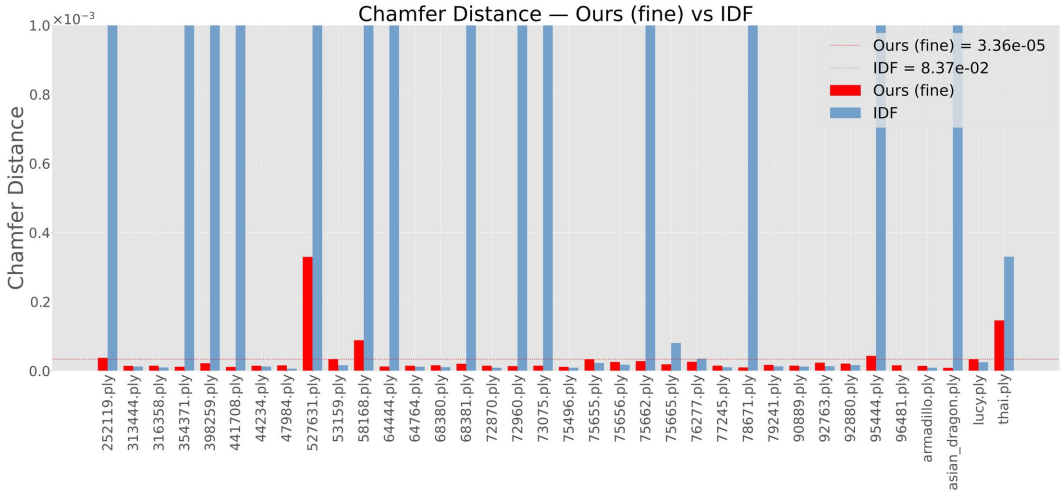


Figure S25: Comparison of our fine level versus IDF with 1% noise, highlighting IDF's vulnerability to extreme outliers.

Table S6: Comparison with Screened Poisson reconstruction (depth 10, 1% density trim). Mean CD and IoU over the 10-shape subset of Sec. S3.1.

Setting	SPSR CD / IoU	M-plicits CD / IoU
clean	7.16E-05 / **0.910**	**3.87E-05** / 0.653
1% noise	9.98E-05 / 0.416	**6.39E-05** / **0.687**

octree, capacity growing with the data rather than fixed) so occupancy agreement is close to its objective. A compact neural SDF approximates under a deliberate band-limited prior instead of interpolating. Under the 1% noise of Tab. 3, SPSR loses more than half its occupancy accuracy (0.910 → 0.416) while M-plicits holds (0.653 → 0.687). Running SPSR *without* its density trim (an outlier-rejection heuristic no other compared method uses) gives mean IoU 0.867 clean and 0.394 under noise. SPSR outputs a triangle mesh, which rasterises in real time once extracted, so we make no rendering-speed claim against it. The differences are in what the representation affords (distance queries away from the surface, the attribute mapping of Sec. S5, and level of detail from a single compact model).

S2.4 Comparison with BANF via a custom re-implementation

BANF's released code targets image fitting, NeRFing, and multi-view 3D reconstruction, without an official implementation covering SDF fitting from point clouds. We therefore re-implemented BANF

from its method section. The numbers below reflect our custom implementation, not the authors' code, and should be read with that caveat.

Configuration. Band-limiting with each level evaluated at its own lattice and reconstructed by linear interpolation, cascaded residual training (level k fits $f - \sum_{l<k} f_l$, recomposed as $\sum_k f_k$), their losses (L2 on ground-truth SDF, Eikonal, Laplacian, with derivatives by finite differences at $\delta = 1/(4r)$), their sampling (500K pool, 40/40/20 on-surface/near-surface/uniform), their schedule ($\{32, 64, 128, 256\}$ lattices; 5K iterations at level 0 with warmup, 10K per finer level; Adam 10^{-3}, batch 100K, sphere initialization), and the iNGP hash encoding (16 levels $\times$ 2 features, table 2^{19}, resolutions 16$\rightarrow$2048, 3$\times$32 decoder).

Fidelity. Our cascade produces the resolutions BANF tabulates, so we can check against their published numbers. In their convention (mean Euclidean Chamfer, $\times 10^{-2}$) we obtain at lattices 32/64/128 on the Asian Dragon 1.57 / 1.20 / 0.92 against their 1.23 / 0.65 / 0.32, and on the Thai Statue 1.17 / 0.92 / 0.74 against 1.41 / 0.68 / 0.40. We match at the coarsest level (0.83–1.27$\times$) but fall behind at finer ones (up to 2.9$\times$), likely an optimisation gap (BANF trains inside iNGP's optimised stack, our re-implementation is pure PyTorch). Absolute comparisons should be read with that gap in mind.

Table S7: Comparison with our BANF re-implementation (CD; paper protocol). "degr." is the clean$\rightarrow$noise degradation factor; best of each pair in bold.

	Dragon	**Armadillo**	**Lucy**	**Thai**
M-plicits clean	**8.65E-06**	7.41E-05	**2.18E-05**	1.94E-04
BANF clean	3.03E-04	**2.79E-05**	1.03E-04	**8.27E-05**
M-plicits noise	**8.69E-06**	**1.38E-05**	**3.33E-05**	**4.90E-04**
BANF noise	3.20E-04	2.03E-04	1.18E-02	1.07E-02
M-plicits degr.	**1.0**$\times$	**0.2**$\times$	**1.5**$\times$	**2.5**$\times$
BANF degr.	1.1$\times$	7.3$\times$	115$\times$	130$\times$

Two within-method observations survive the re-implementation gap. On clean data BANF is the better method on the Thai Statue and the Armadillo. And its noise degradation tracks thin-structure content (negligible on the solid Dragon, moderate on the Armadillo, catastrophic on Lucy and Thai, whose occupancy collapses to IoU 0.001). On Thai, noise takes the mesh from 81 to 474 connected components. The finer levels turn noise into off-surface geometry, which our nested bands prevent structurally, residuals never being supervised or evaluated outside the δ-band (Fig. S10). M-plicits stays within 2.5$\times$ throughout. Independently of this re-implementation, cell (b) of the ablation in Sec. S3.1 isolates the cascaded full-domain supervision pattern BANF shares with IDF and BACON, and iNGP covers the grid-based representation axis.

S3 Additional ablation studies

Loss terms and neighborhood δ. We performed two additional ablation studies for our approach: (i) loss term assessment, (ii) δ influence over the reconstructions. Tables S8,S9, and S10 show the ablation results of our loss function using different weights for each component, while maintaining the remaining hyper-parameters fixed. We performed these studies both for the medium level and the fine level. Note that all studies used the Lucy mesh as a baseline. Table S11 shows the results for varying the delta values while maintaining the remaining hyper-parameters fixed.

S3.1 Isolating the residual composition and the nested bands

We ablate the two design axes separately, against the full method as the reference, on 10 shapes (Armadillo, Asian Dragon, Lucy, Thai Statue from Stanford; 44234, 64764, 68381, 72870, 73075, 77245 from Thingi32), clean and under the 1% vertex noise of Tab. 3, with the paper's evaluation protocol (500K samples, L_2 Chamfer, Hausdorff distance, voxel IoU, 512^3 extraction). All cells share the $(128, 2) \triangleright (256, 2) \triangleright (400, 2)$ architectures, the $\omega_0 = 30/45/100$ schedule, and the same coarse network. On clean inputs cell (a) is the released model. Cell (b) is *residual learning without nested bands* (residuals supervised over the full domain, the IDF/BACON/BANF pattern), cell (c) is *nested bands without residual learning* (independent SDFs, each trained inside the previous level's band). Each cell is extracted with its own sound inference: band-culled hierarchical marching cubes

Table S8: Gradient constraint ablation studies.

(a) Gradient constraint fine level.

Gradient constraint	Approx. Error
0.0	0.0013
10.0	0.0013
30.0	0.0013
100.0	0.0012
300.0	0.0013
1000.0	0.0014
3000.0	0.0017
10000.0	0.0022
30000.0	0.0030

(b) Gradient constraint medium level.

Gradient Constraint	Approx. Error
0.0	0.0086
10.0	0.0084
30.0	0.0082
100.0	0.0078
300.0	0.0074
1000.0	0.0069
3000.0	0.0073
10000.0	0.0087
30000.0	0.0116

Table S9: Normal constraint ablation studies.

(a) Normal constraint fine level.

Normal Constraint	Approx. Error
0.0	0.0017
10.0	0.0013
30.0	0.0013
100.0	0.0013
300.0	0.0013
1000.0	0.0013
3000.0	0.0013
10000.0	0.0013
30000.0	0.0013

(b) Normal constraint medium level.

Normal Constraint	Approx. Error
0.0	0.0073
10.0	0.0074
30.0	0.0077
100.0	0.0081
300.0	0.0083
1000.0	0.0085
3000.0	0.0086
10000.0	0.0087
30000.0	0.0087

for (a) and (c), full-domain extraction for (b), whose residuals are trained domain-wide, so culling it would be unsound.

Comparing (a) and (b) isolates the nested bands. Typical-case Chamfer is a tie, but the tail separates: mean Hausdorff is $3.6\times$ worse without bands (5.23E-03 versus 1.45E-03), and under noise (b) leaves off-surface components on 4 of 10 shapes (reaching 0.135 from the true surface on the Armadillo) against 1 of 10 for the full method, which reconstructs 8 of 10 noisy shapes as a single connected component. This is the quantitative form of Tab. 3's claim: nested bands stop noise from turning into off-surface geometry.

Comparing (a) and (c) isolates the residual sum, which is what noise robustness needs: (c) leads the clean aggregates but degrades $3.0\times$ under noise in mean CD (3.47E-05 $\rightarrow$ 1.04E-04) where the full method stays within $1.7\times$, has higher Hausdorff on 10 of 10 noisy shapes, and fragments even under culled extraction (spurious components on 7 of 10 noisy shapes, up to 373 on 77245). Re-extracting the same noisy checkpoints *without* band culling separates them completely: the residual sum is essentially unchanged (mean CD 6.39E-05 $\rightarrow$ 6.42E-05, 1–4 components per shape), while the independent band-trained SDFs disintegrate into 365 to 4,109 components filling the whole domain (mean CD 2.8E-01). That measures the phenomenon Fig. S10 shows. Without the residual coupling the composed field is only meaningful inside the band, and the design works only if inference culls.

Table S10: SDF constraint ablation studies.

(a) SDF constraint fine level.

SDF Constraint	Approx. Error
0.0	0.0076
10.0	0.0013
30.0	0.0013
100.0	0.0013
300.0	0.0013
1000.0	0.0012
3000.0	0.0013
10000.0	0.0013
30000.0	0.0013

(b) SDF constraint medium level.

SDF Constraint	Approx. Error
0.0	0.0490
10.0	0.0080
30.0	0.0079
100.0	0.0079
300.0	0.0080
1000.0	0.0081
3000.0	0.0080
10000.0	0.0082
30000.0	0.0082

Table S11: Ablation studies of the delta factor. We multiply the delta by the values in the first column and measure the SDF error compared to the Open3D calculated SDF, which we use an ground-truth.

Max delta fraction	Medium level error	Fine level error
1.01	0.0098	0.0048
1.05	0.0098	0.0048
1.10	0.0100	0.0049
1.20	0.0101	0.0049
1.30	0.0103	0.0050
1.50	0.0106	0.0051
2.00	0.0113	0.0053
5.00	0.0139	0.0066

Table S12: Isolation ablation: mean CD / median CD / mean Hausdorff / mean IoU over 10 shapes. Best per column in bold.

Variant	1% noise	clean
(a) w residual, w bands	6.39E-05 / **1.49E-05** / **1.45E-03** / **0.687**	3.87E-05 / **1.38E-05** / 5.86E-03 / 0.653
(b) w residual, w/o bands	**6.34E-05** / 2.26E-05 / 5.23E-03 / 0.555	4.26E-05 / 2.36E-05 / 6.75E-03 / 0.655
(c) w/o residual, w bands	1.04E-04 / 2.36E-05 / 5.72E-03 / 0.669	**3.47E-05** / 1.62E-05 / **4.01E-03** / **0.749**

S3.2 Robustness to coarse-stage under-training

To evaluate sensitivity to the coarse stage's convergence, we trained f_1 for 25%, 50%, and 100% of its epochs and ran the unchanged pipeline on top (Armadillo and Lucy, protocol of Sec. S3.1).

Table S13: Coarse-stage under-training: the adaptive band absorbs an under-trained f_1.

Coarse epochs	δ_1 (Arm.)	final CD (Arm.)	δ_1 (Lucy)	final CD (Lucy)
25%	6.20E-02	1.11E-05	2.83E-02	7.13E-05
50%	2.00E-02	1.06E-04	2.70E-02	5.03E-05
100%	1.71E-02	9.81E-05	2.94E-02	5.24E-05

The band tracks the predecessor's convergence as designed: with a 25%-trained coarse stage, the Armadillo's δ_1 widens automatically by $3.6\times$ and the final quality is unchanged or better (CD 1.11E-05, IoU 0.91), the slack giving the residual more room. Lucy is stable across all three budgets. The stages are therefore not fragile to predecessor state. As for over-training, the coarse level's $\omega_0 = 30$ bounds its spectrum, so it cannot absorb high-frequency noise regardless of epochs because of the low-pass mechanism behind Tab. 3.

S3.3 Outlier contamination

Following the setting of outliers without noise, we add 0.1% and 1% stray points, drawn uniformly inside the unit ball with random normals, to the clean inputs of the Armadillo and Lucy, and evaluate against the clean ground truth (IoU). For reference we run SPSR on the same contaminated clouds, both with and without its low-density trim (an outlier-rejection heuristic).

Table S14: Outlier contamination (IoU against clean ground truth).

Contamination	M-plicits	SPSR (no trim)	SPSR (trim)
Armadillo 0.1%	0.909	0.982	0.978
Armadillo 1%	0.863	0.985	0.981
Lucy 0.1%	0.799	0.995	0.986
Lucy 1%	0.002	0.004	0.938

Sparse contamination is handled (the Armadillo holds IoU 0.86 at 1%), while heavy contamination on Lucy defeats our method and, without its trim, defeats SPSR just as completely (0.004). Its advantage there comes from the rejection heuristic instead of the representation. The mechanism on our side is narrow: Eq. 5 of the main paper sets each band from the *maximum* residual error over the input points, so a single stray point inflates it (on Lucy, δ_1 grows from 2.7E-02 to 1.1 at 0.1% contamination, degenerating the band to the whole domain). A quantile-based δ in place of the max is the natural robustification. This behaviour is consistent with the outlier analysis of Fig. S21. The robustness claims of Tab. 3 concern measurement noise, for which the mechanism is intact, rather than outlier contamination, for which classical pipelines prescribe prefiltering.

S3.4 Measured adaptive band widths

Table S15 reports the adaptive band widths of Eq. 5 ($\varepsilon = 0.3$) measured from the released models on the unit-sphere domain (radius 1). Every band is under 3% of the domain radius and shrinks from level to level as the approximation improves. On noisy inputs the bands widen (Thai: $\delta_1 = 6.9$E-02, $\delta_2 = 5.2$E-02), which is the intended adaptive behaviour. A noisier fit leaves more residual error at the data, so the next level gets more room.

Table S15: Measured band widths (clean inputs, unit-sphere domain).

Shape	$\boldsymbol{\delta_1}$	$\boldsymbol{\delta_2}$
Armadillo	1.68E-02	1.11E-02
Asian Dragon	2.93E-02	1.65E-02
Lucy	2.67E-02	1.43E-02
Thai Statue	2.71E-02	1.57E-02

S4 Rendering and mesh extraction details

To render the zero level set $f^{-1}(0)$ of a SDF f, two common strategies can be used: **sphere tracing** (ST) Hart et al. [1989], which directly traces rays through the SDF field, and **marching cubes** Lorensen and Cline [1987], which extracts an explicit mesh followed by standard rasterization. When the SDF is represented using M-plicits, both rendering approaches become more efficient.

We first introduce a **multiscale sphere tracing** scheme. Given a view ray $\gamma(t) = p_0 + tv$, with origin at point p_0, unit direction v, and intersecting the zero level set $f^{-1}(0)$, the standard sphere tracing (ST) approximates the first intersection point by iterating

$$p_{i+1} = p_i + vf(p_i)$$

along γ. However, querying a high-capacity neural SDF at each step can be computationally expensive. To reduce this cost, we leverage the multiscale SDF hierarchy $\{f_i\}$, using coarser networks to guide the early steps of the tracing process. Thanks to the nesting condition introduced in Eq. 3 of the paper, coarse levels can safely be used to trace offset surfaces before switching to finer levels closer to the surface. The ray initially traces the offset surface $f_1^{-1}(\delta_1)$ using f_1, proceeds to $f_2^{-1}(\delta_2)$ with f_2, and finally goes to the target surface $f_3^{-1}(0)$ using f_3. Each coarser level performs offset tracing via

$$p_{i+1} = p_i + v\left(f_j(p_i) - \delta_j\right),$$

which ensures convergence toward the true surface with minimal reliance on high-cost evaluations. Figure S26 illustrates this procedure, focusing on how the ray reaches S_3 by tracing within the neighborhood $\{|f_2| < \delta_2\}$. For neural SDF inference, we use the GEMM algorithm Dongarra et al. [1990].

Importantly, if $\gamma \cap S_3 \neq \emptyset$, the multiscale ST approximates the first intersection point between γ and S_3. This is guaranteed by the nesting condition, which implies that if $\gamma \cap S_3 \neq \emptyset$, then necessarily $\gamma \cap f_2^{-1}(\delta_2) \neq \emptyset$, and thus also $\gamma \cap f_1^{-1}(\delta_1) \neq \emptyset$. The values δ_i play a critical role in this process, as setting them appropriately helps avoid failures, as illustrated in Fig. 4 of the paper. Equation 5 (in the main paper) provides a principled definition of δ_i, linking them to network training.

Finally, M-plicits also accelerates mesh extraction via **marching cubes**. We propose an adaptive grid inference strategy: we first evaluate the coarse SDF f_1 to cull grid vertices, and only evaluate finer SDFs for vertices inside the δ_1-neighborhood. This yields efficient and focused SDF sampling, as depicted in Fig. 5 of the paper. Fig. S34 shows reconstructions using this approach.

S5 Normal and texture mapping details

Let S be a surface nested within a δ-neighborhood of the zero level set of a neural SDF f, i.e., $S \subset \big[|f| \leq \delta\big]$. Assume f is a finer neural SDF. Then, the *neural normal mapping* assigns to each point $p \in S$ the attribute

$$g(p) := \nabla f(p).$$

This corresponds to restricting ∇f to S, effectively transferring the normal of $f^{-1}(0)$ along the shortest path connecting it to p. Since f is a signed distance function, the gradient ∇f remains constant along such paths.

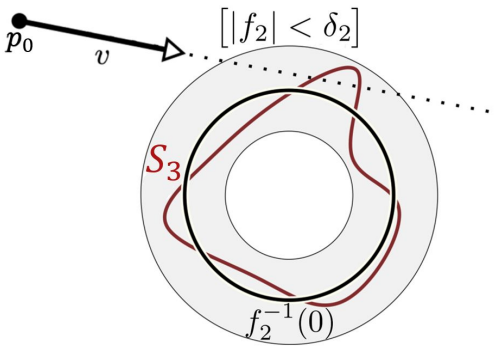


Figure S26: Ray intersecting S_3 nested in a δ-neighborhood of a coarse SDF f_2. Notice that sphere tracing f_2 directly would lead to a false negative, thus we use $[|f_2| < \delta_2]$ instead.

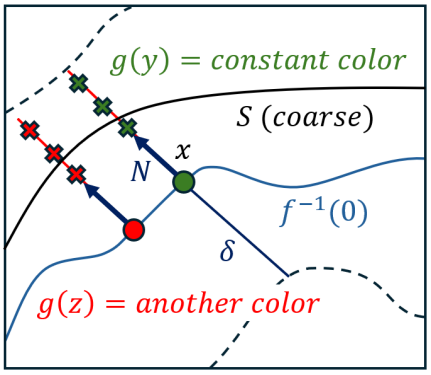


Figure S27: Volumetric texture mapping. The texture g should be constant along the normals N near the coarse surface S (red/green). Having such volumetric representation in the δ-neighborhood ensures that g can be assigned to any point in the coarse surface S.

We consider two cases. First, let S be a triangle mesh. We use neural normal mapping to transfer detailed normals from the level sets of f onto S. This approach is analogous to classical normal mapping, which typically relies on UV parameterizations. However, since our method is volumetric, such parameterizations are not required (see Fig. 7 in the paper, middle).

In the second case, let S be the zero level set of a coarser neural SDF. Here, neural normal mapping allows us to bypass additional sphere tracing iterations (see Fig. 7 in the paper, left). In this case, surface extraction via marching cubes is not necessary.

Similarly, we define a neural network $g : \mathbb{R}^3 \to \mathcal{C}$ to encode a *texture* over the δ-neighborhood of f, where the codomain $\mathcal{C}$ is the RGB color space. The attribute mapping associated with the triple $\{S, f, g\}$ is referred to as *neural texture mapping*.

To train the parameters ϕ of g, we use the following loss functional:

$$\mathcal{T}(\phi) = \int_{f^{-1}(0)} (g - \cancel{g})^2 \, dx + \int_{[|f| \le \delta]} \langle \nabla g, \nabla f \rangle^2 \, dx,$$

where the first term encourages g to match the *ground-truth* texture $\cancel{g}$, and the second term enforces consistency of g along the gradient paths of f, regularizing the network within the δ-neighborhood. Figure S27 illustrates the texture mapping scheme.

S6 GEMM-based Analytical Normal Calculation for MLPs details

We propose a GEMM-based analytical computation of normals, which are continuous and do not need auto-differentiation, resulting in smooth normals. To compute them, we recall that a MLP with $n-1$ hidden layers has the following form $f(x) = W_n \circ h_{n-1} \circ \cdots \circ h_0(x) + b_n$, where $h_i(x_i) = \varphi(W_i x_i + b_i)$ is the i-layer. The *activation* φ is applied on each coordinate of the linear map $W_i : \mathbb{R}^{N_i} \to \mathbb{R}^{N_{i+1}}$ translated by $b_i \in \mathbb{R}^{N_{i+1}}$. The gradient of f is given using the *chain rule*:

$$\nabla f(x) = W_n \cdot \mathbf{J}h_{n-1}(x_{n-1}) \cdot \cdots \cdot \mathbf{J}h_0(x), \quad \text{with} \quad \mathbf{J}h_i(x_i) = W_i \odot \varphi'\big[a_i| \cdots |a_i\big] \tag{S6}$$

$\mathbf{J}$ is the *Jacobian*, $x_i := h_{i-1} \circ \cdots \circ h_0(x)$, $\odot$ is the *Hadamard* product, and $a_i = W_i(x_i) + b_i$. Eq. S6 is used in Gropp et al. [2020], Novello et al. [2022] to compute the level set normals analytically.

We now use Eq. S6 to derive a GEMM-based algorithm for computing the normals (∇f) in real-time. The gradient ∇f is given by a sequence of matrix multiplications which is not appropriate for a GEMM setting because $\mathbf{J}h_0(x) \in \mathbb{R}^{3 \times N_1}$. The GEMM algorithm organizes the input points into a matrix, where its lines correspond to the points and its columns organize them and enable parallelism. We can solve this problem using three GEMMs, one for each normal coordinate. Therefore, each GEMM starts with a column of $\mathbf{J}h_0(x)$, eliminating one of the dimensions. The resulting multiplications can be asynchronous since they are completely independent.

The j-coord of ∇f is given by $G_n = W_n \cdot G_{n-1}$, where G_{n-1} is given by iterating $G_i = \mathbf{J}h_i(x_i) \cdot G_{i-1}$, with the initial condition $G_0 = W_0[j] \odot \varphi'(a_0)$. The vector $W_0[j]$ denotes the j-column of W_0. We

use a kernel and a GEMM to compute G_0 and G_n. For G_i with $0 < i < n$, observe that

$$G_i = (W_i \odot \varphi'\,[a_i| \cdots |a_i]) \cdot G_{i-1} = (W_i \cdot G_{i-1}) \odot \varphi'(a_i).$$

The first equality comes from Eq. S6 and the second from a commutative property of the Hadamard product. The second expression needs fewer computations and is solved using a GEMM followed by a kernel.

Algorithm 1 presents the gradient computation for a batch of points. The input is a matrix $P \in \mathbb{R}^{3\times k}$ with columns storing the k points generated by the GEMM version of the sphere tracing algorithm. The output is a matrix $\nabla f_\theta(P) \in \mathbb{R}^{3\times k}$, where its j-column is the gradient of f_θ evaluated at $P[j]$. Lines $2-5$ are responsible for computing G_0, Lines $6-11$ compute G_{n-1}, and Line 13 provides the result gradient G_n. Table S16 shows a comparison between this algorithm and automatic differentiation using PyTorch.

Table S16: Runtime comparison, in seconds, between Pytorch autograd and our algorithm to calculate the normals. Ours performs 2× faster. Samples evaluated are from Stanford dataset.

Model	Autograd	Ours	Resolution
Armadillo 256x2	0.007	**0.003**	512x512
Armadillo 256x2	0.024	**0.010**	1024x1024
Armadillo 256x3	0.010	**0.005**	512x512
Armadillo 256x3	0.025	**0.012**	1024x1024
Buddha 256x2	0.008	**0.005**	512x512
Buddha 256x2	0.021	**0.014**	1024x1024
Buddha 256x3	0.011	**0.005**	512x512
Buddha 256x3	0.024	**0.012**	1024x1024
Lucy 256x2	0.007	**0.004**	512x512
Lucy 256x2	0.021	**0.012**	1024x1024
Lucy 256x3	0.011	**0.007**	512x512
Lucy 256x3	0.025	**0.015**	1024x1024

Input: neural SDF f_θ, positions P
Output: Gradients $\nabla f_\theta(P)$
for $j = 0$ *to* 2 *(async)* **do**
  using a GEMM: `// Input Layer`
    $A_0 = W_0 \cdot P + b_0$
  using a kernel:
    $G_0 = W_0[j] \odot \varphi'(A_0);\ P_0 = \varphi(A_0)$
  `// Hidden layers`
  for *layer* $i = 1$ *to* $n-1$ **do**
    using GEMMs:
      $A_i = W_i \cdot P_{i-1} + b_i;$
      $G_i = W_i \cdot G_{i-1}$
    using a kernel:
      $G_i = G_i \odot \varphi'(A_i);\ P_i = \varphi(A_i)$
  end
  using a GEMM: `// Output layer`
    $G_n = W_n \cdot G_{n-1}$
end

Algorithm 1: Normal computation

S7 Additional applications

S7.1 Surface reconstruction from posed images

We demonstrate that M-plicits can be seamlessly integrated into image-based reconstruction pipelines, such as NeuS Wang et al. [2021]. To this end, we replace the standard neural SDF used in NeuS with our multiscale SDF architecture, composed of two neural networks: a coarse-level network and a higher-resolution refinement network. We compare our modified pipeline against the baseline NeuS to evaluate the impact of our approach both quantitatively and qualitatively. Specifically, we implement the coarse network with 4 hidden layers of 128 neurons each, and the fine-level network with 5 hidden layers of 256 neurons. Despite having 37% fewer parameters than the original NeuS architecture, our multiscale approach achieves improved performance. Quantitatively, under the default volume rendering configuration, our method yields an average PSNR improvement of 3.74% across models from the DTU dataset Jensen et al. [2014]. Table S17 summarizes the PSNR comparisons between our approach and the baseline NeuS.

Table S17: PSNR comparison between the baseline NeuS and our multiscale method on selected scans from the DTU dataset Jensen et al. [2014]. Our method consistently improves reconstruction quality while using fewer network parameters. All values are reported in dB.

	Scan 24	Scan 37	Scan 40	Scan 55	Scan 63	Avg.
NeuS	28.20	27.10	28.13	28.80	32.05	28.86
Ours	31.50	26.50	27.78	29.01	34.89	29.94

Figure S28 presents a comparison between NeuS and our multiscale variant on Scan 24 of the DTU dataset. The top row shows the reconstructed mesh geometry, including zoomed-in insets

that highlight fine surface details. The bottom row displays renderings showing that ours provide reconstruction with sharp details. Our method yields improved geometric fidelity and cleaner surface reconstructions, especially in regions with architectural features.

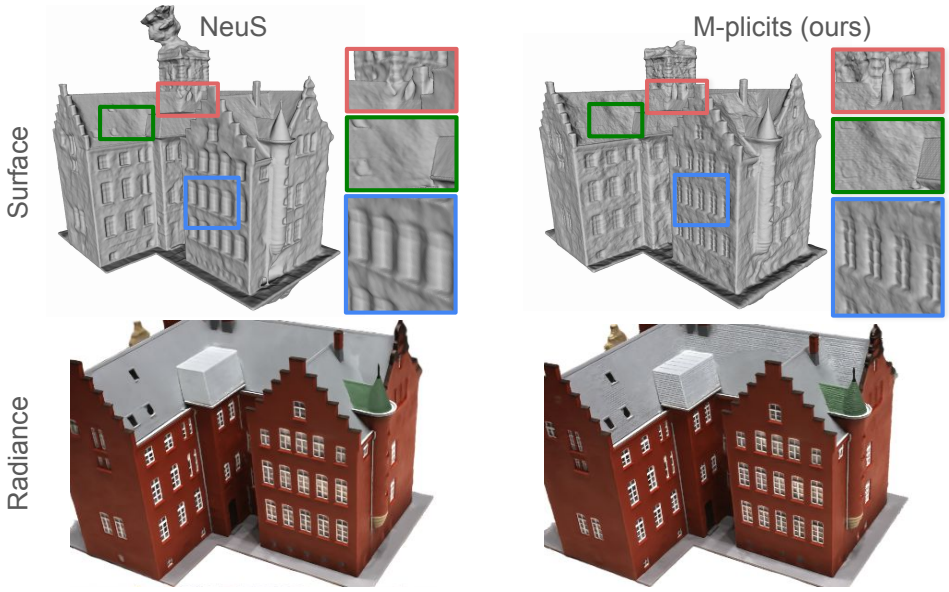


Figure S28: Comparison between baseline NeuS and our multiscale variant on Scan 24 of the DTU dataset. Top: extracted surface meshes with zoomed-in details. Bottom: rendered appearance. Our approach recovers finer geometric details, as evident in architectural structures and window boundaries.

S7.2 Neural implicit surface evolution

Note that neural SDFs provide a smooth representation of a static scene. By adding an additional input coordinate, we can encode time into the representation. We leverage this approach to train dynamic evolutions of static neural SDFs, following the training schemes introduced in Novello et al. [2023]. Fig. S29 presents an example of interpolation between the Spot and Bob models using this method. Importantly, the implicit model handles topology changes, demonstrating that our representation can be integrated into differentiable pipelines. The visualization is in real-time (120 FPS) using an extension of our multiscale ST to dynamic SDFs.

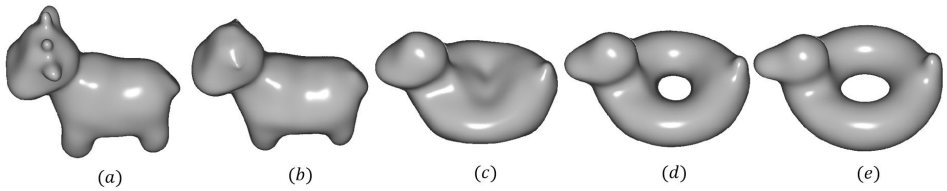


Figure S29: A dynamic multiscale SDF is trained using the pipeline from Novello et al. [2023]. Note the change in topology (c-d), which is challenging to handle using meshes. Also, octree/mesh-based approaches require generating a surface for each time, an overhead that our model avoids.

S7.3 Additional experiments

Normals. We compare our GEMM normal calculation against `torch.autograd`. As shown in Tab. S16, ours performs $2\times$ faster. We tested 6 different INRs trained for Stanford dataset samples: Armadillo, Buddha, and Lucy, varying between 2-3 hidden layers.

Point cloud from images: Fig S30 shows our model trained with a point cloud reconstructed from an image. We use Depth Anything Yang et al. [2024] to generate the depth of the pixels and use that depth to create the point cloud based on the view.

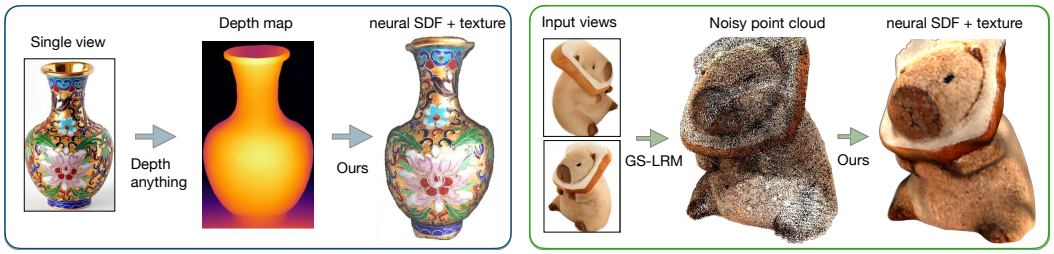


Figure S30: Training a textured SDF from images/noisy point cloud. On the left, our model (neural SDF + texture) is trained using the unprojection of a depth map, which is computed from a single view using Depth Anything. The resulting vase is rendered at 32.1 FPS. On the right, we show a reconstruction derived from a noisy point cloud, extracted from multiple views using GS-LRM Zhang et al. [2024]. By combining our method with this feed-forward 3D model (GS-LRM), we achieve fast reconstruction of the SDF with texture.

Scene-scale point cloud. We also evaluated our method on a scene-scale point cloud containing more than 10 million points. Figure S31 shows that M-plicits successfully reconstructs the entire scene across all scales (coarse, medium, and fine), whereas Instant-NGP fails in our tests using both 3 levels (coarse) and 16 levels (fine). The Chamfer distance further corroborates these observations: M-plicits achieves 2–3 orders of magnitude lower Chamfer distance compared to Instant-NGP.

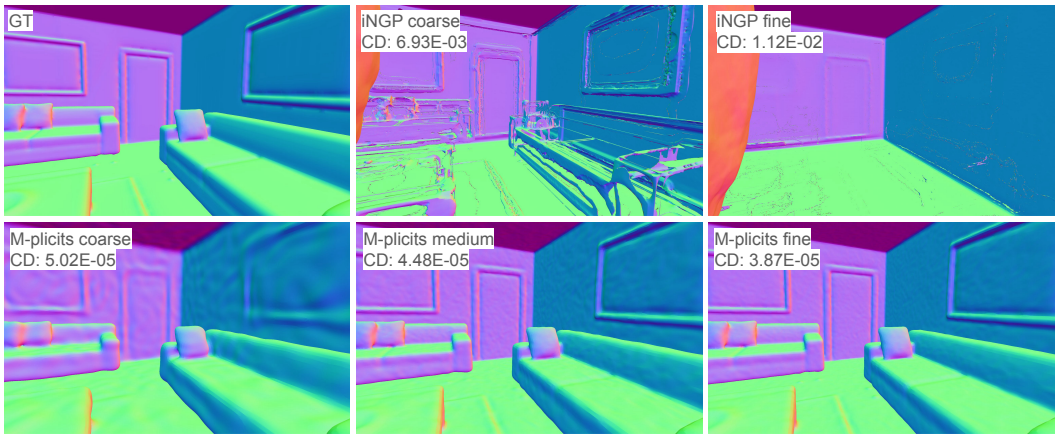


Figure S31: **Scene-scale reconstruction (10M+ points).** Comparison between Instant-NGP (coarse and fine) and M-plicits at three scales (coarse, medium, fine). Instant-NGP fails to reconstruct large regions of the geometry and introduces strong artifacts, even when increasing the number of levels. In contrast, M-plicits yields accurate and stable reconstructions at all scales, closely matching the ground-truth surface, as reflected in the reported Chamfer distances.

Broader perceptual evaluation: Fig. S32 shows a broader perceptual evaluation of the multiscale sphere tracing and the neural normal mapping using several models. Fig. S33 also shows the images we use to calculate the MSE to compare the neural texture mapping with the rendering baseline.

Accelerated Marching Cubes qualitative evaluation: Fig. S34 shows high-fidelity reconstructions computed using our acceleration for the marching cubes algorithm.

Residuals. To evaluate our nested neighborhood scheme and loss design, we compare against a **baseline** residual variant $f_2 = f_1 + r_1$, where both f_1 and r_1 are trained using SIREN's original loss and sampling strategy, following the baseline in IDF Wang et al. [2022a]. In contrast, our method employs the loss in Eq. 4 in the main paper together with *nested neighborhoods* to supervise the ground-truth SDF within a narrow band around the previous stage's zero level set. This exploits the property $f(\mathbf{x}_j + t\,\mathbf{N}_j) = t$ within a tubular neighborhood, enabling supervision beyond the original samples $\mathbf{x}_j$ to improve stability and prevent error growth. This strategy also reduces network complexity, allowing us to use much smaller architectures: a base network with a single hidden layer of 128 neurons ($\omega_0 = 30$) and r_1 with a single hidden layer of 256 neurons ($\omega_0 = 45$).

We compare this baseline with M-plicits on the Thingi32 dataset, where the baseline achieves an average CD of 6.2e-2, while our method reaches 1.2e-2, demonstrating that M-plicits yields substantially better reconstructions. We also evaluate the residual approach qualitatively. Fig. 13 in

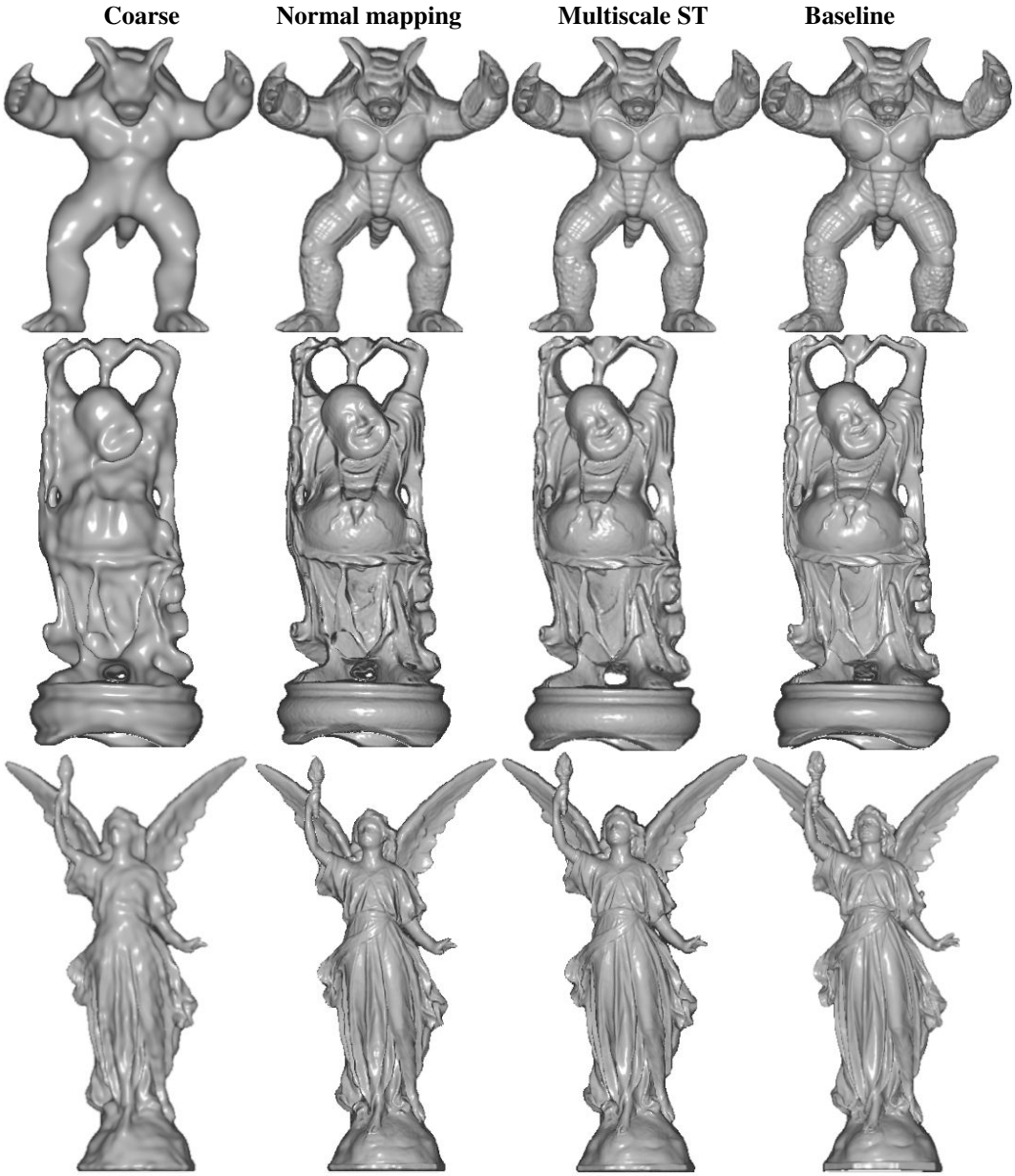


Figure S32: Comparison between our method and the SIREN baseline. The columns represent different configurations. From left to right: $(\mathbf{64}, \mathbf{2})$, $(\mathbf{64}, \mathbf{2}) \triangleright (\mathbf{256}, \mathbf{2})$, and the baseline $(\mathbf{256}, \mathbf{4})$. The second column uses neural normal mapping and the third uses multiscale sphere tracing. Notice that fidelity is improved in the second column and the third column refines the results.



Figure S33: Images we use to calculate the MSE between the ground-truth textured meshes and our approach.



Figure S34: From left to right: Marching cubes reconstruction of Armadillo, Buddha and Lucy using our proposed grid culling method.

the main paper shows that residuals decrease spurious components when combined with neighborhood training. We further exploit this property to accelerate marching cubes in scenarios where mesh extraction is required.

Textures. We define textures directly in a neighborhood of the surface, removing the need for a UV map. This formulation produces visually convincing appearance while decoupling texture from geometry in a compositional way. To assess accuracy, we compared our approach against traditional UV-textured meshes by measuring the MSE between rendered images. Across five test models—Spot, Bob, Bunny, Egg, and Earth—we obtained MSEs of 0.0329, 0.0434, 0.0720, 0.0291, and 0.0033, respectively. Figure S35 illustrates neural texture mapping applied to coarse surfaces.



Figure S35: Neural texture mapping. All networks are $(256, 3)$, except for the the earth, which is $(512, 3)$. The surfaces are marching cubes of $(64, 1)$ SDFs, except for the bunny, which is $(128, 2)$.

Scene scale test. We also evaluated our method on a scene-scale point cloud (Figure S31). M-plicits successfully reconstructs the entire scene across all scales, whereas Instant-NGP fails in our tests. The Chamfer distance further corroborates these observations: M-plicits achieves 2–3 orders of magnitude lower Chamfer distance.

Normals. We compare our GEMM normal calculation against `torch.autograd`. Our method performs $2\times$ faster across 6 different INRs trained for the Armadillo, Buddha and Lucy, with architectures varying between 2 and 3 hidden layers. More details are provided in Tab. S16.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 1 describe M-plicits as a residual sum of MLPs trained on nested neighborhoods, with multiscale sphere tracing and GEMM-based analytical normals. The empirical claims (best mean CD in coarse, best median CD/IoU in fine, strongest noise robustness, real-time rendering, an order of magnitude fewer parameters than grid-based baselines) are verifiable in Sec. 4 (Tabs. 2–5).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 5 contains an explicit "Limitations" statement (function smoothness preventing very sharp edges, scope of the noise regime). The Trade-offs paragraph in Sec. 4.1 also openly acknowledges that no single method dominates all axes simultaneously and characterizes the design region M-plicits occupies.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results requiring proofs. Contributions are methodological (architecture and training scheme) and empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 3 fully describes the architecture (residual MLP sum, nested-neighborhood supervision, loss in Eq. 4, adaptive band threshold in Eq. 5) and Sec. 4 lists implementation details (Adam optimizer, NVIDIA RTX 5090 with 32 GB, MLP-size notation, ω_0 values, sphere-tracing iteration counts). Datasets (Stanford 3D Scanning Repository, Thingi32) are publicly available, and the evaluation protocol (marching cubes at 512^3, 500K samples, PyTorch3D) is fully specified. Loss-term and δ-sensitivity ablations are in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code, trained models, experiment data, and reproduction instructions are available at `https://github.com/dsilvavinicius/m-plicits`, as linked in the abstract. The repository documents the environments, commands, and verification results for training, reconstruction, evaluation, and rendering. The datasets we use (Stanford 3D Scanning Repository, Thingi32) are publicly available with their own licenses; baseline implementations (Instant-NGP, BACON, IDF) are taken from the authors' public repositories.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 4 "Implementation details" specifies the Adam optimizer, hardware (RTX 5090, 32 GB), MLP-size notation (N, d), ω_0 values for the SIRENs (30/45/100 across coarse/medium/fine), and sphere-tracing iteration counts (20 for the first SDF, 5 per residual). The evaluation protocol (point cloud normalization, marching cubes at 512^3, 500K sampled points, PyTorch3D for CD/IoU) is also specified. Loss-term weight ablations are in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The main paper reports means and medians for CD/IoU (Tabs. 2–3) without explicit error bars. Per-mesh CD comparisons in supplementary Figs. S14, S15, S17, and S18 illustrate variation across shapes; they do not quantify variability across repeated training runs. Multi-seed runs were not performed due to compute constraints.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. 4 “Implementation details” specifies that all experiments are conducted on a single NVIDIA RTX 5090 with 32 GB. Per-model average training times (in minutes) are reported in Tab. 2, and rendering throughput (FPS) is reported in Tabs. 2 and 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and confirm that the research conforms. The work uses only publicly available 3D shape datasets, involves no human subjects, and raises no privacy or fairness concerns.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This is foundational research on neural surface representations for 3D geometry. The applications described in Sec. 1 (augmented/virtual reality, digital twins, cultural heritage preservation, autonomous robotics) are positive use cases; we are not aware of a direct path to negative societal impact from the technique itself.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release pre-trained generative models, language models, or scraped datasets. The released code targets surface reconstruction from oriented point clouds and poses no high risk for misuse.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the Stanford 3D Scanning Repository and a subset of Thingi10K (Thingi32), both publicly available and cited (Curless and Levoy, 1996; Zhou et al., 2016; Takikawa et al., 2021). Baseline implementations (Instant-NGP, BACON, IDF) are used per their public licenses and credited in Sec. 2 and Sec. 4. PyTorch3D, CUTLASS, and PyTorch are used per their open-source licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces M-plicits as a new method. Code, trained models, training scripts, and the evaluation pipeline are publicly available with documentation at `https://github.com/dsilvavinicius/m-plicits`. Details of the architecture, training procedure, and evaluation protocol are provided in Sec. 3, Sec. 4, and the supplementary material to enable replication.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subjects research; only public 3D shape datasets are used.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects are involved in the research; IRB approval is therefore not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology—residual MLPs with nested-band training, multiscale sphere tracing, and GEMM-based analytical normals—does not involve LLMs as important, original, or non-standard components. We use Depth Anything in one supplementary demonstration (Fig. S30), but it is a depth estimator rather than an LLM; the contribution of the paper does not depend on it.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.