High-Fidelity Lightweight Mesh Reconstruction from Point Clouds

Chen Zhang Wentao Wang Ximeng Li Xinyao Liao Wanjuan Su Wenbing Tao*
National Key Laboratory of Science and Technology on Multi-spectral Information Processing,
Huazhong University of Science and Technology, China
{zhangchen_, wentaowang, ximengli, xinyao_liao, suwanjuan, wenbingtao}@hust.edu.cn

Abstract

Recently, learning signed distance functions (SDFs) from point clouds has become popular for reconstruction. To ensure accuracy, most methods require using high-resolution Marching Cubes for surface extraction. However, this results in redundant mesh elements, making the mesh inconvenient to use. To solve the problem, we propose an adaptive meshing method to extract resolution-adaptive meshes based on surface curvature, enabling the recovery of high-fidelity lightweight meshes. Specifically, we first use point-based representation to perceive implicit surfaces and calculate surface curvature. A vertex generator is designed to produce curvature-adaptive vertices with any specified number on the implicit surface, preserving the overall structure and high-curvature features. Then we develop a Delaunay meshing algorithm to generate meshes from vertices, ensuring geometric fidelity and correct topology. In addition, to obtain accurate SDFs for adaptive meshing and achieve better lightweight reconstruction, we design a hybrid representation combining feature grid and feature triplane for better detail capture. Experiments demonstrate that our method can generate high-quality lightweight meshes from point clouds. Compared with methods from various categories, our approach achieves superior results, especially in capturing more details with fewer elements.

1. Introduction

With the development of neural representations, learning signed distance fields (SDFs) from point clouds has become popular for solving the surface reconstruction problem [4, 9, 10, 20, 24, 24, 31, 34, 36]. To maximize reconstruction accuracy, these methods typically require high-resolution Marching Cubes (MC) [21] to generate densely packed meshes with excessive vertices and faces. This leads to excessive memory requirements, limiting the practical applicability of the mesh for downstream tasks like rendering and editing. Although lightweight meshes can be extracted using MC with low-resolution grids, it often struggles to recover sharp features, resulting in significant loss of detail (See Figure 1). Recovering high-fidelity lightweight meshes is an important and valuable challenge in surface reconstruction from point cloud.

*Corresponding author.

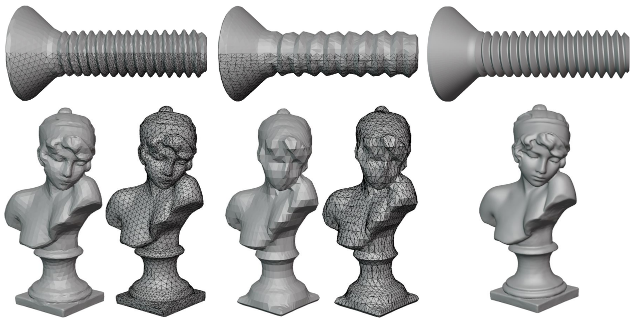

Figure 1. The comparison of our adaptive meshing method with MC using the same element count. The input SDFs for both methods are the same. Our method achieves a curvature-adaptive distribution of vertices and generates more detailed meshes.

More recently, some learning-based methods [6, 7, 26] are proposed to improve traditional meshing algorithms and construct more faithful meshes with low-resolution grids. Instead of edge interpolation in MC, they apply 3D convolutions on the grids to predict vertices. Then, they modify the meshing templates of traditional algorithms like MC and Dual Contouring (DC) [15], or construct a Voronoi diagram to generate the mesh. As alternatives to MC, these neural methods improve detail preservation and can extract more accurate low-resolution meshes. However, they still face problems in recovering high-quality lightweight meshes:
1) Explicit geometry perception of the implicit surface is lacking. Predicting vertices within a voxel through convolution cannot guarantee accurate distribution on the zero-value surface of SDFs, with error depending on the voxel size. When using a low-resolution grid with a large voxel size, these methods represent dense, continuous SDFs as sparse, discrete structures, leading to a significant loss of

original geometric details. In such cases, capturing variations in surface curvature becomes challenging.

2) The local resolution of mesh cannot be adjusted adaptively. Regular grids constrain these methods to producing meshes with an approximately uniform vertex distribution, resulting in globally uniform resolution. However, an ideal lightweight mesh should have variable resolution: higher in detailed areas and lower in flat regions. Adaptive resolution allows for more accurate representation of geometry with fewer elements. Yet, current meshing methods ignore this key property for lightweight reconstruction. A resolution-adaptive meshing algorithm is essential for extracting high-quality lightweight meshes from SDFs.

3) The topological properties of mesh are difficult to guarantee. A high-quality mesh must not only preserve geometric fidelity but also ensure correct topology, including manifoldness, watertightness, and non-self-intersection. This is important for the mesh's broad applicability in downstream tasks. However, current learning-based methods often focus on local vertex connectivity and lack effective constraints on global combinations, leading to low-quality topologies.

In this paper, we present a new adaptive meshing method to achieve high-fidelity ***L**ightweight **M**esh **R**econstruction (**LMR**)*. It generates resolution-adaptive meshes based on surface curvature, exhibiting three important properties: ***1) Precise surface perception.*** We use the gradient of SDFs to project spatial queries onto the implicit surface, capturing the surface geometry and curvature information. This point-based representation makes the implicit surface explicit and provides clear constraints for vertex generation. ***2) Adaptive mesh resolution.*** We design a vertex generator to initialize vertices from surface queries and learn positional refinements for them. The vertices are adaptively distributed on the implicit surface based on surface curvature, dense in high-curvature areas and sparse in low-curvature areas. ***3) Correct mesh topology.*** A Delaunay meshing algorithm is proposed to infer global vertex connectivity from the Delaunay triangulation. It ensures that the resulting mesh closely approximates the implicit surface while maintaining correct topological properties, such as watertightness, manifoldness, and non-self-intersection.

Moreover, to obtain accurate SDFs for adaptive meshing and achieve better lightweight reconstruction from point clouds, we explicitly define a voxel grid and a tri-plane within the SDF network to store learnable features related to spatial locations. Compared with most neural implicit reconstruction methods that rely solely simple MLPs, this approach enhances the representation capability of SDF network, enabling more detailed SDFs.

Extensive experiments demonstrate the superiority of our method in recovering high-quality lightweight meshes with complex details from point clouds. In summary, our contributions are as follows:

- Point-based representation is used to capture the geometry information of implicit surface, and a vertex generator is proposed to achieve curvature-adaptive distribution for any specified number of vertices.
- Robust tetrahedral classification is achieved for meshing from Delaunay triangulation, producing accurate meshes that approximate the implicit surface while ensuring correct topology.
- Multi-geometry hybrid features from explicit grid and tri-plane enable the capture of fine-grained details, resulting in accurate SDFs.
- A new paradigm applies explicit adaptive meshing to implicit neural representation, tackling the challenge of reconstructing high-fidelity lightweight meshes from point clouds.

2. Related Work

Neural Meshing. Marching Cubes (MC) [21] is a widely used method for extracting meshes from SDFs. To improve the meshing performance of MC at low resolutions, several learning-based methods are proposed. They apply 3D convolutions on the grids to predict vertices in meshes, instead of using edge interpolation. NMC [6] and NDC [7] improve the meshing templates of MC and Dual Contouring (DC) [15] by incorporating the predicted vertices. VoroMesh [26] optimizes vertex prediction and generates meshes using the Voronoi diagram, while PoNQ [27] employs quadric error metric. Current neural meshing methods still rely on the voxel-based approach, limiting their meshing capabilities at low-resolution grids. In addition, ensuring correct mesh topology is also a challenge for them.

Neural Implicit Reconstruction. Learning SDFs by overfitting a network on an unseen point cloud has become a popular approach for surface reconstruction [5, 13, 14, 19, 23, 41, 44]. Most methods focus on designing various constraint losses to guide the SDF optimization. However, the widely used MLP structure limits their ability to capture fine details. To address this, GridPull [4] constructs a voxel grid to explicitly store SDF values, but it is prone to artifacts caused by SDF discretization. Some methods from multi-view neural rendering and reconstruction [16, 18, 29, 40, 43, 47, 50] demonstrate that different explicit geometric representations have varying effects on learning 3D geometry. Inspired by this, we introduce hybrid learnable features to enhance detail learning while maintaining the continuity of SDFs.

Explicit Reconstruction. Point set triangulation [3, 22, 38, 42, 48, 49] generates explicit meshes by inferring local triangle connectivity from point clouds. By combining point cloud downsampling [17, 28, 28, 30, 35], lightweight mesh reconstruction can be achieved. However, this process often struggles to ensure the smoothness and watertightness of meshes. Additionally, current learning-based methods of-

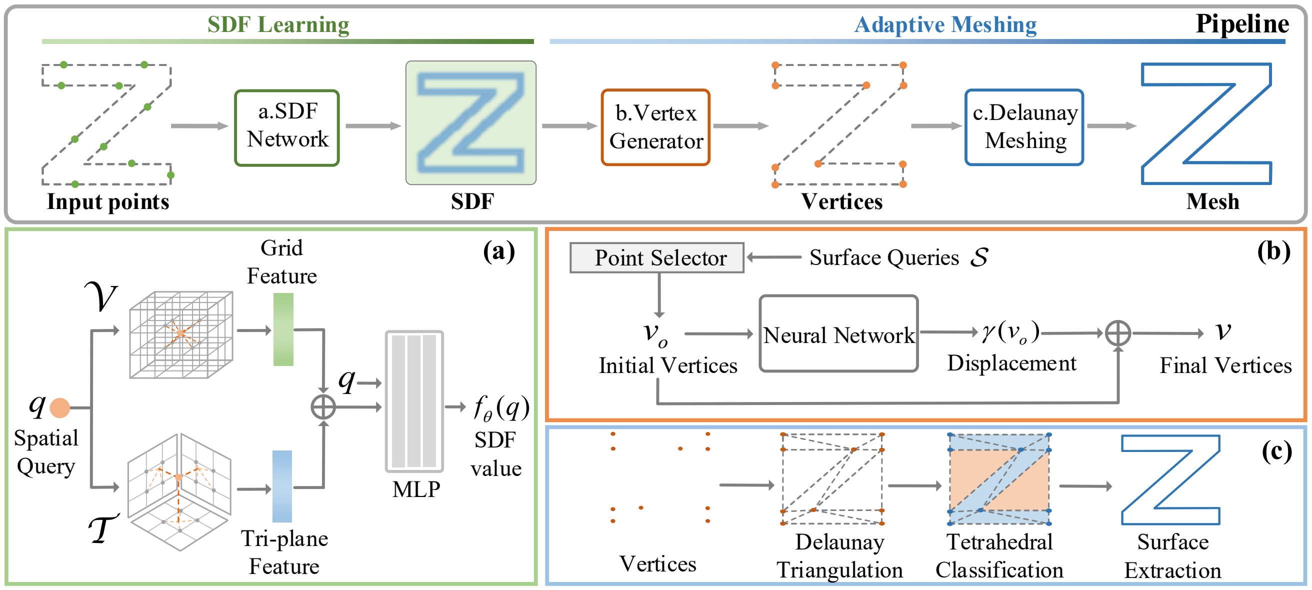

Figure 2. The pipeline of our ***L**ightweight **M**esh **R**econstruction (LMR)*. It consists of two stages: SDF learning and adaptive meshing. In SDF learning, we use hybrid features to enhance the SDF representation. In adaptive meshing, a vertex generator creates curvature-adaptive vertices on the implicit surface, followed by a Delaunay meshing algorithm to produce high-quality meshes. (a) SDF network architecture. (b) Vertex generator architecture. (c) 2D example of the Delaunay meshing algorithm.

ten impose strict restrictions on the number of input/output points, limiting their practical applicability. Nonetheless, this explicit reconstruction process serves as inspiration for our adaptive meshing on the implicit surface.

3. Method

Given a point cloud $\mathcal{P}$ (which can be unoriented), our aim is to recover accurate geometry with a lightweight mesh. We first fit an accurate SDF from the point cloud and then use an adaptive meshing algorithm to extract a resolution-adaptive mesh from the implicit surface. With a limited number of vertex elements, the mesh resolution is adaptively adjusted according to surface curvature, achieving lightweight, high-fidelity, and topologically correct properties. The pipeline of our method is shown in Figure 2.

3.1. SDF Learning

We use popular neural pulling [23] to learn SDF f_θ by establishing projection relationships between spatial queries $\mathcal{Q}$ and the input point cloud $\mathcal{P}$. Since the gradient ∇f_θ represents the direction of the fastest increase in the signed distance, a spatial query $q \in \mathcal{Q}$ can be projected onto a surface query $s \in \mathcal{S}$ on the implicit surface with the values of $f_\theta(q)$ and $\nabla f_\theta(q)$:

$$s_{\theta,q} = q - f_\theta(q) \times \nabla f_\theta(q) / \|\nabla f_\theta(q)\|_2 \tag{1}$$

where $\nabla f_\theta(q) / \|\nabla f_\theta(q)\|_2$ is the normalized gradient. By minimizing the distance between s and its nearest neighbor p in $\mathcal{P}$, the SDF network θ can simultaneously learn the correct signed distance and the gradient:

$$\mathcal{L}_{pull} = \frac{1}{Q} \sum_{q \in Q} \|s_{\theta,q} - p\| \tag{2}$$

Hybrid Feature Representation. Although minimizing $\mathcal{L}_{pull}$ can promote SDF optimization, the MLP structure used in NeuralPull [23] limits the network's representation capability, leading to a loss of details in SDFs. To address this, we explicitly introduce a voxel grid $\mathcal{V}$ and a tri-plane $\mathcal{T}$ to store learnable features related to both spatial and planar geometry, as shown in Figure 2.a. For a spatial query q, we obtain a 32-dimensional feature vector from $\mathcal{V}$ through trilinear interpolation, and feature vectors of the same dimension from $\mathcal{T}$ (including $\mathcal{T}_{xy}$, $\mathcal{T}_{yz}$, $\mathcal{T}_{zx}$) through bilinear interpolation. The summed results are further input into the MLP g_{mlp} as additional feature information for the query q to learn the SDF value $f_\theta(q)$. The learnable feature vectors stored in $\mathcal{V}$ and $\mathcal{T}$ are randomly initialized and optimized as part of the network parameters. The entire process is formulated as follows:

$$fea(q) = TriI(q, \mathcal{V}) + \sum_{s \in xy,yz,zx}^{3} BiI(q_s, \mathcal{T}_s) \tag{3}$$

$$f_\theta(q) = g_{mlp}(Concat(q, fea(q))) \tag{4}$$

The hybrid representation combines the complementary advantages of 3D spatial features and 2D planar features to enhance geometric perception. It also effectively mitigates the side effects that each representation might introduce individually. Specifically, grid features exhibit strong spatial representation ability but struggle to maintain smoothness during the gradual refinement of the SDF, often resulting in overfitting and artifacts. In contrast, planar features help to refine the SDF smoothly and reduce artifact generation, but their reduced parameter dimensions limit the representation of local details. By combining the two features to construct information embedding, accurate implicit surfaces with rich fine-grained details can be effectively modeled.

3.2. Curvature-Adaptive Vertex Generation

Extracting resolution-adaptive meshes from SDFs is the key for constructing high-fidelity lightweight meshes. To this end, we develop an adaptive meshing algorithm that first generates any specified number of curvature-adaptive vertices on implicit surfaces and then meshes them. High-curvature areas typically contain rich details, so the vertices are expected to be densely distributed in these regions and sparsely distributed in low-curvature areas.

Point-Based Representation. The generated vertices must preserve both overall shape and salient features of the implicit surface. However, since the SDF is parameterized by a network, it is difficult to directly perceive the geometry of implicit surface. To address this, we employ point-based representation to make the implicit surface explicit. According to Equation 1-2, the SDF f_θ models the signed distance and gradient of spatial queries $\mathcal{Q}$. Therefore, the surface queries $\mathcal{S}$ obtained by projection can be used to represent the implicit surface. In our method, the input point cloud $\mathcal{P}$ is randomly perturbed to generate sufficient $\mathcal{Q}$, thereby ensuring a dense coverage of $\mathcal{S}$ on the implicit surface. Farthest point sampling (FPS) is then applied to obtain a uniform distribution of $\mathcal{S}$. This point-based explicit representation of implicit surface can provide a clear and direct constraint for vertex generation.

Surface Curvature. Curvature information represents the degree of change in the local surface normal. Thus, we calculate the normal deviation of the local point set to determine the curvature. Specifically, for a surface query s, its normal n is equivalent to the normalized gradient of the SDF. We index its K-nearest neighbors $\{s_k, k \in \mathcal{N}_s\}$ with normals $\{n_k\}$ and calculate the normal deviation $\{\delta_k\}$ based on the cosine similarity between n and $\{n_k\}$:

$$\delta_k = 1 - cos(n, n_k) = 1 - n \cdot n_k / (\|n\| \cdot \|n_k\|) \tag{5}$$

The values $\{\delta_k\}$ reflect the change in the normal in each direction at s, and are then weighted and summed using a distance-based Gaussian kernel function to produce a surrogate for the mean curvature c_s:

$$w_k = \frac{e^{-(d_k)^2/\sigma^2}}{\sum_{k\in\mathcal{N}_s} e^{-(d_k)^2/\sigma^2}} \tag{6}$$

$$c_s = \sum\nolimits_{k\in\mathcal{N}_s} w_k \cdot \delta_k \tag{7}$$

where $d_k = \|s - s_k\|_2$ is the Euclidean distance between the query s and its neighbor s_k. σ is the scaling coefficient, taking the average of $\{d_k\}$. The curvature of a surface query is considered as the curvature on the local implicit surface, guiding the distribution of vertices in mesh.

Vertex Generator. To obtain curvature-adaptive vertices, we design a vertex generator, as shown in Figure 2.b. We first use a point selector to initialize a specified number of vertices from the surface queries $\mathcal{S}$ and then adaptively adjust their positions based on the curvature information. This process can also be viewed as a resampling of $\mathcal{S}$, preserving both the high-curvature features and the overall shape. Since FPS provides a uniform distribution, we use it as the point selector implementation so that the initial vertices $\{v_o\}$ can well preserve the overall structure. Afterwards, we use a neural network γ to learn a displacement for each v_o, refining the vertex positions. For convenience, Point-TransformerV3 [46] with a plug-and-play design is used as the backbone for γ. The final positions of predicted vertices $\mathcal{V} = \{v\}$ are expressed as follows:

$$v = v_o + \gamma(v_o) \tag{8}$$

To achieve curvature awareness and adaptive position refinement, we devise two crucial optimizations for the displacement network γ. First, we minimize the distance between each surface query $s \in \mathcal{S}$ and its nearest neighbor in vertices $\mathcal{V}$, using the point curvature of s as a weighting factor. It ensures that surface queries with large curvature attract more vertices effectively, reducing the discrepancy in high-curvature features between $\mathcal{V}$ and $\mathcal{S}$. The objective function is as follows:

$$\mathcal{L}_{cur} = \frac{1}{\mathcal{S}} \sum_{s\in\mathcal{S}} c_s \cdot \min_{v\in\mathcal{V}} \|s - v\|^2 \tag{9}$$

Next, we minimize the cosine error between the normals of $\mathcal{V}$ and S, ensuring that $\mathcal{V}$ maintains normal consistency with S. This promotes a gradual change in vertex density and surface normals, resulting in a smooth surface. The required normals are computed from the normalized gradient of the SDF. The objective function is as follows:

$$\mathcal{L}_{nc} = \frac{1}{\mathcal{S}} \sum_{s\in\mathcal{S}} (1 - cos(n_s, n_{v'})) \tag{10}$$

where

$$v' = \arg\min_{v\in\mathcal{V}} \|s - v\| \tag{11}$$

In the last, for the final generated vertices, we additionally apply projection operations based on the SDF to ensure the accurate distribution on the implicit surface.

Progressive Upsampling. To ensure more vertices are distributed in high-curvature areas and facilitate position refinement, we gradually increase vertices during the optimization process. Specifically, we calculate the curvature

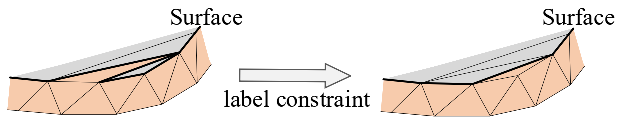

Figure 3. 2D example of neighborhood label constraint. Triangles and edges in 2D correspond to tetrahedrons and faces in 3D. The extracted surface is shown with bold black lines. Orange and gray represent different categories. Each tetrahedron's label matches the predominant category of neighbors.

for vertices $\mathcal{V}$ according to Equation 5-7, and insert the nearest neighbors of high-curvature vertices from surface queries $\mathcal{S}$ into the existing vertices. We perform such upsampling at fixed intervals until the specified vertex count is reached.

3.3. Delaunay Meshing

Given the generated vertices $\mathcal{V}$ and the SDF, we construct Delaunay triangulation $\mathcal{D}$ to produce a triangulated mesh with geometry fidelity and correct topology. As shown in Figure 2.c, $\mathcal{D}$ constructs dense tetrahedrons connecting the vertices in $\mathcal{V}$ according to strict geometric principles, with adjacent tetrahedrons sharing a common triangular face. By appropriate tetrahedral classification, an accurate surface approximation can be extracted [1].

Multi-label Voting. The complex intersection between the implicit surface and the Delaunay triangulation in space presents challenges for tetrahedral classification. A practical approach is to classify them based on the volume of tetrahedrons inside/outside the implicit surface, but direct calculation proves difficult and intricate. To address this, we propose a multi-label voting method that randomly samples multiple reference points within each tetrahedron. Since the point labels are easily determined based on their SDF values, we use these labels to vote on the tetrahedral label, assigning it to the class with the highest count. This probabilistic approach avoids the need for volume calculations, achieving simple and accurate tetrahedral classification. By extracting the triangular faces shared by tetrahedrons of different labels, a watertight and non-self-intersecting mesh can be generated.

Neighborhood Label Constraint. To ensure correct topology, the mesh must also be manifold. We observe that non-manifold edges are caused by the misclassification of some narrow tetrahedrons near the implicit surface, as shown in Figure 3. This occurs due to the non-smoothness of local implicit surfaces. For more robust classification, we propose a neighbor label constraint so that each tetrahedron's labeling is constrained by its neighbors. When a tetrahedron's four neighbors have unequal number of inside/outside labels, we adjust its label to match the majority category, promoting more compact clustering of tetrahedrons within the same category. As a result, it corrects the misclassification of narrow tetrahedrons and extracts smoother, more manifold surface.

3.4. Losses

For our SDF network, we adopt the loss $\mathcal{L}_{pull}$ defined in Equation 2 as the primary optimization objective. To enforce gradient consistency and enable more accurate spatial projection, we introduce an additional loss $\mathcal{L}_{grad}$ to align the gradients of spatial queries $\mathcal{Q}$ and surface queries $\mathcal{S}$.

$$\mathcal{L}_{grad} = \frac{1}{\mathcal{Q}} \sum_{q \in \mathcal{Q}} 1 - cos(\nabla f_\theta(q), \nabla f_\theta(s)) \tag{12}$$

where $s \in \mathcal{S}$ is the surface query corresponding to q. Then our total loss function for SDF learning is:

$$\mathcal{L}_{sdf} = \lambda_1 \mathcal{L}_{pull} + \lambda_2 \mathcal{L}_{grad} \tag{13}$$

For the displacement network in our vertex generator, we adopt the losses $\mathcal{L}_{cur}$ and $\mathcal{L}_{nc}$ defined in Equation 9 and 10 to guide the optimization. They ensure vertices $\mathcal{V}$ concentrate in high-curvature areas and promote smooth distribution. Moreover, to ensure that $\mathcal{V}$ closely adheres to the implicit surface while maintaining overall shape, we further constrain the chamfer distance between $\mathcal{V}$ and the surface queries $\mathcal{S}$ with the loss $\mathcal{L}_{cd}$ as follows:

$$\mathcal{L}_{cd} = \frac{1}{\mathcal{S}} \sum_{s \in \mathcal{S}} \min_{v \in \mathcal{V}} \|s - v\|^2 + \frac{1}{\mathcal{V}} \sum_{v \in \mathcal{V}} \min_{s \in \mathcal{S}} \|s - v\|^2 \tag{14}$$

Finally, to prevent $\mathcal{V}$ from clustering too densely in high-curvature areas, we introduce a repulsion loss $\mathcal{L}_{rep}$. This also ensures the vertices are evenly distributed and helps maintain uniform angles of triangles in the mesh.

$$L_{rep} = -\frac{1}{\mathcal{V}} \sum_{v_i \in \mathcal{V}} \min_{v_j \in \mathcal{V}, v_i \neq v_j} \|v_i - v_j\|^2 \tag{15}$$

Then our total loss function for vertex generation is:

$$\mathcal{L}_{vg} = \lambda_a \mathcal{L}_{cur} + \lambda_b \mathcal{L}_{nc} + \lambda_c \mathcal{L}_{cd} + \lambda_d \mathcal{L}_{rep} \tag{16}$$

4. Experiments

4.1. Experimental Setting

Datasets and Metrics. We conduct experiments on four challenging datasets, including Stanford [11], Thingi10K [51], SRB [45] and ScanNet [12]. For evaluation, we use the most common metrics with $100k$ points sampled on meshes, ie., chamfer distance (CD), normal consistency (NC), and F1-score (F1). Additionally, to evaluate feature preservation, the L1 curvature error (CE) is calculated as proposed by Mark Pauly [33]. We also evaluate the percentage of meshes that exhibit correct topology (CT), defined by three properties: watertightness (W), manifoldness (M), and non-self-intersection (NS). In our tables, when the correct topology cannot be ensured (i.e., the CT value is not 1.0), we

Method	CD ↓ (10^{-5})	NC ↑	F1 ↑	CE ↓ (10^{-3})	CT ↑ W/M/NS	Vertex (10^3)	Face (10^3)
DMTet [39]	4.912	0.937	0.793	0.637	**1.0**	7.4	14.8
MC [21]	1.167	0.952	0.854	0.709	**1.0**	5.4	10.8
NDC [7]	0.849	0.961	0.908	0.312	1.0/0.1/0.1	5.4	10.8
NMC [6]	0.776	0.969	0.923	0.197	0.0/0.0/0.0	36.8	73.6
VoroMesh [26]	1.021	0.939	0.906	0.585	**1.0**	39.4	78.9
PoNQ [27]	0.758	0.971	0.924	0.201	1.0/0.5/1.0	20.4	40.9
Ours-lite	0.752	0.976	0.930	0.124	**1.0**	5.4	10.8
Ours	**0.682**	**0.985**	**0.938**	**0.067**	**1.0**	20.4	40.9
DMTet [39]	14.488	0.883	0.530	1.119	**1.0**	1.9	3.8
MC [21]	9.089	0.905	0.604	0.859	**1.0**	1.3	2.6
NDC [7]	6.390	0.920	0.745	0.829	1.0/0.2/0.4	1.3	2.6
NMC [6]	5.188	0.936	0.796	0.653	0.1/0.2/0.0	8.7	17.3
VoroMesh [26]	2.825	0.902	0.758	0.817	**1.0**	9.9	19.9
PoNQ [27]	1.344	0.942	0.810	0.575	1.0/0.6/1.0	5.0	10.0
Ours-lite	1.301	0.958	0.832	0.315	**1.0**	1.3	2.6
Ours	**0.755**	**0.975**	**0.929**	**0.126**	**1.0**	5.0	10.0

Table 1. Quantitative comparison on Thingi10K. Above the double horizontal line, the comparison methods use a grid resolution of 64, while below, their grid resolution is set to 32. In addition, the comparison methods use GT SDF values for meshing, while our method uses learned SDFs from point clouds. Best accuracy results are highlighted as **first** and second.

separately present the percentage of meshes that meet each of the three sub-metrics. In addition, the number of vertices and faces in meshes are also counted.

Implement Details. Our SDF network follows the settings of NeuralPull [23] and performs for $20k$ iterations. In our vertex generation, the displacement network iterates $6k$ times for each object and the learning rate is set to 0.001. We perform vertex upsampling every $1k$ iterations, increasing the number of vertices by 20% each time, for a total of five times. Surface queries $\mathcal{S}$ are downsampled to $0.5M$ points by FPS, and the number of K-nearest neighbors for curvature calculation is set to 32. Additionally, during the meshing process, the number of reference points sampled inside the Delaunay tetrahedron is set to 101 to ensure sufficiency. The weight parameters $\{\lambda_1, \lambda_2\}$ in the loss $\mathcal{L}_{sdf}$ are set to $\{1, 0.001\}$, and $\{\lambda_a, \lambda_b, \lambda_c, \lambda_d\}$ in the loss $\mathcal{L}_{vg}$ are set to $\{100, 100, 1, 1\}$.

4.2. Meshing Comparison

To evaluate the ability of our method in reconstructing lightweight meshes, we first compare the results with different meshing methods using the same number of mesh elements. Classic MC [21] and recent neural meshing methods are compared, including DMTet [39], NMC [6], NDC [7], VoroMesh [26], and PoNQ [27]. Following the settings of VoroMesh, we conduct experiments on a subset of the Thingi10K dataset [51], which contains 30 complex shapes. To increase the challenge, we use the ground truth (GT) SDF values generated from the GT meshes as input for the comparison methods, while our method learns SDFs using $100k$ unoriented points sampled on the meshes.

Since current meshing methods still rely on voxel grids, we extract their surfaces with grid resolutions of 64 and 32 for evaluation, as presented in Table 1. Instead, our method can freely set the number of mesh elements. For a fair comparison, the number of elements in each of our meshes is set to be the same as that in PoNQ, which performs the best among comparison methods. In addition, since the number of mesh elements generated by MC, DMTet, and NDC at the same grid resolution is significantly fewer than that of other methods, we match the number of our mesh elements with that of MC for a more comprehensive comparison, denoted as 'Ours-lite'. The results show that our method demonstrates significant advantages in lightweight mesh reconstruction and stronger robustness against resolution reduction. In terms of curvature error (CE) on low-resolution meshes, our method outperforms PoNQ by 78%, while 'Ours-lite' achieves a 45% improvement with only 1/4 of the elements, demonstrating superior preservation of detailed features. The visual results in Figure 4 and 5 also provide convincing evidence. Notably, the SDFs used in our method are learned from unoriented point clouds, whereas other methods rely on the GT data. Despite this, more sharpness and detail are recovered by our method, demonstrating superior capability in reconstructing lightweight meshes. As another important advantage, our method produces meshes with correct topology, ensuring watertightness, manifoldness, and non-self-intersection.

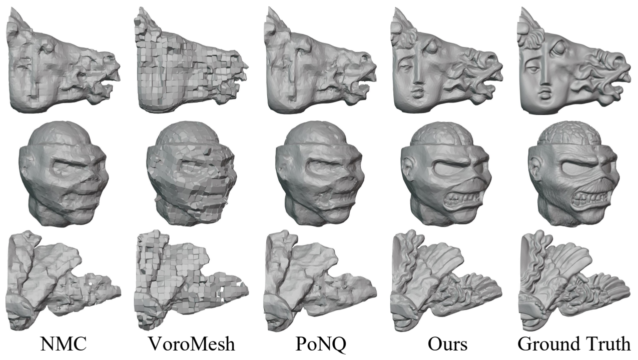

Figure 4. Visual results on Thingi10K at a grid resolution of 32.

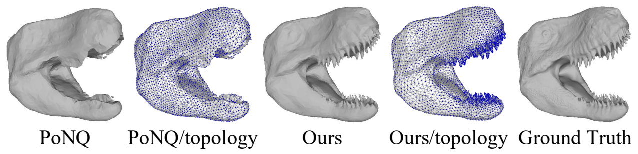

Figure 5. Visual comparison of our method with PoNQ. The blue dots represent vertices. Our method generates curvature-adaptive vertices, capturing more details with the same number of elements.

4.3. Implicit Reconstruction Comparison

In Section 4.2, we provide GT SDF values as input for the comparison meshing methods to increase the challenge. However, learning SDFs from point clouds is inherently challenging. To further evaluate the ability of our method in lightweight reconstruction, we compare our lightweight meshes with dense meshes produced by various neural implicit reconstruction methods. We conduct experiments on

Method	CD ↓ (10^{-5})	NC ↑	F1 ↑	CE ↓ (10^{-3})	Vertex (10^4)	Face (10^4)
PCP [25]	3.767	0.938	0.742	1.732	98.8	197.6
NeuralIMLS [44]	1.072	0.957	0.933	0.442	60.9	121.8
GridPull [4]	1.063	0.943	0.874	0.422	32.2	64.5
DiGS [2]	0.698	0.957	0.950	0.542	138.3	276.6
PINC [32]	0.631	0.962	0.950	0.557	124.0	248.0
NeuralPull [23]	0.606	0.963	0.950	0.329	89.6	179.3
Ours / MC512	**0.558**	0.967	0.958	**0.106**	86.8	182.6
Ours / AM15%	**0.558**	**0.969**	**0.959**	0.113	12.6	25.1
Ours / AM10%	0.559	0.968	**0.959**	0.125	8.6	17.2
Ours / AM5%	0.560	0.966	**0.959**	0.154	4.4	8.9

Table 2. Quantitative comparison on the Stanford dataset. The compared methods and 'Ours / MC512' use 512-resolution MC to extract dense meshes, while 'Ours / AM' uses adaptive meshing (AM) to extract lightweight meshes, with three different vertex count ratios compared to our dense meshes. Best accuracy results are highlighted as **first**, second, third and fourth.

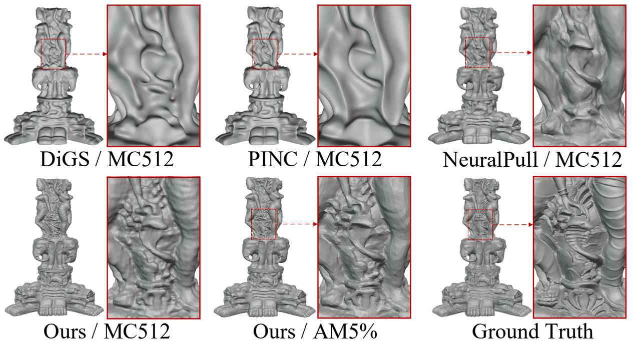

Figure 6. Visual comparison on the Stanford dataset.

the Stanford dataset [11] consisting of challenging data with complex details. Each object is sampled $200k$ unoriented points, and recent neural implicit reconstruction methods are compared, including NeuralPull [23], PCP [25], DiGS [2], NeuralIMLS [44], GridPull [4], and PINC [32].

To maximize reconstruction accuracy, we first use a regular 512-resolution MC for all methods to extract dense meshes. As shown in Table 2, they all result in a large number of mesh elements. Next, we use our adaptive meshing (AM) to extract lightweight meshes for comparison, setting the vertex counts to 15%, 10%, and 5% of those in our dense mesh. Despite using significantly fewer mesh elements, our adaptive mesh still achieves competitive results compared to high-resolution MC. It also exhibits strong robustness to variations in the number of mesh elements. Compared to the dense meshes produced by other methods, our method achieves superior results even with only 5% of the element count. A visual comparison is shown in Figure 6. The significant advantages in detail preservation highlight our method's ability to reconstruct high-fidelity lightweight meshes from point clouds.

4.4. Explicit Reconstruction Comparison

Another manner for reconstructing lightweight meshes is to combine point cloud downsampling with point set triangulation. In contrast to such fully explicit methods, our method learns SDFs and adaptive meshing from point clouds, organically integrating explicit and implicit reconstruction. To further evaluate our method, we construct experiments on the SRB dataset [45], which includes noisy and complex real scans. Various combinations of point cloud downsampling and point set triangulation methods are compared. For downsampling, we use the classic FPS and the recent learning-based method MetaSample [8]. For triangulation, we use the classic Ball pivoting [3] and recent learning-based methods, including PointTriNet [38] and DSE [37]. The number of downsampled points is set to $5k$, and our method generates the same number of vertices. The comparison results are shown in Table 3 and Figure 7. Our method significantly improves reconstruction accuracy and smoothness, with the generated vertices adaptively distributed along edges and high-curvature regions. In addition, our method well addresses the challenge of ensuring correct mesh topology in explicit reconstruction.

	Method	CD ↓ (10^{-5})	NC ↑	F1 ↑	CE ↓ (10^{-3})	CT ↑ W/M/NS
FPS	Ball pivoting [3]	4.931	0.920	0.696	0.809	0.0/1.0/1.0
FPS	PointTriNet [38]	4.004	0.942	0.712	0.348	0.0/1.0/1.0
FPS	DSE [37]	4.100	0.934	0.722	0.767	0.0/1.0/1.0
MS [8]	Ball pivoting [3]	5.433	0.923	0.672	0.799	0.0/1.0/1.0
MS [8]	PointTriNet [38]	4.135	0.935	0.696	0.413	0.0/1.0/1.0
MS [8]	DSE [37]	4.246	0.925	0.696	1.002	0.0/1.0/1.0
Ours		**2.561**	**0.964**	**0.747**	**0.211**	**1.0**

Table 3. Quantitative comparison on the SRB dataset. 'MS' represents MetaSample. All results have the same number of vertices.

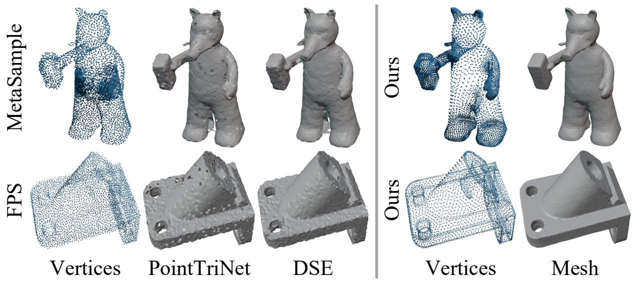

Figure 7. Visual results on the SRB dataset. Our vertices achieve a curvature-adaptive distribution, allowing the resulting lightweight meshes to well preserve edges and sharpness.

4.5. Extensions and Ablation Studies

Scalability. To evaluate the ability of our method to reconstruct large-scale scenes, we conduct experiments using 1 million points sampled from ScanNet [12]. In Figure 8, the left side shows the dense meshes of NeuralPull [23], GridPull [4], and our method using 512-resolution MC to compare the accuracy of SDF modeling. The right side compares the lightweight meshes generated by our adaptive meshing (AM) and 128-resolution MC with the same number of elements. The significant advantages in both compar-

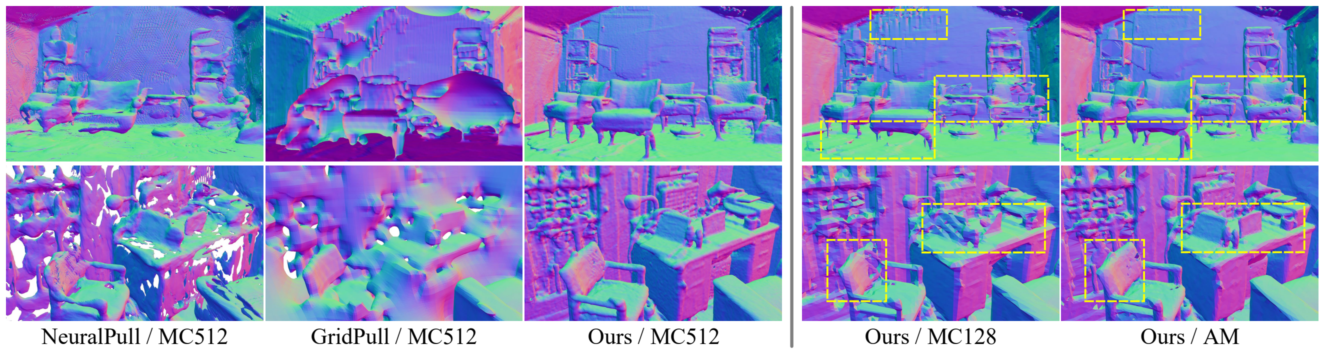

Figure 8. Visual results on Scannet. 'MC512' represents 512-resolution MC, used to extract dense meshes for comparing the learned SDFs. 'MC128' and 'AM' refer to 128-resolution MC and our adaptive meshing, both of which generate the same number of mesh elements.

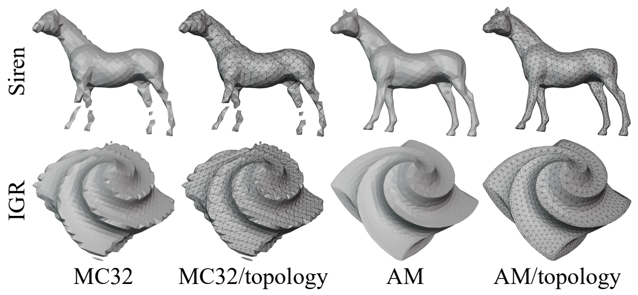

Figure 9. Visual comparison of our adaptive meshing (AM) and 32-resolution MC across different neural implicit reconstruction methods. Both have the same number of mesh elements.

Method	FPS	$\mathcal{L}_{nc}$	$\mathcal{L}_{cur}$	PU	CD ↓ (10^{-5})	F1 ↑	CE ↓ (10^{-3})
Model-I	✓				0.9643	0.8929	0.1586
Model-II	✓	✓			0.8551	0.9198	0.1421
Model-III	✓		✓		0.8147	0.9194	0.1370
Model-IV	✓	✓	✓		0.7989	0.9267	0.1344
Ours	✓	✓	✓	✓	**0.7550**	**0.9292**	**0.1256**

Table 4. Ablation experiments about vertex generator.

	w/o both	w / grid	w / tri-plane	w / both
CD (10^{-5}) ↓	0.5894	0.8282	0.5639	**0.5580**
CE (10^{-3}) ↓	0.3052	0.1269	0.2053	**0.1059**

Table 5. Ablation experiments about SDF network.

isons demonstrate the good scalability of our method. More comparisons are shown in the supplementary material.

Universality of Adaptive Meshing. We apply the proposed adaptive meshing (AM) algorithm to two early neural implicit reconstruction methods, Siren [41] and IGR [14], to verify its universality. For a fair comparison, we generate the same number of mesh elements as MC at a resolution of 32. The lightweight meshes of both methods extracted from the SDFs are shown in Figure 9. Our method effectively captures high-curvature features on the implicit surface, preserving fine structures and sharp edges.

Ablation Study of Vertex Generator. Table 4 evaluates the impact of various components within our vertex generator on the Thingi10K dataset [51]. FPS is the point selector implementation that initializes vertices from surface queries $\mathcal{S}$. $\mathcal{L}_{cur}$ and $\mathcal{L}_{nc}$ represent the two key losses that guide the displacement network γ to learn the adaptive movement of vertices. It is evident that each loss enables the vertices to learn better position refinements, resulting in a more accurate mesh. Moreover, the significant improvement in accuracy of Model-IV compared to Model-I demonstrates that curvature-adaptive vertex distribution plays a crucial role in realistic lightweight mesh construction. PU is our progressive upsampling strategy, which further enhances the high-quality representation of meshes.

Ablation Study of SDF Network. To justify the effectiveness of our proposed hybrid features in SDF network θ, we conduct ablation experiments on the Stanford dataset [11]. The grid features and tri-plane features are removed separately to test the effect of each component. Table 5 shows the quantitative results of high-resolution meshes extracted by 512-resolution MC. Grid features capture details better than tri-plane features, leading to lower curvature error (CE) values. However, they show lower accuracy in terms of chamfer distance (CD), due to the introduction of overfitting and artifacts. In contrast, hybrid features effectively avoid these issues and preserve detailed representations, leading to overall performance improvements.

5. Conclusion

We introduce a new solution for high-fidelity ***L**ightweight* ***M**esh* ***R**econstruction* (***LMR***) from point clouds. At its core, an adaptive meshing method accurately perceives the geometry of implicit surfaces in SDFs and extracts resolution-adaptive meshes based on surface curvature. This enables the recovery of precise geometry with fewer mesh elements. Moreover, a hybrid representation effectively enhances detail capture in SDFs, contributing to better lightweight reconstruction. We hope this work provides new perspectives and insights to assist the community in addressing some reconstruction challenges in practical applications.

Acknowledgements. This work was supported by the National Natural Science Foundation of China under Grant 62176096.

References

[1] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 39–48, 1998. 5

[2] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. Digs: Divergence guided shape implicit neural representation for unoriented point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19323–19332, 2022. 7

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 2, 7

[4] Chao Chen, Yu-Shen Liu, and Zhizhong Han. Gridpull: Towards scalability in learning implicit representations from 3d point clouds. In *Proceedings of the ieee/cvf international conference on computer vision*, pages 18322–18334, 2023. 1, 2, 7

[5] Chao Chen, Yu-Shen Liu, and Zhizhong Han. Neuraltps: Learning signed distance functions without priors from single sparse point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 2

[6] Zhiqin Chen and Hao Zhang. Neural marching cubes. *ACM Transactions on Graphics (TOG)*, 40(6):1–15, 2021. 1, 2, 6

[7] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 1, 2, 6

[8] Ta-Ying Cheng, Qingyong Hu, Qian Xie, Niki Trigoni, and Andrew Markham. Meta-sampler: Almost-universal yet task-oriented sampling for point clouds. In *European Conference on Computer Vision*, pages 694–710. Springer, 2022. 7

[9] Gene Chou, Ilya Chugunov, and Felix Heide. Gensdf: Two-stage learning of generalizable signed distance functions. *Advances in Neural Information Processing Systems*, 35: 24905–24919, 2022. 1

[10] Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2262–2272, 2023. 1

[11] 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. 5, 7, 8

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 5, 7

[13] Qiujie Dong, Rui Xu, Pengfei Wang, Shuangmin Chen, Shiqing Xin, Xiaohong Jia, Wenping Wang, and Changhe Tu. Neurcadrecon: Neural representation for reconstructing cad surfaces by enforcing zero gaussian curvature. *arXiv preprint arXiv:2404.13420*, 2024. 2

[14] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3789–3799, 2020. 2, 8

[15] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, 2002. 1, 2

[16] Jonas Kulhanek and Torsten Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18458–18469, 2023. 2

[17] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7578–7588, 2020. 2

[18] Hai Li, Xingrui Yang, Hongjia Zhai, Yuqian Liu, Hujun Bao, and Guofeng Zhang. Vox-surf: Voxel-based implicit surface representation. *IEEE Transactions on Visualization and Computer Graphics*, 30(3):1743–1755, 2022. 2

[19] Shengtao Li, Ge Gao, Yudong Liu, Ming Gu, and Yu-Shen Liu. Implicit filtering for learning neural signed distance functions from 3d point clouds. In *European Conference on Computer Vision*, pages 234–251. Springer, 2025. 2

[20] Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1788–1797, 2021. 1

[21] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998. 1, 2, 6

[22] Yiming Luo, Zhenxing Mi, and Wenbing Tao. Deepdt: Learning geometry from delaunay triangulation for surface reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2277–2285, 2021. 2

[23] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-pull: Learning signed distance function from point clouds by learning to pull space onto surface. In *International Conference on Machine Learning*, pages 7246–7257. PMLR, 2021. 2, 3, 6, 7

[24] Baorui Ma, Yu-Shen Liu, and Zhizhong Han. Reconstructing surfaces for sparse point clouds with on-surface priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6315–6325, 2022. 1

[25] Baorui Ma, Yu-Shen Liu, Matthias Zwicker, and Zhizhong Han. Surface reconstruction from point clouds by learning predictive context priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6326–6337, 2022. 7

[26] Nissim Maruani, Roman Klokov, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. Voromesh: Learning watertight surface meshes with voronoi diagrams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14565–14574, 2023. 1, 2, 6

[27] Nissim Maruani, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. Ponq: a neural qem-based mesh representation.

In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3647–3657, 2024. 2, 6

[28] Gal Metzer, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Self-sampling for neural point cloud consolidation. *ACM Transactions on Graphics (TOG)*, 40(5):1–14, 2021. 2

[29] 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. 2

[30] Ehsan Nezhadarya, Ehsan Taghavi, Ryan Razani, Bingbing Liu, and Jun Luo. Adaptive hierarchical down-sampling for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12956–12964, 2020. 2

[31] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019. 1

[32] Yesom Park, Taekyung Lee, Jooyoung Hahn, and Myungjoo Kang. p-poisson surface reconstruction in curl-free flow from point clouds. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 60077–60098, 2023. 7

[33] Mark Pauly, Markus Gross, and Leif P Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization, 2002. VIS 2002.*, pages 163–170. IEEE, 2002. 5

[34] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *Advances in Neural Information Processing Systems*, 34:13032–13044, 2021. 1

[35] Rolandos Alexandros Potamias, Giorgos Bouritsas, and Stefanos Zafeiriou. Revisiting point cloud simplification: A learnable feature preserving approach. In *European Conference on Computer Vision*, pages 586–603. Springer, 2022. 2

[36] Albert Pumarola, Artsiom Sanakoyeu, Lior Yariv, Ali Thabet, and Yaron Lipman. Visco grids: Surface reconstruction with viscosity and coarea grids. *Advances in Neural Information Processing Systems*, 35:18060–18071, 2022. 1

[37] Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy J Mitra, and Maks Ovsjanikov. Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22–31, 2021. 7

[38] Nicholas Sharp and Maks Ovsjanikov. Pointtrinet: Learned triangulation of 3d point sets. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 762–778. Springer, 2020. 2, 7

[39] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021. 6

[40] Ruoxi Shi, Xinyue Wei, Cheng Wang, and Hao Su. Zerorf: Fast sparse view 360deg reconstruction with zero pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21114–21124, 2024. 2

[41] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 2, 8

[42] Sanghyun Son, Matheus Gadelha, Yang Zhou, Zexiang Xu, Ming C Lin, and Yi Zhou. Dmesh: A differentiable representation for general meshes. *arXiv e-prints*, pages arXiv–2404, 2024. 2

[43] Yiqun Wang, Ivan Skorokhodov, and Peter Wonka. Pet-neus: Positional encoding tri-planes for neural surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12598–12607, 2023. 2

[44] Zixiong Wang, Pengfei Wang, Pengshuai Wang, Qiujie Dong, Junjie Gao, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Wenping Wang. Neural-imls: Self-supervised implicit moving least-squares network for surface reconstruction. *IEEE Transactions on Visualization & Computer Graphics*, (01):1–16, 2023. 2, 7

[45] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10130–10139, 2019. 5, 7

[46] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4840–4851, 2024. 4

[47] Wang Yifan, Shihao Wu, Cengiz Oztireli, and Olga Sorkine-Hornung. Iso-points: Optimizing neural implicit surfaces with hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–383, 2021. 2

[48] Chen Zhang and Wenbing Tao. Learning meshing from delaunay triangulation for 3d shape representation. *International Journal of Computer Vision*, pages 1–24, 2025. 2

[49] Chen Zhang, Ganzhangqin Yuan, and Wenbing Tao. Dmnet: Delaunay meshing network for 3d shape representation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14372–14382. IEEE Computer Society, 2023. 2

[50] Chen Zhang, Wanjuan Su, Qingshan Xu, Xinyao Liao, and Wenbing Tao. Pg-neus: Robust and efficient point guidance for multi-view neural surface reconstruction. *IEEE Transactions on Visualization & Computer Graphics*, (01):1–15, 2024. 2

[51] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 5, 6, 8