OffsetOPT: Explicit Surface Reconstruction without Normals

Huan Lei
AIML, University of Adelaide
huan.lei@adelaide.edu.au
https://github.com/EnyaHermite/OffsetOPT

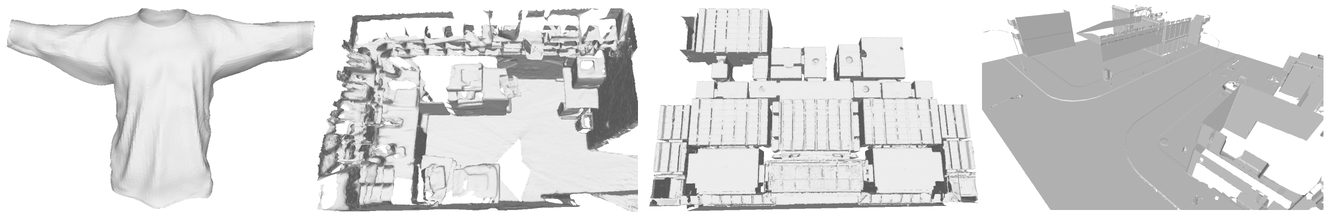
Figure 1. We present OffsetOPT (Offset OPTimization) for explicit surface reconstruction from 3D point clouds, without the need for point normals. The prediction model is trained on synthetic meshes with supervision and then generalized to unseen point clouds through unsupervised optimization of per-point offsets. All surfaces in this figure are reconstructed using the same trained model with offset optimization. Our method achieves state-of-the-art performance in overall surface quality, sharp detail preservation, and scalability.

Abstract

Neural surface reconstruction has been dominated by implicit representations with marching cubes for explicit surface extraction. However, those methods typically require high-quality normals for accurate reconstruction. We propose ***OffsetOPT****, a method that reconstructs explicit surfaces directly from 3D point clouds and eliminates the need for point normals. The approach comprises two stages: first, we train a neural network to predict surface triangles based on local point geometry, given uniformly distributed training point clouds. Next, we apply the frozen network to reconstruct surfaces from unseen point clouds by optimizing a per-point offset to maximize the accuracy of triangle predictions. Compared to state-of-the-art methods, OffsetOPT not only excels at reconstructing overall surfaces but also significantly preserves sharp surface features. We demonstrate its accuracy on popular benchmarks, including small-scale shapes and large-scale open surfaces.*

1. Introduction

Surface reconstruction from 3D point clouds is essential in applications across computer vision, graphics, and robotics. Traditional solutions to this problem include computational methods such as ball-pivoting [3] and Delaunay triangulation [12], along with the classic Poisson method [22, 23], which estimates an implicit indicator function by solving a linear system. The success of Poisson surface reconstruction [22] in industry has resulted in the predominant exploration of implicit neural representations in geometric deep learning [20, 30, 33, 42]. These methods generally require high-quality, oriented normals to predict a scalar field as the implicit surface representation. Explicit surfaces are then extracted using Marching Cubes [27] or its variants [21, 40].

However, Marching Cubes becomes incompatible with the unsigned distance fields (UDFs) [13] in open-surface reconstruction because it relies on sign changes across the surface for mesh extraction. Researchers have explored various methods [18, 20, 43] to incorporate signs into UDFs such that Marching Cubes can still be applied. In contrast to the widespread interest in implicit neural representations, neural computational methods for explicit surface reconstruction have been largely overlooked [25, 26, 39, 41], despite their bypass of implicit representations, no reliance on point normals, and good generalization to diverse surfaces. The key limitations restricting their applications are: (i) a strong bias toward input points distributed like those from Poisson disk sampling [49], which is impractical, (ii) insufficient or inconvenient handling of edge-manifoldness in reconstructed surfaces.

With this work, we contribute a novel neural computational method, OffsetOPT, for explicit surface reconstruction directly from 3D point clouds. The method does not rely on normals but reconstructs surfaces of both shapes and open scenes with high accuracy. More importantly, it demonstrates remarkable performance in producing edge-manifold triangles without requiring specific handling [25, 26, 39, 41]. OffsetOPT consists of two stages. First, we train a transformer-based network [44] to predict triangle faces adjacent to each point based on local geometry. All of our training point clouds [24] approximate an uniform distribution. Second, we freeze the network to reconstruct surfaces from unseen point clouds by optimizing per-point offsets to enhance the quality of triangle predictions.

Compared to state-of-the-art methods, the proposed OffsetOPT excels not only in overall surface reconstruction but also in preserving sharp surface details. We demonstrate its accuracy on popular benchmarks, using shapes from ABC [24], FAUST [5], and MGN [4], as well as large-scale scene surfaces from ScanNet [14], Matterport3D [8], and CARLA [15]. Below are our main contributions:

- A novel neural computational method is proposed for surface reconstruction from general 3D point clouds.
- It does not rely on point normals or Poisson-disk sampled points for accurate surface reconstruction.
- Unlike the previous computational methods, it produces edge-manifold triangles without explicit handling.
- It outperforms existing approaches in reconstructing the overall structure and fine details of surfaces.

2. Related Work

Surface reconstruction from 3D point clouds is a central research topic in geometry processing. Existing solutions for this problem generally fall into two categories: (a) computational methods that directly reconstruct explicit surfaces from point clouds; (b) implicit methods that solve for different scalar fields to represent surfaces implicitly (*e.g.*, binary occupancy, signed distance, or unsigned distance fields). The latter must resort to Marching Cubes [27] for explicit surface extraction from the scalar fields. They also depend on consistently oriented normals for reliable performance.

Traditional methods in the computational category include Alpha shapes [16], the ball pivoting algorithm [3], and Delaunay triangulation [7, 12], while Poisson surface reconstruction [22, 23] is a representative approach in the implicit category. Below, we briefly review neural methods advancing the two directions in geometric deep learning.

2.1. Implicit Neural Representations

Implicit neural representations initially focused on watertight surface reconstruction, where neural networks are used to predict occupancy fields or signed distance fields (SDFs) for the surface. Researchers have developed multiple loss functions to learn the underlying implicit surface from oriented, dense point clouds [1, 2, 17, 30, 33, 42].

For improved reconstruction, various encoder architectures have been explored to extract more effective local features for each point, allowing the decoder to predict the occupancy status or signed distance associated with each point more accurately [6, 10, 34, 46]. Meanwhile, inspired by the success of basis functions in SPSR [22], a number of works have applied (learnable) basis functions or neural kernels to surface reconstruction [19, 20, 35, 47, 48].

Given the significance of open surface reconstruction in real-world applications, research on unsigned distance fields (UDFs) has gained increasing attention [13, 28, 50]. However, a key limitation of UDFs is their incompatibility with Marching Cubes due to the absence of sign information. The primary solution to this issue is to convert UDFs into SDFs by introducing signs [18, 20, 43, 45]. Among implicit methods, NKSR [20] is a notable contribution that holds strong inductive bias. It leverages well-oriented normals to establish SDFs for surfaces with varying topologies, including both watertight and open surfaces. While the method remains effective without normals, it prefers high-quality normals for superior performance. For their representativeness, we compare our reconstruction accuracy with the implicit methods, SPSR [22] and NKSR [20].

A known drawback of implicit methods is that they tend to oversmooth the sharp surface details [38], which we show empirically in our experiments. It is worth noting that we focus on reconstructing surfaces from general 3D point clouds, assuming they adequately capture the underlying surface. Surface reconstruction from noisy and sparse point clouds [9, 32] is beyond the scope of this work.

2.2. Neural Computational Reconstruction

In contrast to the popularity of implicit neural representations, neural computational reconstruction has received less attention. Existing methods generally establish candidate triangles based on the KNN neighborhood information [36] of each point. PointTriNet [41] proposes candidate triangle faces using a proposal network and classifies surface triangles with a separate network. It handles edge-manifoldness between triangles explicitly in the proposal network. IER [26] predicts whether a candidate triangle face belongs to the reconstructed surface based on the intrinsic-extrinsic distance ratio. The complexity for establishing candidate triangles is $\mathcal{O}(K^2)$ per point, and the method is highly restricted to Poisson-disk sampled points. DSE [39] parameterizes the 3D neighborhood of each point onto a 2D chart, enabling the use of Delaunay triangulation [29]. Both IER and DSE rely on a voting-based mechanism during post-processing to handle edge-manifoldness in the reconstructed surface. Different from the combinatorial formulation of previous methods, CircNet [25] exploits the duality between a triangle and its circumcenter to reformulate the reconstruction as a detection of triangle circumcenters, resulting in reduced complexity. Therefore, it is much faster at producing the primitive surfaces. Yet, the edge-manifoldness handling at post-processing takes time.

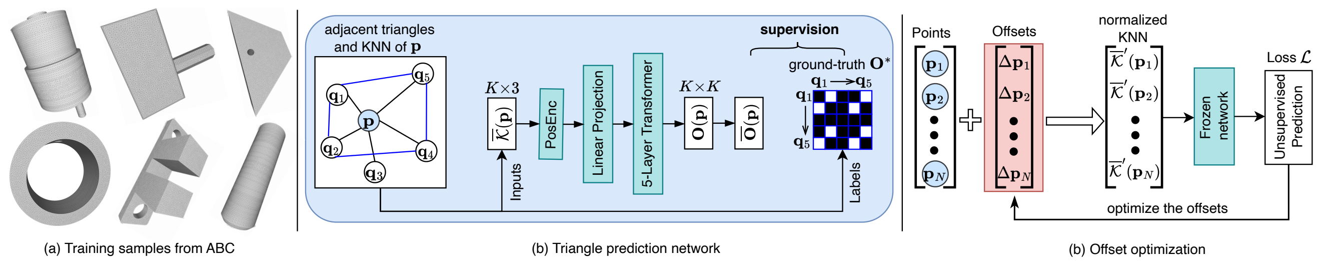

Figure 2. Overview of the proposed OffsetOPT method. (a) provides examples of training samples from the ABC dataset, showing meshes with uniformly distributed points and equilateral triangles (zoom-in for a better view). (b) is the training of our triangle prediction network in a supervised manner, where ground-truth labels are established from adjacent triangles of each point in the training meshes. The network predicts surface triangles based on KNN neighborhoods of points. (c) is the offset optimization for surface reconstruction. For a point cloud $\{\mathbf{p}_n\}$, we optimize its offsets $\{\Delta\mathbf{p}_n\}$ by backpropagating the unsupervised prediction loss through the frozen network. For each offset update during optimization, the KNN geometry used by the network is recomputed with points $\{\mathbf{p}_n + \Delta\mathbf{p}_n\}$.

The neural computational methods exhibit strong generalization to unseen point clouds across diverse shapes and scenes. In contrast, implicit neural representations are highly data-driven and constrained by their training priors when applied to unseen data. For instance, NKSR [20] required extensive training on a combined dataset of varied shapes and scenes to establish a robust inductive bias. Currently, a key limitation hindering the applications of computational methods is their over-reliance on input points being ideally distributed, simulating a Poisson disk sampling [49].

To justify the promise of neural computational reconstruction, we present OffsetOPT. Similar to previous approaches, it reconstructs the surface based on the local point geometry derived from their KNN neighborhoods. The network is trained to predict surface triangles from the $\mathcal{O}(K^2)$ candidates. Our proposed *offset optimization* effectively addresses edge-manifoldness without requiring explicit handling. Besides, it extends the applicability of neural computational reconstruction to general point clouds, rather than being restricted to those produced by the Poisson method.

3. Explicit Surface Reconstruction

Overview. Computational reconstruction preserves the input points as mesh vertices, avoiding the need for interpolating new points as in implicit methods [23]. It directly reconstructs the explicit surface from its point cloud representation by predicting adjacent triangle faces for each point. Our method involves training a transformer-based network on point clouds with an approximately ideal uniform distribution, followed by the optimization of 3D coordinate offsets for each input point. We train the network [44] with Binary Cross-Entropy (BCE) to predict triangle faces based on local point geometry, as detailed in §3.1. The trained network is then frozen, and we optimize per-point offsets to improve the prediction accuracy, extending explicit surface reconstruction to general point clouds while achieving satisfactory edge-manifoldness (see §3.2). Here, edge-manifoldness refers to each edge being adjacent to at most two triangles in the reconstructed surface.

3.1. Triangle Prediction Network

Local point geometry. The local geometry of a point has been widely exploited in geometric deep learning to learn expressive feature representations [25, 37, 39]. We therefore utilize input features derived from the neighborhood of each point to predict surface triangles with a neural network. Let $\mathcal{K}(\mathbf{p}) = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_K\}$ be the KNN neighborhood of point $\mathbf{p}$, where points are sorted such that $\|\mathbf{q}_k - \mathbf{p}\| \le \|\mathbf{q}_{k+1} - \mathbf{p}\|$. To ensure the prediction robustness across different data resolutions, we normalize the KNN neighborhood by the smallest non-zero distance as

$$\overline{\mathcal{K}}(\mathbf{p}) = \left\{\overline{\mathbf{q}}_k | \overline{\mathbf{q}}_k = \eta_0 \frac{\mathbf{q}_k - \mathbf{p}}{\|\mathbf{q}_1 - \mathbf{p}\|}\right\}_{k=1}^{K}, \tag{1}$$

with η_0 fixed as 0.01. The neural network then takes these normalized KNN coordinates with positional encoding [31] to predict the surface triangles, similar to [25].

Candidate triangles. Given the K neighbors of a point $\mathbf{p}$, all combinations yield $\binom{K}{2}$ candidate triangles. However, for better control of the edge-manifoldness, we predict K^2 triangles, represented by a symmetric matrix of size $K\times K$. Each entry (i, j) of the matrix for point $\mathbf{p}$ corresponds to a triangle formed by $(\mathbf{p}, \mathbf{q}_i, \mathbf{q}_j)$, which is identical to the one in entry (j, i). Triangles in the same row, such as those involving $\mathbf{q}_3$, share the edge $(\mathbf{p}, \mathbf{q}_3)$. The diagonal entries correspond to degenerate triangles, where two vertices are identical and hence ignored. For each edge $(\mathbf{p}, \mathbf{q}_i)$, we extract either two triangles or none. Our network predicts the probabilities for each candidate triangle in this matrix. For symmetric control, we predict a raw probability matrix $\mathbf{O}$ of size $K\times K$ and symmetrize it as $\overline{\mathbf{O}} = \mathbf{O} + \mathbf{O}^\intercal$. The ground-truth labels $\mathbf{O}^*$ for the supervised training of $\overline{\mathbf{O}}$ with BCE loss are established based on the adjacent triangles to each point in the training meshes.

The prediction network. Our neural network for triangle face prediction consists of a linear projection layer followed by a 5-layer transformer [44]. Let N be the number of points in the point cloud. The inputs to our network have dimensions $N \times K \times C_{in}$, and the outputs have dimensions $N \times K \times K$. We train the network using point clouds that approximate an uniform distribution, leveraging triangle meshes from the ABC dataset. Figure 2(a) shows some examples from our training set, which consists of simple synthetic shapes. Figure 2(b) illustrates how the network is trained using the adjacent triangles and the KNN neighborhood of each point in the mesh.

Triangle Extraction. While applying the trained network to surface reconstruction, we extract triangle faces based on the predicted probabilities $\overline{\mathbf{O}}$ at each point. Specifically, we sort the columns of each row in $\overline{\mathbf{O}}$ to select the top two most likely triangle faces. Different confidence thresholds are applied to filter the selected triangles, with p_1 for the first triangle and p_2 for the second. Furthermore, we check the angle between the two triangles to avoid unwanted face folding, enforcing an angle greater than A. This strategy is applied to extract adjacent triangles for each point. Duplicate triangle predictions from different points are removed.

3.2. Point Offset Optimization

The trained network in §3.1 performs well only on surface reconstruction from point clouds with distributions similar to the training data. To extend its applicability to general point clouds, we propose optimizing an offset for each input point, which constitutes our key contribution. The intuition behind this design is to adjust the point locations to match the preferred distribution of the network.

Offset initialization. We initialize the offsets of each point based on their nearest neighbors. For example, the offset of point $\mathbf{p}$ is initialized as

$$\Delta\mathbf{p}^0 = 0.25\times(\mathbf{p} - \mathbf{q}_1), \tag{2}$$

where $\mathbf{q}_1$ is the nearest neighbor of $\mathbf{p}$. This initialization pulls each point *further apart* from its nearest neighbor. With these offsets, we re-establish the normalized KNN neighborhood $\overline{\mathcal{K}}(\mathbf{p})$ using the new points with added offsets (*e.g.*, $\mathbf{p} + \Delta\mathbf{p}$). Note that we do not search those KNN neighborhoods again but use the original indexing and ordering. Our trained network employs the re-computed $\overline{\mathcal{K}}'(\mathbf{p})$ to calculate new input features.

Offset optimization. We freeze the network and compute the gradients of the offsets by minimizing the average BCE loss for triangle predictions:

$$\mathcal{L} = \frac{1}{N\times K\times K}\sum_n\sum_i\sum_j \mathrm{BCE}(O_{nij}). \tag{3}$$

As there are no ground-truth labels in this process, we construct pseudo-labels based on the triangle extraction strategy described in §3.1. Specifically, for each predicted $\overline{\mathbf{O}}$, we encourage the top two most likely triangles along each row to have labels of 1, with the condition that the highest confidence $p_1 > 0.5$.

Let the raw gradient for $\Delta\mathbf{p}$ at the t-th iteration be $\nabla_t(\Delta\mathbf{p})$, and γ_t be the learning rate. The original distance between $\mathbf{p}$ and its nearest neighbor $\mathbf{q}_1$ is $d_0(\mathbf{p}) = \|\mathbf{p} - \mathbf{q}_1\|$. Based on the directions provided by the raw gradient, we control the offset update to prevent each point from drifting arbitrarily away from the surface, using the distance to its nearest neighbor. Our uncontrolled updates for each offset are computed as

$$\Delta\mathbf{p}^{t+1} = \Delta\mathbf{p}^t - \gamma_t \cdot \widetilde{\nabla}_t(\Delta\mathbf{p}), \tag{4}$$

in which

$$\widetilde{\nabla}_t(\Delta\mathbf{p}) = d_0(\mathbf{p}) \cdot \frac{\nabla_t(\Delta\mathbf{p})}{\|\nabla_t(\Delta\mathbf{p})\|}. \tag{5}$$

For every uncontrolled update of the offset, we evaluate how it affects the new distance between each point and its nearest neighbor, calculated as

$$d_{t+1} = \min_k \|(\mathbf{p} + \Delta\mathbf{p}^{t+1}) - (\mathbf{q}_k + \Delta\mathbf{q}_k^{t+1})\|, \tag{6}$$

To prevent collisions and promote the repulsion between nearest-neighbor points during the offset optimization, we update the offset $\Delta\mathbf{p}$ only when $d_{t+1} > \frac{d_0}{2}$. Finally, the controlled updates of each offset become

$$\Delta\mathbf{p}^{t+1} = \Delta\mathbf{p}^t - m \cdot \gamma_t \cdot \widetilde{\nabla}_t(\Delta\mathbf{p}), \tag{7}$$

with

$$m = \mathbb{I}\big[d_{t+1}(\mathbf{p}) > 0.5\times d_0(\mathbf{p})\big]. \tag{8}$$

In the experiments, we apply the offset optimization with a decaying learning rate γ_t for a specified number of iterations T to reconstruct the surface effectively.

Comparison to NKSR [20]. Our approach reconstructs surfaces by optimizing an offset for each point, making it applicable to arbitrary surfaces, even though the network is trained solely on the ABC dataset. In contrast, NKSR requires training on a diverse set of shapes and surfaces for effective generalization and relies on high-quality, well-oriented normals for optimal performance. Although normals are not used, OffsetOPT outperforms NKSR in both reconstruction accuracy and the preservation of fine details.

3.3. Reconstruction Application

Although our method handles general point cloud inputs, its primary target is surface reconstruction from dense point clouds. In that case, we begin by voxelizing[1] the dense point cloud to create a more regular input, then apply the trained network in §3.1 with offset optimization in §3.2 to reconstruct the surface. A recommended setting for the voxel size v is to approximate the largest nearest-neighbor distance between point pairs, which is the coarsest resolution

[1] voxel-based point cloud subsampling.

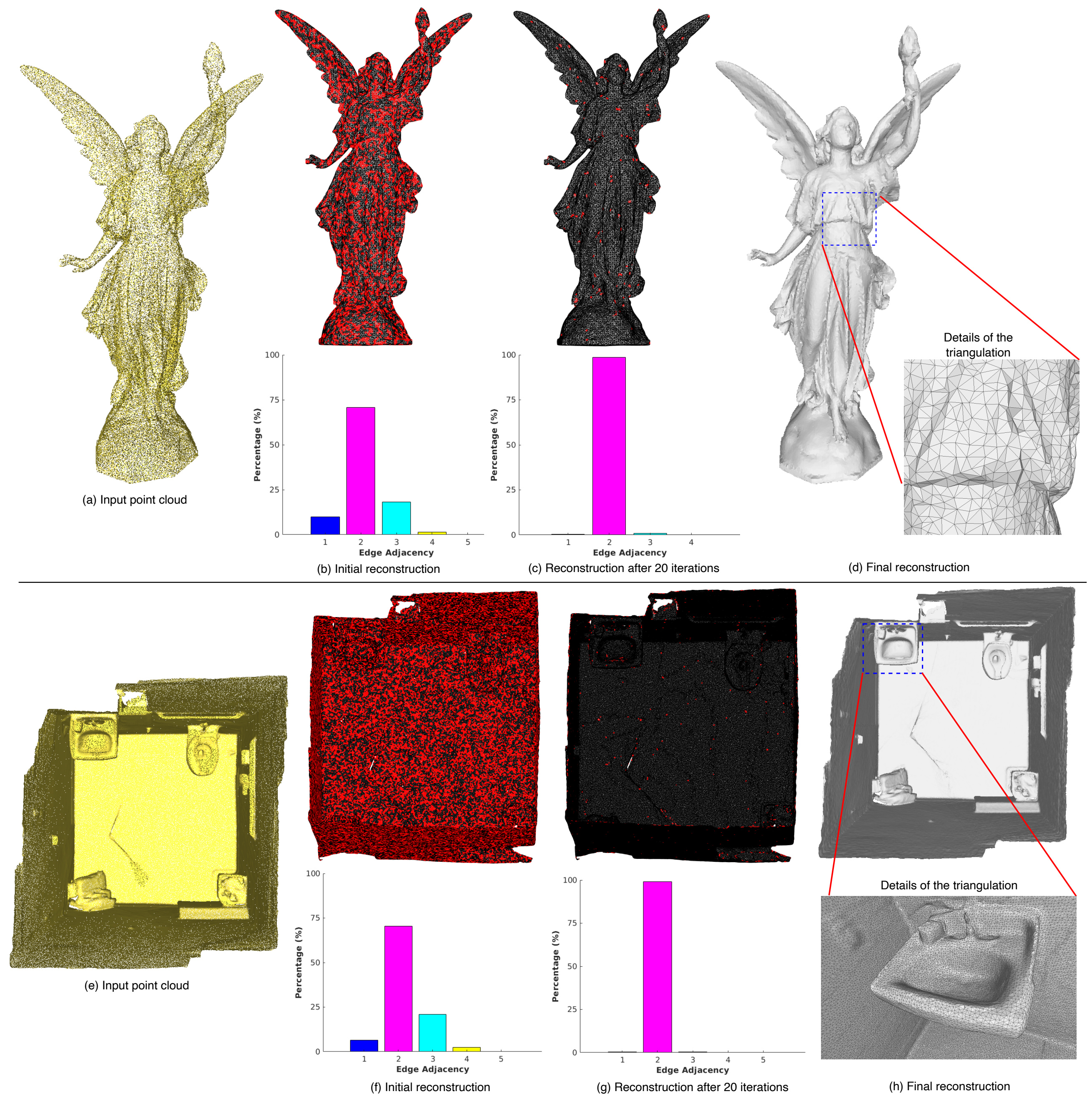
(a) Input point cloud
Percentage (%)
Edge Adjacency
(b) Initial reconstruction
(c) Reconstruction after 20 iterations
Details of the triangulation
(d) Final reconstruction
(e) Input point cloud
(f) Initial reconstruction
(g) Reconstruction after 20 iterations
Details of the triangulation
(h) Final reconstruction

Figure 3. Evolution of manifold edges during OffsetOPT reconstruction. In the top row, we illustrate the reconstruction process for a shape from Thingi10k, with manifold edges in *black* and non-manifold edges in *red*. Given the input point cloud in (a), the initial reconstruction contains a high percentage of non-manifold edges, as shown in the histogram in (b), where 'Edge Adjacency' refers to the number of faces adjacent to each edge. We note that manifold edges have an adjacency of no more than 2. After 20 iterations of offset optimization, the percentage of manifold edges increases significantly from 75% to 99% in (c), while the number of non-manifold edges diminishes. The final reconstructed mesh, with detailed triangulation around the belly, is shown in (d). The bottom row shows a similar effect in scene reconstruction from ScanNet. Comparing (f) and (g), iterative offset optimization substantially increases the number of manifold edges. In (h), we display the final reconstruction with detailed triangulation around a sink.

of the point cloud. We briefly summarize the proposed OffsetOPT method for surface reconstruction in Algorithm 1.

For a visual understanding of how OffsetOPT promotes manifold edges during surface reconstruction, we present two examples to demonstrate the process: one for a shape and another for a scene, as illustrated in Fig. 3. In both cases, the number of manifold edges increases significantly through offset optimization, resulting in high-quality trian-

Algorithm 1 Offset optimization for surface reconstruction.

```
Input: (1) A point cloud $\{\mathbf{x}_s\}_{s=1}^S$; (2) The trained network.
Output: The reconstructed surface as a triangle mesh.
1: Voxelize the point cloud as $\{\mathbf{p}_n\}_{n=1}^N$ with grid size $v$.
2: Establish the $K$NN neighborhood $\mathcal{K}(\mathbf{p})$ of each point.
3: Initialize the corresponding offsets $\{\Delta\mathbf{p}_n^0\}_{n=1}^N$.
4: for iteration $t < T$ do
5:     Re-compute the normalized neighborhood $\overline{\mathcal{K}}'(\mathbf{p})$.
6:     Predict triangles with the network using $\overline{\mathcal{K}}'(\mathbf{p})$.
7:     Compute loss $\mathcal{L}$ in Eq. (3) and backpropagate to get
       raw offset gradients $\nabla_t(\Delta\mathbf{p})$.
8:     Compute the uncontrolled $\Delta\mathbf{p}^{t+1}$ using Eq (4).
9:     Check the updated NN distance $d_{t+1}$ with Eq. (6).
10:    Update the offset with the controlled Eq. (7).
11: end for
12: Extract surface triangles using the strategy in §3.1.
13: [Optional] Post-process for strict edge-manifoldness.
14: return The reconstructed triangle mesh.
```

gulation details in the final reconstructed surface.

Despite achieving notable performance in yielding edge-manifold triangles, our method may still reconstruct the surface with a small percentage of non-manifold edges. These can be removed through post-processing using functions from [25] or 3D libraries like Open3D [52]. Given the low percentage of non-manifold edges, such post-processing is efficient and has minimal impact on the reconstruction quality. We report reconstruction accuracy of our method on the non-post-processed surfaces in the experiments.

4. Experiments

Implementation details. The transformer layers in our neural network has 64 channels and 4 attention heads. We use K=50 in the KNN search, and a basis level of 8 for the positional encoding. During training, we randomly sample points from each training mesh, using their adjacent triangles for ground-truth labels and KNN geometry as network inputs. The input point clouds are augmented with random rotations, scaling, and jittering. For offset optimization, the learning rate is initialized to $\gamma_0 = 0.1$ and decays by a factor of 0.7 every 10 iterations. We optimize the offsets for 100 iterations. The threshold settings for triangle extraction are $(p_1, p_2, A) = (0.8, 0.5, 120°)$.

Evaluation criteria. We assess the overall surface quality of each reconstructed mesh using symmetric Chamfer distances (CD1, CD2), F-Score (F1), normal consistency (NC), and normal reconstruction error (NR) in degrees. The quality of fine surface details is further evaluated using Edge Chamfer Distance (ECD1) and Edge F-score (EF1), following [11, 25]. Specifically, for shape reconstruction, we sample 10^5 points from both the ground-truth and reconstructed meshes, while for scene reconstruction, we use 10^6 points to ensure adequate surface coverage. All meshes are re-scaled to fit within a unit sphere for metric reporting.

Training data. The ABC dataset [24] offers a collection of clean synthetic meshes with high-quality, nearly equilateral triangle faces. We use the 9,026 voxelized[2] meshes from [25], split into 25% for training and 75% for testing, to train and evaluate our triangle prediction network.

Method comparison. We compare the reconstruction quality of OffsetOPT to computational reconstruction methods, including ball-pivoting ([3]), PointTriNet [41], DSE [39], and CircNet [25], as well as representative implicit neural methods, including SPSR [22] and NKSR [20].

4.1. Shape Reconstruction

Trained on the ABC training set, our prediction network outperforms existing methods on the ABC test set with a simple forward pass, without offset optimization, due to the similarity in point distributions between the training and test sets. The results are reported in Table 1.

For surface reconstruction from unseen point clouds in general, we apply the model trained on ABC with offset optimization, *i.e.*, the proposed OffsetOPT. We validate the generalization of OffsetOPT using FAUST [5], a dataset of watertight meshes of human bodies, and MGN [4], a dataset of open meshes of clothes. The results are reported in Table 2 and Table 3, respectively.

It can be seen that OffsetOPT consistently outperforms other approaches, especially in reconstructing sharp surface details. We note that the performance drop of NKSR on the MGN dataset is due to its tendency to close small holes in the open surfaces, such as those around the neck and legs.

For all shape reconstruction experiments, the default input points consist of the vertices of the test meshes. We additionally sample 0.1 millon points from each test mesh to report the improved reconstruction results of NKSR. We compute estimated normals $\hat{\mathbf{n}}$ for ball-pivoting from the input point cloud and the ground-truth normals $\mathbf{n}$ for SPSR and NKSR from the ground-truth mesh.

4.2. Scene Reconstruction

To demonstrate the performance of OffsetOPT in large-scale scene reconstruction, we apply it to surfaces from the test splits of ScanNet [14], Matterport3D [8], and 10 random scenes from CARLA [15, 20]. In all experiments, we randomly sample one million points from the ground-truth meshes (ScanNet and Matterport3D) or dense point clouds (CARLA) as inputs. In the reconstruction with OffsetOPT, we voxelize the input points with a grid size of 2 cm for ScanNet and 10 cm for Matterport3D and CARLA. We compare OffsetOPT to SPSR and the state-of-the-art NKSR, with quantitative results provided in Table 4 and visualized comparisons in Fig. 4. It is observed that

[2]Mesh decimation with voxel-based vertex clustering.

Table 1. Method comparison on the **ABC** test set. '+$\hat{\mathbf{n}}$, +$\mathbf{n}$' indicate the usage of estimated and ground-truth normals, respectively; '0.1M' represents reconstruction from 0.1 million points randomly sampled from the ground-truth mesh.

Method	Surface Quality						
	overall					sharp	
	CD1($\times 10^2$)↓	CD2($\times 10^5$)↓	F1↑	NC↑	NR↓	ECD1($\times 10^2$)↓	EF1↑
ball-pivot (+$\hat{\mathbf{n}}$)	0.297	0.684	0.939	0.981	2.244	0.782	0.873
SPSR (+$\mathbf{n}$)	0.400	6.081	0.901	0.972	6.020	26.160	0.108
DSE	0.285	0.548	0.949	0.985	1.793	0.538	0.929
PointTriNet	0.288	0.790	0.948	0.984	1.931	0.688	0.926
CircNet	0.284	0.544	0.950	0.985	1.758	0.708	0.924
NKSR (+$\mathbf{n}$)	0.370	3.968	0.918	0.978	5.225	27.499	0.097
NKSR (+$\mathbf{n}$, 0.1M)	0.306	1.167	0.938	**0.989**	2.929	4.152	0.514
OffsetOPT (**Prop.**)	**0.283**	**0.540**	**0.951**	0.988	**1.318**	**0.402**	**0.941**

Table 2. Method comparison on the **FAUST** dataset. Each point cloud contains 6,890 points by default.

Method	Surface Quality						
	overall					sharp	
	CD1($\times 10^2$)↓	CD2($\times 10^5$)↓	F1↑	NC↑	NR↓	ECD1($\times 10^2$)↓	EF1↑
ball-pivot (+$\hat{\mathbf{n}}$)	0.323	1.002	0.923	0.970	6.037	2.887	0.184
SPSR (+$\mathbf{n}$)	0.427	4.108	0.915	0.969	10.269	1.069	0.810
DSE	0.218	0.307	0.995	0.984	**3.910**	0.883	0.801
PointTriNet	0.219	0.308	0.995	0.983	4.393	1.233	0.807
CircNet	0.221	0.316	0.993	0.980	4.557	0.939	0.820
NKSR (+$\mathbf{n}$)	0.302	0.654	0.972	0.973	9.410	2.737	0.501
NKSR (+$\mathbf{n}$, 0.1M)	0.227	0.319	**0.997**	**0.987**	6.303	0.970	0.813
OffsetOPT (**Prop.**)	**0.217**	**0.301**	0.996	0.985	4.038	**0.561**	**0.896**

Table 3. Method comparison on the **MGN** open surfaces.

Method	Surface Quality						
	overall					sharp	
	CD1($\times 10^2$)↓	CD2($\times 10^5$)↓	F1↑	NC↑	NR↓	ECD1($\times 10^2$)↓	EF1↑
ball-pivot (+$\hat{\mathbf{n}}$)	0.462	4.917	0.844	0.974	5.803	11.847	0.083
SPSR (+$\mathbf{n}$)	1.077	10.481	0.402	0.948	12.224	7.912	0.137
DSE	0.270	0.530	0.968	0.983	3.970	4.508	0.440
PointTriNet	0.272	0.562	0.967	0.981	4.398	5.936	0.399
CircNet	**0.269**	0.512	**0.968**	0.981	4.230	**3.231**	0.486
NKSR (+$\mathbf{n}$)	0.946	23.263	0.611	0.947	11.818	13.815	0.065
NKSR (+$\mathbf{n}$, 0.1M)	0.381	0.884	0.891	0.990	4.997	6.488	0.441
OffsetOPT (**Prop.**)	0.278	**0.511**	0.964	**0.991**	**2.967**	5.447	**0.538**

SPSR recovers the overall scene structure well but suffers from strong over-smoothing, limiting its reconstruction accuracy. In contrast, our method-without relying on normals-consistently outperforms NKSR, particularly in recovering the fine details, whereas ground-truth normals are provided for the latter. For space concern, we show additional visual comparisons in the supplementary material.

4.3. Ablation Study

Impact of offset optimization. Table 5 presents the reconstruction accuracy (CD1) and the percentage of manifold edges (%) using our trained model with (*w*) and without (*w/o*) the offset optimization in stage two. The results clearly indicate that optimizing offsets significantly promotes manifold edges, as shown in the histograms of Fig. 3.

Different offset initialization. In Eq. (2), we slightly move each point away from its nearest neighbor for offset initialization. A more straightforward approach is to initialize all offsets as zeros. Figure 5 illustrates the percentage of manifold edges in reconstructed surfaces using different initializations for the FAUST (§4.1) and Thingi10k [51] datasets. This experiment uses 100 shapes from Thingi10k, with point clouds consisting of 0.1 million points randomly sampled from the ground-truth meshes. See Table 7 in the supplementary for the reconstruction accuracies using the two initializations. We adopt the proposed initialization as it yields slightly better performance.

5. Limitation

Although the proposed OffsetOPT demonstrates remarkable improvements in producing manifold edges compared to prior computational reconstruction methods, a small percentage of non-manifold edges may still occur. Future work will aim to address this limitation more comprehen-

Table 4. Method comparison on *large-scale* open surface datasets: **ScanNet**, **Matterport3D**, and **CARLA**.

	Method	Surface Quality						
		overall					sharp	
		CD1($\times 10^2$)↓	CD2($\times 10^5$)↓	F1↑	NC↑	NR↓	ECD1($\times 10^2$)↓	EF1↑
ScanNet	SPSR (+**n**)	5.428	352.674	0.194	0.709	35.904	7.370	0.131
	NKSR (+**n**)	0.157	0.164	0.997	**0.963**	9.824	0.618	0.885
	NKSR	0.423	1.648	0.793	0.901	17.699	1.743	0.586
	OffsetOPT (**Prop.**)	**0.147**	**0.136**	**1.0**	0.960	**9.533**	**0.389**	**0.931**
MPort3D	SPSR (+**n**)	0.926	28.893	0.724	0.830	23.322	2.202	0.344
	NKSR (+**n**)	0.183	0.220	0.995	0.936	12.713	0.619	0.842
	NKSR	0.271	0.762	0.939	0.894	18.076	0.903	0.718
	OffsetOPT (**Prop.**)	**0.148**	**0.139**	**1.0**	**0.938**	**10.665**	**0.250**	**0.973**
CARLA	SPSR (+**n**)	4.407	234.338	0.121	0.733	30.835	6.799	0.062
	NKSR (+**n**)	0.175	0.299	0.974	0.953	7.740	0.679	0.903
	NKSR	0.238	2.682	0.968	0.936	10.761	1.013	0.810
	OffsetOPT (**Prop.**)	**0.124**	**0.272**	**0.987**	**0.963**	**5.530**	**0.317**	**0.946**

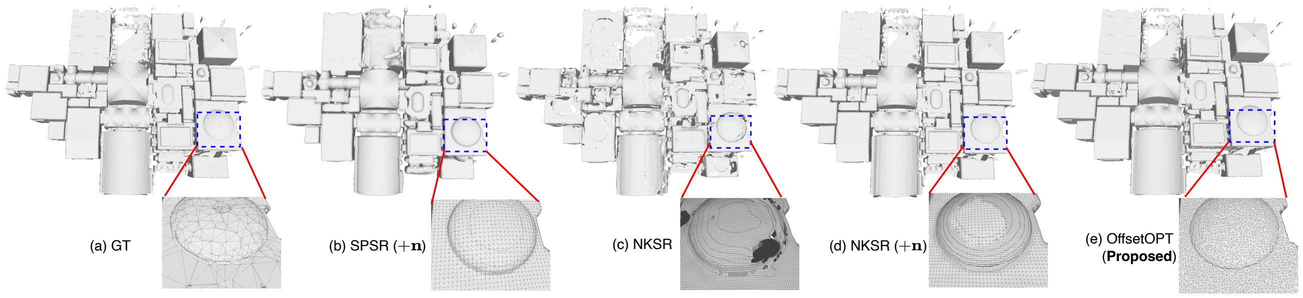

Figure 4. Comparison of different reconstruction methods on a large-scale building from Matterport3D [8]. (a) The ground-truth surface. (b) SPSR [22] reconstruction using ground-truth normals, showing strong oversmoothing. (c) NKSR [20] reconstruction without normals, resulting in many disconnected components. (d) The improved NKSR reconstruction with ground-truth normals. (e) Our reconstruction with OffsetOPT, which captures fine details without requiring normals. Zoomed-in triangulation details are shown for each surface.

Table 5. Impact of offset optimization.

offset	ABC	FAUST	MGN	ScanNet	MP3D
w/o	**0.283 (99%)**	0.221 (82%)	0.280 (88%)	0.154 (74%)	0.164 (78%)
w	-	**0.217 (98%)**	**0.278 (99%)**	**0.147 (99%)**	**0.148 (99%)**

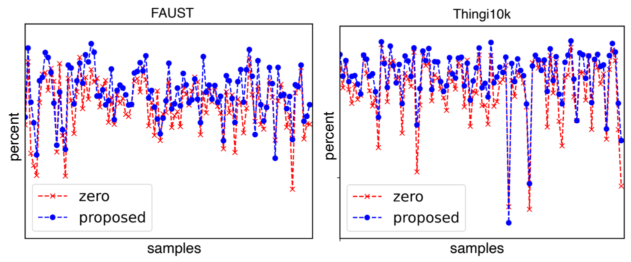

Figure 5. The percentage of manifold edges in the reconstructed surfaces using zero and proposed initializations, with *red* for zero and *blue* for proposed. Results are shown for the FAUST and Thingi10k datasets. The x-axis in our plots indicates the number of samples, while the y-axis represents the percentage.

sively. Additionally, our prediction network utilizes a transformer architecture, which, while achieving superior reconstruction accuracy, is less efficient than the graph convolutional networks [25]. Exploring the combination of graph convolutions with transformers in future work could better balance accuracy and efficiency in surface reconstruction, while potentially enhancing method robustness. Currently, our approach takes an average of 15.67 seconds and 13.83 seconds to reconstruct each shape in FAUST and MGN, respectively, and approximately 6, 12, and 9 minutes to reconstruct each scene in ScanNet, Matterport3D, and CARLA, using a single NVIDIA GeForce RTX 4090 GPU.

6. Conclusion

We have introduced a novel neural computational method, OffsetOPT, for reconstructing explicit surfaces directly from 3D point clouds. Unlike the implicit methods, it does not require point normals. OffsetOPT operates in two stages: it first trains a triangle prediction network based on local point geometry to reconstruct surfaces from ideally distributed points, and then applies the trained model to optimize per-point offsets for accurate surface reconstruction from general point clouds. Our approach outperforms existing methods in both overall reconstruction accuracy and the preservation of sharp surface features. We validate its effectiveness across a diverse range of benchmark datasets, including both small-scale shapes and large-scale indoor and outdoor surfaces, demonstrating its ability to achieve high-quality surface reconstructions without relying on normals.

7. Acknowledgments

This research was supported by the Centre for Augmented Reasoning (CAR) at the Australian Institute for Machine Learning (AIML), University of Adelaide.

References

[1] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 2

[2] Ma Baorui, Han Zhizhong, Liu Yu-Shen, and Zwicker Matthias. Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. In *International Conference on Machine Learning (ICML)*, 2021. 2

[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. 1, 2, 6

[4] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *proceedings of the IEEE/CVF international conference on computer vision*, pages 5420–5430, 2019. 2, 6

[5] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3794–3801, 2014. 2, 6

[6] Alexandre Boulch and Renaud Marlet. Poco: Point convolution for surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6302–6314, 2022. 2

[7] Frédéric Cazals and Joachim Giesen. *Delaunay triangulation based surface reconstruction: ideas and algorithms*. PhD thesis, INRIA, 2004. 2

[8] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. 2, 6, 8

[9] 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

[10] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5939–5948, 2019. 2

[11] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *arXiv preprint arXiv:2202.01999*, 2022. 6

[12] Siu-Wing Cheng, Tamal Krishna Dey, Jonathan Shewchuk, and Sartaj Sahni. *Delaunay mesh generation*. CRC Press Boca Raton, 2013. 1, 2

[13] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33: 21638–21652, 2020. 1, 2

[14] 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. 2, 6

[15] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 2, 6

[16] Herbert Edelsbrunner and Ernst P Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1):43–72, 1994. 2

[17] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020. 2

[18] Benoit Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *European Conference on Computer Vision*, pages 576–592. Springer, 2022. 1, 2

[19] Jiahui Huang, Hao-Xiang Chen, and Shi-Min Hu. A neural galerkin solver for accurate surface reconstruction. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022. 2

[20] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4369–4379, 2023. 1, 2, 3, 4, 6, 8

[21] 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

[22] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 1, 2, 6, 8

[23] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 1, 2, 3

[24] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big CAD model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019. 2, 6

[25] Huan Lei, Ruitao Leng, Liang Zheng, and Hongdong Li. Circnet: Meshing 3d point clouds with circumcenter detection. *International Conference on Learning Representations*, 2023. 1, 2, 3, 6, 8

[26] Minghua Liu, Xiaoshuai Zhang, and Hao Su. Meshing point clouds with predicted intrinsic-extrinsic ratio guidance. In *European Conference on Computer Vision*, pages 68–84. Springer, 2020. 1, 2

[27] 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. 1, 2

[28] Yujie Lu, Long Wan, Nayu Ding, Yulong Wang, Shuhan Shen, Shen Cai, and Lin Gao. Unsigned orthogonal distance fields: An accurate neural implicit representation for diverse 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20551–20560, 2024. 2

[29] de Berg Mark, Cheong Otfried, van Kreveld Marc, and Overmars Mark. *Computational geometry algorithms and applications*. Spinger, 2008. 2

[30] 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. 1, 2

[31] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*. Springer, 2020. 3

[32] Amine Ouasfi and Adnane Boukhayma. Unsupervised occupancy learning from sparse point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21729–21739, 2024. 2

[33] 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, 2

[34] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. 2

[35] 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. 2

[36] Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012. 2

[37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 3

[38] Marie-Julie Rakotosaona, Noam Aigerman, Niloy J Mitra, Maks Ovsjanikov, and Paul Guerrero. Differentiable surface triangulation. *ACM Transactions on Graphics (TOG)*, 40(6): 1–13, 2021. 2

[39] 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. 1, 2, 3, 6

[40] Scott Schaefer and Joe Warren. Dual marching cubes: Primal contouring of dual grids. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pages 70–76. IEEE, 2004. 1

[41] Nicholas Sharp and Maks Ovsjanikov. Pointtrinet: Learned triangulation of 3d point sets. In *European Conference on Computer Vision*, pages 762–778. Springer, 2020. 1, 2, 6

[42] 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. 1, 2

[43] Federico Stella, Nicolas Talabot, Hieu Le, and Pascal Fua. Neural surface detection for unsigned distance fields. In *European Conference on Computer Vision*, pages 394–409. Springer, 2024. 1, 2

[44] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 2, 3

[45] Li Wang, Weikai Chen, Xiaoxu Meng, Bo Yang, Jintao Li, Lin Gao, et al. Hsdf: Hybrid sign and distance field for modeling surfaces with arbitrary topologies. *Advances in Neural Information Processing Systems*, 35:32172–32185, 2022. 2

[46] Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (TOG)*, 41(4): 1–15, 2022. 2

[47] Francis Williams, Matthew Trager, Joan Bruna, and Denis Zorin. Neural splines: Fitting 3d surfaces with infinitely-wide neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9949–9958, 2021. 2

[48] Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. Neural fields as learnable kernels for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18500–18510, 2022. 2

[49] Cem Yuksel. Sample elimination for generating poisson disk sample sets. In *Computer Graphics Forum*, pages 25–32. Wiley Online Library, 2015. 1, 3

[50] Congyi Zhang, Guying Lin, Lei Yang, Xin Li, Taku Komura, Scott Schaefer, John Keyser, and Wenping Wang. Surface extraction from neural unsigned distance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22531–22540, 2023. 2

[51] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 7, 1

[52] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 6