Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance

Minghua Liu, Xiaoshuai Zhang, and Hao Su

University of California, San Diego
{minghua,zxs,haosu}@ucsd.edu

Abstract. We are interested in reconstructing the mesh representation of object surfaces from point clouds. Surface reconstruction is a prerequisite for downstream applications such as rendering, collision avoidance for planning, animation, etc. However, the task is challenging if the input point cloud has a low resolution, which is common in real-world scenarios (e.g., from LiDAR or Kinect sensors). Existing learning-based mesh generative methods mostly predict the surface by first building a shape embedding that is at the whole object level, a design that causes issues in generating fine-grained details and generalizing to unseen categories. Instead, we propose to leverage the input point cloud as much as possible, by only adding connectivity information to existing points. Particularly, we predict which triplets of points should form faces. Our key innovation is a surrogate of local connectivity, calculated by comparing the intrinsic/extrinsic metrics. We learn to predict this surrogate using a deep point cloud network and then feed it to an efficient post-processing module for high-quality mesh generation. We demonstrate that our method can not only preserve details, handle ambiguous structures, but also possess strong generalizability to unseen categories by experiments on synthetic and real data. The code is available at https://github.com/Colin97/Point2Mesh.

Keywords: mesh reconstruction, point cloud

1 Introduction

Among various 3D representations (e.g., polygonal meshes, voxels, point clouds, multi-view 2D images, part-based primitives, and implicit field functions), polygonal meshes capture the geometric details of the shape in an efficient way, which prevents high memory footprints and artifacts caused by discretization. Reconstructing high-quality 3D meshes from point clouds thus has been studied for quite a long time and serves as a prerequisite for numerous real-world applications, including autonomous driving, augmented reality, and robotics.

Despite its long history, the mesh reconstruction problem remains unresolved. Traditional methods [24,2,30] typically reconstruct the mesh either by explicitly connecting the points or implicitly approximating the surface, both of which resort to local geometric hints. Without reasoning about the shape, traditional

methods may be hard to handle the ambiguous structures when the resolution of the input point cloud is limited. For example, the ambiguous structures may include thin structures consisting of two very close surfaces, independent but spatially adjacent parts, and corners. Traditional methods tend to produce distortion or connect independent parts incorrectly when facing these structures. However, the reconstruction of these fine-grained structures may be essential for many downstream applications such as robotics grasping which needs an accurate understanding of part-level mobility. Moreover, traditional methods are typically sensitive to hyper-parameters. For most of these methods, a dedicated parameter-tuning is required for each input, making batch processing of point clouds impractical.

With the rapid development of 3D deep learning and the availability of large-scale 3D datasets, people tend to learn geometric or semantic priors from data. Unlike 2D images and 3D voxels, polygon meshes is an irregular geometric representation, which prevents it from being generated by the neural network directly. However, there are still lots of attempts to explore the neural-network-compatible representations for mesh generation, including template meshes with deformation [43,18,14,35,29,23,16,44], 2D squares with folding [46,19,10], primitives with assembly [6,41,39], implicit field function [36,7,15,32], and meshlets with optimization [1,45]. Existing learning-based methods typically follow the "encoder-decoder" paradigm. The limited capability of the network prevents existing methods from generating fine-grained structures and details. Also, since most existing methods learn the priors at the object level, they tend to memorize the overall shapes and typically cannot generalize to unseen categories.

To this end, we propose a novel method that reconstructs meshes from point clouds by leveraging the intermediate representation of triangle faces. Unlike existing methods, our method fully utilizes the input point clouds, which are on the ground truth surface in most cases, and then estimate the local connectivity with the help of learned guidance. More specifically, we first propose a set of candidate triangle faces, which could be the elements of the reconstructed mesh, by constructing a k-nearest neighbor (k-NN) graph on the input point cloud. We then utilize the neural network to filter out the incorrect candidates and provide cues for sorting the remaining candidates. We find that the ratio of geodesic distance (intrinsic metric) and Euclidean distance (extrinsic metric) between two vertices may provide strong cues for inferring the connectivity and can naturally serve as the supervision for the candidate classification task. Since there are multiple ways to triangulate a surface, we only filter out those candidates that should never appear in the reconstructed mesh, such as the candidates linking two independent parts. A greedy post-processing algorithm is then used to sort all the remaining candidates and merge them into the final mesh.

We demonstrate that our algorithm can preserve fine-grained details and handle ambiguous structures with the help of learned intrinsic-extrinsic guidance. Since our method reconstructs meshes by estimating local connectivity, which relies mainly on the local geometric information, it can well generalize to unseen categories. In experiments on the ShapeNet dataset, our method outper-

forms both the existing traditional methods and learning-based mesh generative methods with regard to all commonly used metrics, including the F-score, Chamfer distance, and normal consistency. We also provide extensive ablation studies on different sampling densities, sampling strategies, noisy levels, and real scans to demonstrate our generalizability and robustness.

2 Related Work

3D mesh reconstruction is a core problem for many applications. Yet despite its long history, the problem is still far from being solved. In this section, we review the existing methods and the remaining difficulties of the problem.

2.1 Traditional Mesh Reconstruction

Traditional mesh reconstruction methods mainly include two paradigms: explicit reconstruction and implicit reconstruction.

Explicit reconstruction methods, such as ball-pivoting algorithm (BPA) [2], Delaunay triangulation [3], alpha shapes [12], and zippering [42], resort to the local surface connectivity estimation and connect the sampled points directly by triangles. For example, the principle of BPA is simple: three points form a triangle if a ball of a user-specified radius touches them without containing any other point. However, the radius of BPA matters a lot: a small radius can lead to holes while a large radius may cause incorrect connections. Although there are some following works trying to utilize multiple radii [11], they still fail to handle ambiguous structures well.

Implicit reconstruction methods [24,25,20,34,4,22] try to find a field function (e.g., signed distance function) approximating the point cloud and then employ the marching cube algorithm [30] to extract the iso-surface of the field function. For example, Poisson surface reconstruction (PSR) [24,25] reconstructs the surface by solving a Poisson problem for the oriented points. However, solving large-scale equations is time-consuming. Also, it is difficult for traditional algorithms to determine the consistent direction of the normals based only on the coordinates of the point cloud. Without correct vertex normal directions, PSR tends to generate poor results. Moreover, implicit reconstruction methods utilize marching cube [30] to generate the mesh, which may lead to expensive voxelization and the artifacts caused by the discretization.

Under the limited resolution, ambiguities of the input point cloud require the integration of strong geometric or semantic priors about our 3D world. With such priors and reasoning, our learning-based methods are expected to handle those ambiguous structures and avoid results with distortion and artifacts. In addition, traditional algorithms heavily rely on selecting a set of proper hyper-parameters, which may require a case-by-case parameter tuning, while our learning-based algorithm should be applied to all cases adaptively and thus enable automatic batch processing.

2.2 Learning-based mesh generation

The recent success of 3D deep learning [37,38] and the availability of large 3D datasets [5,33] nourish the tasks of 3D analysis and 3D synthesis. However, unlike 2D images, 3D polygon meshes are irregular geometric formats and are difficult to be directly generated from the neural networks. Existing learning-based mesh generative methods mainly follow five paradigms: deformation-based methods [43,18,14,35,29,23,16,44], folding-based methods [46,19,10], primitive-based methods [6,41,39], optimization-based methods [1,45], and implicit-field-function-based methods [36,7,15,32,26].

Deformation-based methods resort to deform a template mesh (e.g., a sphere mesh) into the desired shape. However, since they only deform the position of the vertices without changing the connectivity, the topology of the template mesh may restrict the methods from generating shapes of a specific topology. Folding-based methods learn a set of mappings from 2D squares to 3D patches, which are then used to form the mesh. Primitive-based methods utilize a set of primitives (e.g., planes and convex patches) to form the final mesh, and learn the parameters of the primitives. The simplicity of the primitives may prevent the methods from generating fine-grained details. As for the implicit-field-function-based methods, they employ neural networks to learn an implicit field function and then utilize marching cube algorithms [30] to extract the iso-surface, and thus face similar problems as the traditional implicit reconstruction methods. There are also some recent optimization-based methods, which either utilize a deep neural network as local geometric prior [45] or learn some local shape priors from the data [1]. They formulate mesh reconstruction as an optimization problem and are thus computational expensive.

Most existing learning-based methods do not make full use of the input point cloud that processing should be grounded upon. Although they may be able to generate the coarse-grained shapes, they may fail to capture some of the structures and details. Also, most existing methods learn the priors at the object level, which makes them category-specific, and even sensitive to the pose of the object. In contrast, our method will be fully based on the grounded point clouds to preserve all the structures and generate fine-grained details. Moreover, our method relies on local priors and can thus generalize to unseen categories.

3 Method

Given a 3D point cloud $P = \{(x_i, y_i, z_i)\}$, we aim to reconstruct a polygon mesh, which consists of a vertex set and a face set, approximating the underlying surface. Unlike existing learning-based mesh generative methods, we fully utilize the grounded input point cloud and let P serve as the vertex set of the reconstructed mesh, since they are usually on or near the surface of the shape and provide lots of cues for the structures and details. We then reconstruct the mesh by predicting the local connectivity between the vertices. Before presenting our reconstruction method, we would like to introduce a motivating remeshing algo-

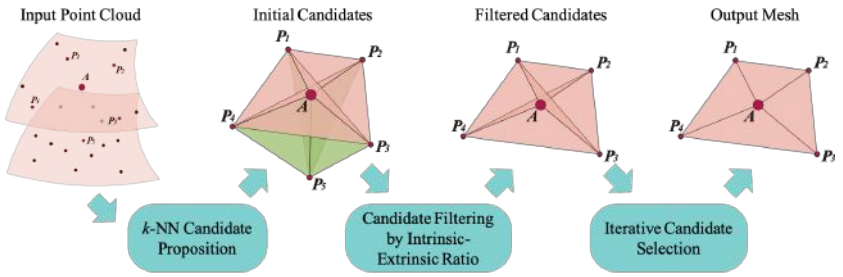

Fig. 1: Our remeshing pipeline from a local view. The candidate triangles near the underlying surfaces are colored in pink, while others are colored in green.

rithm, where the ground truth mesh is known. We then extend the remeshing algorithm to mesh reconstruction by introducing a neural network module.

3.1 A Motivating Remeshing Algorithm

Given a reference triangle mesh M_R and a point cloud P sampled on M_R as input, the remeshing algorithm aims to generate a new mesh M_N whose vertices come from the point cloud P. As shown in Fig. 1, the algorithm first proposes a set of candidate triangle faces and then uses a subset of candidate triangles to form the mesh M_N.

Candidate Proposition Since each vertex should only connect to its neighbors on the surface, the algorithm first constructs a k-nearest neighbor (k-NN) graph for each of the points in P based on the Euclidean distance, and then each vertex form candidate triangle faces with each two of its k-NN neighbors. We expect the union of candidate triangles to cover the whole surface of M_R, which means for an even surface with uniformly distributed vertices, a small k would be enough, while for complex surfaces and nonuniformly distributed vertices, we may need a larger k. However, we could select a k that is large enough to cover most practical cases. Among all the candidates, some of them are on or near the surface of mesh M_R, and we denote them as *correct candidates*, while the others are away from the surface and are denoted as *incorrect candidates*. The incorrect candidates may appear in areas such as (a) thin structures consisting of two very close surfaces, (b) independent but spatially adjacent parts, and (c) surfaces with large curvature. We would like to filter out the incorrect candidates, and use some of the remaining candidate faces to form the mesh M_N.

Candidate Filtering Since the reference mesh M_R is given, we can calculate the geodesic distance between two vertices, which is defined to be the length of the shortest path over the surface of M_R. As the intrinsic metric of the surface manifold, geodesic distance provides strong cues for inferring the connectivity

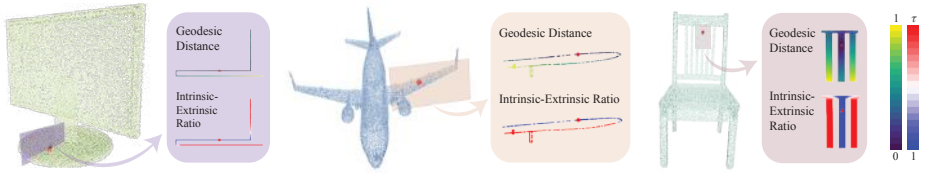

Fig. 2: In each example, we sample a slice from the input point cloud, and demonstrate the geodesic distance and the intrinsic-extrinsic ratio (IER) to the key point (marked in red).

between two vertices. The geodesic distance could be inconsistent with the Euclidean distance (extrinsic metric). However, in a small neighborhood, if the geodesic distance between two vertices is equal or close to their Euclidean distance, the connecting line between them is likely to be on or close to the surface of mesh M_R. We thus define the intrinsic-extrinsic ratio (IER) for a pair of vertices u, v as:

$$\mathrm{IER}(u, v) = \frac{d_G(u, v)}{d_E(u, v)}, \tag{1}$$

where d_G and d_E indicate the geodesic distance and the Euclidean distance respectively. Fig. 2 shows some examples of how geodesic distance and the IER to a key point change within a slice. The key point should not be connected with the points in the red region where IER is much higher than 1. The cases in Fig. 2 demonstrate that IER can effectively handle corners (see display), thin structures (see jet), and adjacent parts (see chair). We thus propose to employ the IER to filter out the incorrect triangle candidates. For a triangle face with vertices u, v, and w, the IER is extended to be:

$$\mathrm{IER}(u, v, w) = \frac{d_G(u, v) + d_G(u, w) + d_G(v, w)}{d_E(u, v) + d_E(u, w) + d_E(v, w)}. \tag{2}$$

With the definition above, we filter out candidates whose IER is greater than τ, and $\tau > 1$ is a preset threshold. After filtering out the incorrect candidates, the remaining candidates should be on or near the surface of M_R.

Sort and Merge Since there is no canonical way to triangulate a surface, we sort the remaining candidate triangles and merge them into mesh M_N in a greedy way. Specifically, we prefer triangles that are closer to the surface of M_R. Also, we prefer candidate triangles with three short edges, which typically correspond to the equilateral triangles when the input point cloud is uniformly distributed. We thus sort the remaining candidates with respect to their distance to M_R and the length of their longest edge. Specifically, we first divide the remaining candidate triangles into l bins based on the distance to M_R, and then sort them in each bin according to the length of their longest edge. After sorting the triangles, we visit each candidate one by one and add the candidate into the mesh M_N if it satisfies

Fig. 3: In each pair, the left one is the original reference mesh (M_R), and the right one is the result of our remeshing algorithm (M_N).

two constraints. Specifically, the new candidate face should not intersect with the previously added faces. Also, M_N should not contain non-manifold edges after adding the new candidate. That is, M_N should not contain an edge that has more than two incident faces. If both constraints hold, the candidate will be added into M_N; otherwise, it will be discarded. After visiting all the candidates, we get the final mesh M_N.

Please refer to our supplementary materials for the pseudo code of the remeshing algorithm. Some remeshing results of the algorithm are shown in Fig. 3.

3.2 From Remeshing to Reconstruction

In the mesh reconstruction setting, we are only given a point cloud P as input and aim to reconstruct the mesh M_N. As shown in Fig. 4, we follow the remeshing algorithm to construct a k-NN graph on P and propose m candidate triangle faces. Unlike in the remeshing algorithm, the reference mesh is not available, we thus cannot directly calculate the intrinsic-extrinsic ratio to serve as the guidance. At this point, the neural network may be helpful for estimating the local connectivity and geometry of the input point cloud. We thus resort to the neural network to filter out the incorrect candidate triangles and provide cues for sorting the remaining candidates.

In training, we have ground truth mesh and we follow the above remeshing algorithm to generate a label for each candidate, which serves as dense supervision for the candidate classification network. Specifically, the candidates are divided into $l + 1$ categories. The incorrect candidates with IER $\geq \tau$ are in category 0. The remaining candidates are near the ground truth surface and are divided into l categories according to their distances to the surface. Empirically, we set $l = 2$ in all our experiments, which means candidates that are very close to the ground truth surface are in category 1, and other correct ones are in category 2. The network is thus trained to predict a 3-class label for each candidate.

As shown in Fig. 4, our network consists of several parts. We first utilize SparseConvNet [17], which are designed to process spatially-sparse data with convolutional operations, for point cloud feature extraction. Specifically, it maps the input point cloud into a feature set $\{\varphi(p_i) | p_i \in P\}$, where $\varphi(p_i) \in \mathbb{R}^{3+C}$ is a concatenation of the xyz coordinates and the C dimensional embedding of a point p_i. The three latent features $\varphi(u)$, $\varphi(v)$, and $\varphi(w)$ are then concatenated

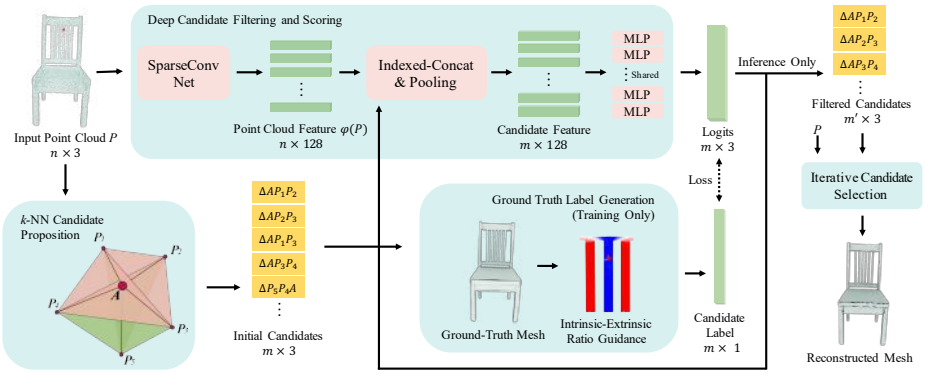

Fig. 4: Full pipeline of our reconstruction algorithm: given a point cloud as input, we first propose a set of candidate triangles. During training, the network is trained to classify the candidate triangles with the supervision of intrinsic-extrinsic ratio. During inference, the predicted label is used to filter out and score the remaining candidate triangles, which are then merged into the output mesh by our iterative selection algorithm.

together for every candidate triangle face with vertices u, v and w. A symmetry function (e.g., a max-pooling layer) then takes the concatenated feature as input to aggregates the information from the three vertices. The resulting tensors are fed into a shared weight multiple layer perceptron (MLP) to predict the final label for each triangle. The convolutional operation and the shared weight MLP are designed to inspire the learning of generalizable local priors across different parts and shapes.

During inference, we utilize the predicted labels to filter out the incorrect candidates and then sort the remaining candidates according to their labels and the length of their longest edges. We finally merge them into the output mesh through a greedy post-processing, as in the remeshing algorithm.

Although we utilize the ratio between the geodesic distance and the Euclidean distance (IER) to determine the label of the candidate triangles, the neural network does not need to regress the geodesic distances directly, since it could learn to utilize local geometric and semantic cues to recognize those incorrect triangles. In our experiments, the network achieves high accuracy classifying the candidate triangles, and thus enables high-quality mesh reconstruction. Please refer to the supplementary materials for the experiment of directly inferring the geodesic distance, which produces less effective results.

We find our model can transfer well to unseen categories. Since the estimation of local connectivity relies more on local inductive biases, which encourages the generalizability. Similar phenomenons are also observed in [27,1,31]. Although our method proposes $O(k^2n)$ candidate triangles, the candidate classification can be processed in batch. The total inference time for a single point cloud with $12,800$ points is typically less than 10 seconds.

4 Experiments

4.1 Data Generation and Network Training

To evaluate our method, we sampled 23,108 synthetic CAD models, which cover eight categories, from the ShapeNet dataset [5]. All the models are normalized to the origin of the canonical frame with a diameter of 1. Since there is no watertight or manifold guarantee for the ShapeNet meshes, we pre-clean the meshes to facilitate the calculation of geodesic distance. Specifically, we merge the vertices that are within a small distance of 0.001, remove all the duplicate faces, and split the edges that go through the non-endpoint vertices. For each model, we then utilize the Poisson-disk sampling [9] to uniformly sample 10,000 $\sim$ 12,800 points on the mesh surface. The points are then unified into the size of 12,800 by randomly replicating, and serve as the input point cloud. For each vertex, we construct a k-NN graph ($k = 50$). We follow the idea of the MMP algorithm [40] to calculate the exact geodesic distance between each vertex and its small neighborhood over the mesh surface. For calculating the Euclidean distance between a candidate triangle and the ground truth mesh, we randomly sample 10 points on the candidate face and average the distances of the sampled points to the ground truth mesh surface. To filter out the incorrect candidates, we empirically set τ to be 1.3 in our experiments. The correct candidates are then divided into two categories according to their distances to the mesh surface with a threshold of 0.005.

We reserve 3,146 models for testing, and the rest is used for training. All the 8 categories are trained together in a single model. The resolution of the SparseConvNet is set to be 150. In each training iteration, we randomly sample 25,000 candidate triangles per shape. For the following evaluations, our models were trained on a single Nvidia 2080Ti GPU for 50 epochs with a batch size of 24. The Adam optimizer was used. The initial learning rate was set to 1e-3 and decayed by 0.7 per 5 epochs.

4.2 Comparison with Existing Methods

We compare our method to both traditional surface reconstruction methods and learning-based mesh generative methods. Traditional methods include screened Poisson surface reconstruction (PSR) [24,25], marching cube (APSS variant) [30,20], and ball-pivoting algorithm (BPA) [2]. Learning-based methods include AtlasNet [19], Deep Geometric Prior (DGP) [45], Deep Marching Cubes (DMC) [26], and DeepSDF [36], which are representatives of different paradigms. Since meshes in ShapeNet are not manifolds, it's not trivial to calculate point normals with consistent directions, but many algorithms rely on correct normals. We thus employ PCPNet [21] to predict normals for the input point clouds. We then utilize MeshLab [8] to reconstruct the meshes for the three traditional methods. Specifically, for Poisson surface reconstruction, an outlier removal as post-processing is applied. Since the ball-pivoting algorithm is sensitive to the radius, we tried the auto-guess mode of the MeshLab and also selected 3 radii manually to choose

Table 1: Quantitative results on the ShapeNet test set: F-score with two different thresholds, Chamfer distance, and normal consistency score.

	F-score (μ) ↑								F-score (2 μ) ↑							
category	PSR	MC	BPA	ATLAS	DMC	DSDF	DGP	OURS	PSR	MC	BPA	ATLAS	DMC	DSDF	DGP	OURS
display	0.468	0.495	0.834	0.071	0.108	0.632	0.417	**0.903**	0.666	0.669	0.929	0.179	0.246	0.787	0.607	**0.975**
lamp	0.455	0.518	0.826	0.029	0.047	0.268	0.405	**0.855**	0.648	0.681	0.934	0.077	0.113	0.478	0.662	**0.951**
airplane	0.415	0.442	0.788	0.070	0.050	0.350	0.249	**0.844**	0.619	0.639	0.914	0.179	0.289	0.566	0.515	**0.946**
cabinet	0.392	0.392	0.553	0.077	0.154	0.573	0.513	**0.860**	0.598	0.591	0.706	0.195	0.128	0.694	0.738	**0.946**
vessel	0.415	0.466	0.789	0.058	0.055	0.323	0.387	**0.862**	0.633	0.647	0.906	0.153	0.120	0.509	0.648	**0.956**
table	0.233	0.287	0.772	0.080	0.095	0.577	0.307	**0.880**	0.442	0.462	0.886	0.195	0.221	0.743	0.494	**0.963**
chair	0.382	0.433	0.802	0.050	0.088	0.447	0.481	**0.875**	0.617	0.615	0.913	0.134	0.345	0.665	0.693	**0.964**
sofa	0.499	0.535	0.786	0.058	0.129	0.577	0.638	**0.895**	0.725	0.708	0.895	0.153	0.208	0.734	0.834	**0.972**
average	0.407	0.446	0.769	0.062	0.091	0.468	0.425	**0.872**	0.618	0.626	0.885	0.158	0.209	0.647	0.649	**0.959**

	Chamfer Distance (×100) ↓								Normal Consistency ↑							
category	PSR	MC	BPA	ATLAS	DMC	DSDF	DGP	OURS	PSR	MC	BPA	ATLAS	DMC	DSDF	DGP	OURS
display	0.273	0.269	0.093	1.094	0.662	0.317	0.293	**0.069**	0.889	0.842	0.952	0.828	0.882	0.932	**0.974**	
lamp	0.227	0.244	0.060	1.988	3.377	0.955	0.167	**0.053**	0.876	0.872	0.951	0.593	0.725	0.864	**0.963**	
airplane	0.217	0.171	0.059	1.011	2.205	1.043	0.200	**0.049**	0.848	0.835	0.926	0.737	0.716	0.872	**0.955**	
cainet	0.363	0.373	0.292	1.661	0.766	0.921	0.237	**0.112**	0.880	0.827	0.836	0.682	0.845	0.872	**0.957**	
vessel	0.254	0.228	0.078	0.997	2.487	1.254	0.199	**0.061**	0.861	0.831	0.917	0.671	0.706	0.841	**0.953**	
table	0.383	0.375	0.120	1.311	1.128	0.660	0.333	**0.076**	0.833	0.809	0.919	0.783	0.831	0.901	**0.962**	
chair	0.293	0.283	0.099	1.575	1.047	0.483	0.219	**0.071**	0.850	0.818	0.938	0.638	0.794	0.886	**0.962**	
sofa	0.276	0.266	0.124	1.307	0.763	0.496	0.174	**0.080**	0.892	0.851	0.940	0.633	0.850	0.906	**0.971**	
average	0.286	0.276	0.116	1.368	1.554	0.766	0.228	**0.071**	0.866	0.836	0.923	0.695	0.794	0.884	**0.962**	

the best radius. Please refer to the supplementary materials for more details about the baseline algorithms.

We use F-score [43], Chamfer distance [13], and normal consistency score [32] to evaluate the methods. Specifically, we uniformly sample 10^6 points on the reconstructed mesh and the ground truth mesh respectively, and calculate a normal for each point. For F-score, the precision and recall are calculated by checking the percentage of points in one point set that can find a neighbor from the other point set within a threshold μ. The F-score is then calculated as the harmonic mean of precision and recall. To align with different sampling densities, μ is set to $\sqrt{S/10^6}$ for each shape and S is the surface area of the ground truth mesh. For Chamfer Distance (CD), it measures the mean distance between each point in one point set to its nearest neighbor in the other point set. The normal consistency score is defined as the average of the absolute dot product between the normals in one mesh and the normals of the corresponding nearest points in the other mesh.

The quantitative results are shown in Table 1. Our method outperforms all the baseline algorithms with regard to the three metrics across all categories with a large margin. Fig. 5 demonstrates some of the representative cases. As shown in the figure, existing learning-based mesh generative methods (e.g., AtlasNet, DMC, and DeepSDF) are difficult to generate fine-grained details and generalize to unseen shapes (see the vessel of AtlasNet and the desk of DeepSDF). They even fail to preserve all the structures which are revealed in the input (see the bench and the desk of DMC for missing parts). This is due to the fact that they generate meshes based only on an object-level shape embedding. Also, since DMC utilizes 3D convolutional networks to predict the surface, the generated meshes are limited to a low resolution (i.e., $32 \times 32 \times 32$). As for the three traditional methods, they generally preserve the overall structures from the input point clouds. However, it may be difficult for them to handle ambiguous structures when the resolution of the input point cloud is limited. For example,

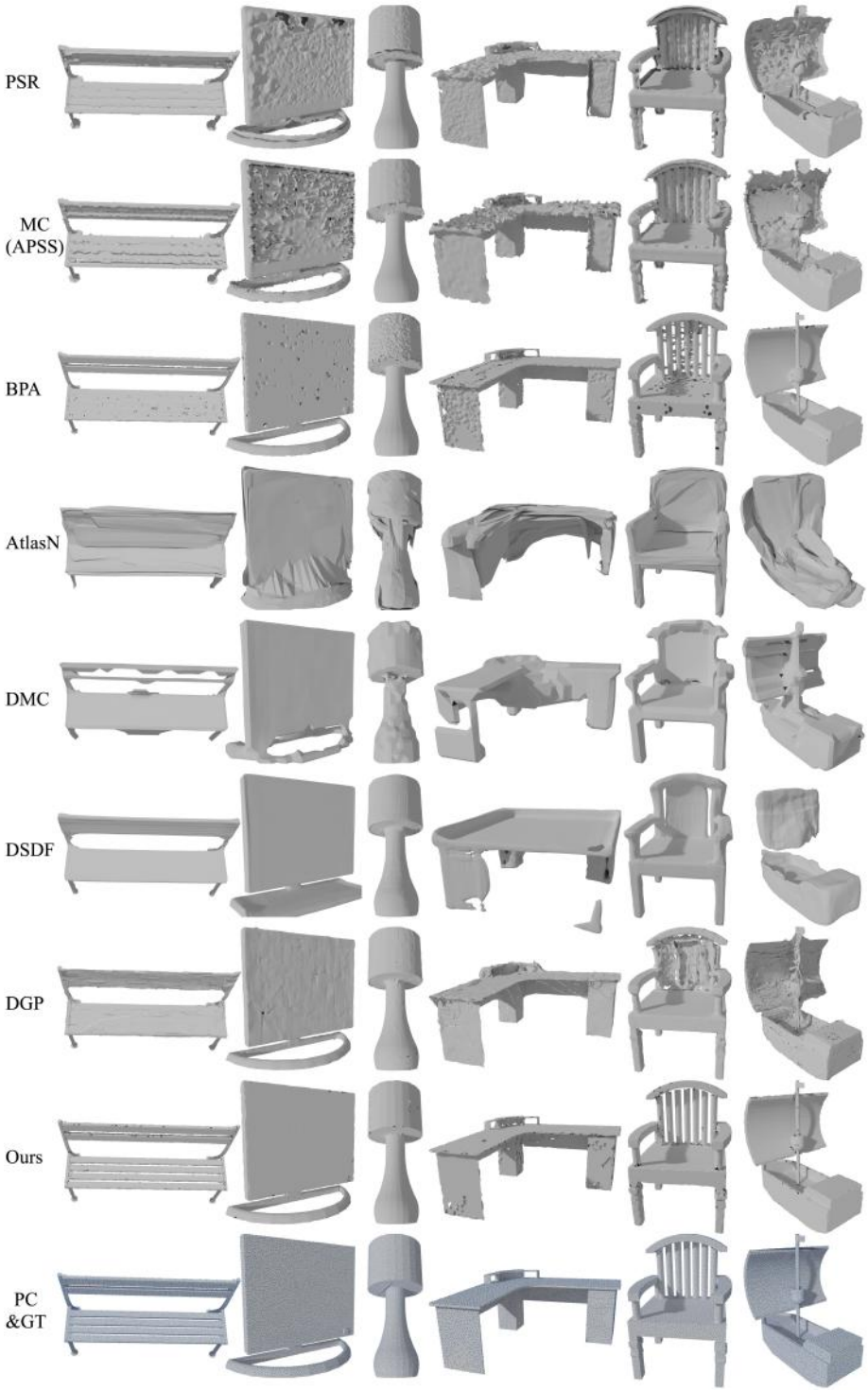

Fig. 5: Poisson surface reconstruction, marching cube (APSS), ball-pivoting algorithm, AtlasNet, Deep Marching Cubes, DeepSDF, Deep Geometric Prior, our method, and ground-truth meshes with input point clouds are shown from top to bottom.

Fig. 6: Our method can transfer to unseen categories. The first row shows the results of our method where the shape categories are unseen during training. The second row shows the results of ball-pivoting algorithm for comparison.

Table 2: First row: the results of our leave-one-out cross-validation. Second row: the results of feeding all candidates directly to the post-processing algorithm without filtering. Last row: our original results for comparison.

	F-score (μ) ↑	F-score (2μ) ↑	CD($\times$100) ↓	normal similarity ↑
leave-one-out	0.870	0.959	0.072	0.961
w/o filtering	0.728	0.882	0.110	0.862
ours	0.872	0.959	0.071	0.962

they failed to distinguish the thin structures consisting of spatially close surfaces, such as the display and the desk, and produced much distortion. Without priors and reasoning about the shape, nor can they distinguish those independent but spatially adjacent parts, such as the long strips of the bench and the armchair. In contrast, our method fully utilizes the input point cloud, which enables the generation of fine-grained structures. The learned local priors also help us to better estimate the local connectivity and generalize to unseen categories.

4.3 Ablation Studies

We would like to evaluate the transferability of our method, the importance of the candidate filtering, and the robustness with regard to various situations.

Category Transferability We utilize leave-one-out cross-validation to evaluate the category transferability of our method. Specifically, we trained a separate model for each of the eight categories with the training data of all the rest seven categories, and a test is made for that category. Table 2 reports the quantitative results where the numbers are averaged across all the eight categories. Compared to the model that trained on all categories and tests on all categories (the third row), the performances are quite similar, from which we can infer that our method does not heavily rely on category-specific priors and thus enable strong generalizability. Examples in Fig. 6 have further confirmed our belief that local priors can be transferred across different categories.

Fig. 7: In each pair, the left one is the result of method, and the right one is the result without candidate filtering.

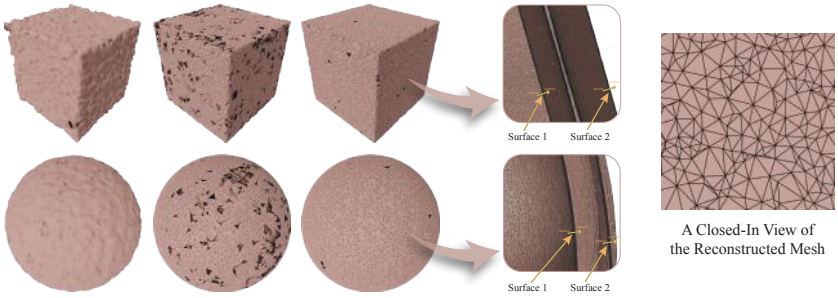

Fig. 8: Qualitative results on uniformly randomly sampled point clouds. The ground truth meshes are concentric dual spheres and concentric dual cubes. From left to right: results from Poisson surface reconstruction (PSR) [24,25], results from the ball-pivoting algorithm [2], and results from our method. The second to the last column is a zoomed-in sliced view showing the interior of our results. The last column is a zoomed-in view of the reconstructed surface.

Effect of Filtering To verify the importance of the candidate filtering, we also test a variant where all the proposed candidates are directly fed into the post-processing algorithm without filtering. The quantitative results are shown in Table 2, from which we find that the performance drops dramatically. Fig. 7 also shows some of the qualitative comparisons. Without passing the candidates through the network and filtering out all the incorrect candidates, the method cannot handle ambiguous structures anymore.

Distribution of Point Cloud In general, our method favors evenly distributed point clouds, and applying a Poisson-disk sampling as pre-processing could improve the performance. To examine the robustness to other point cloud distributions, we test our method on uniformly randomly sampled point clouds, virtual scanned point clouds, as well as Poisson-disk sampling with different density. Due to the space limits, we only include the results of uniformly randomly sampled point clouds in the text. Please refer to the supplementary materials for the results of other experiments. Note that though misleading, the uniformly randomly sampled points are typically not evenly distributed over the surface, as shown in the last column of Fig. 8. Two simple shapes are tested, namely the

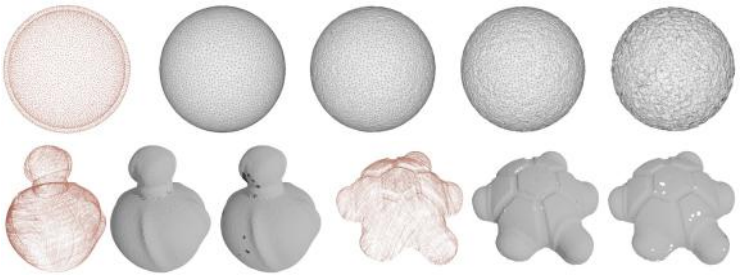

Fig. 9: First row: a point cloud of a concentric dual sphere and reconstructed meshes with different levels of input noise. Second row: two real-world LiDAR scans from Aim@Shape and the reconstructed meshes by our method.

concentric dual spheres and concentric dual cubes. It can be seen from Fig. 8 that although we only trained on evenly distributed Poisson-disk sampled point clouds (on ShapeNet), our method can process the uniformly randomly sampled point clouds effectively, and outperforms both PSR and BPA by a large margin. This further proves the strong generalizability of our method.

Noisy Data and Real Scans Since our method directly interpolate triangles upon input point clouds, the algorithm may be sensitive to the noise. However, as shown in Fig. 10, without explicit denoising and data augmentation mechanisms, our method is still resistant to the noise to a certain extent. As for real scans, there may be more issues, such as part missing and uneven distribution of the points. With the point set consolidation network [47] as pre-processing, our method can generate satisfying meshes from real-world LiDAR scans (see Fig. 10). In the future, we would like to explore explicit ways to propose the position of the vertices and compensate for the structural loss of input [28].

5 Conclusions

In this paper, we proposed a novel learning-based framework for mesh reconstruction that is based on the grounded point clouds and explicitly estimates the local connectivity of the points. By leveraging the intrinsic-extrinsic ratio as training guidance, the method is able to effectively distinguish the surface triangles and non-surface triangles. Extensive experiments have shown our superior performance, especially for preserving the details, handling ambiguous structures, and strong generalizability.

Acknowledgements This work was funded in part by Kuaishou Technology, NSF grant IIS-1764078, NSF grant 1703957, the Ronald L. Graham chair and the UC San Diego Center for Visual Computing.

References

1. Badki, A., Gallo, O., Kautz, J., Sen, P.: Meshlet priors for 3d mesh reconstruction. arXiv preprint arXiv:2001.01744 (2020)
2. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE transactions on visualization and computer graphics **5**(4), 349–359 (1999)
3. Boissonnat, J.D., Geiger, B.: Three-dimensional reconstruction of complex shapes based on the delaunay triangulation. In: Biomedical Image Processing and Biomedical Visualization. vol. 1905, pp. 964–975. International Society for Optics and Photonics (1993)
4. Carr, J.C., Beatson, R.K., Cherrie, J.B., Mitchell, T.J., Fright, W.R., McCallum, B.C., Evans, T.R.: Reconstruction and representation of 3d objects with radial basis functions. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 67–76 (2001)
5. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
6. Chen, Z., Tagliasacchi, A., Zhang, H.: Bsp-net: Generating compact meshes via binary space partitioning. arXiv preprint arXiv:1911.06971 (2019)
7. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5939–5948 (2019)
8. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: Meshlab: an open-source mesh processing tool. In: Eurographics Italian chapter conference. vol. 2008, pp. 129–136 (2008)
9. Corsini, M., Cignoni, P., Scopigno, R.: Efficient and flexible sampling with blue noise properties of triangular meshes. IEEE transactions on visualization and computer graphics **18**(6), 914–924 (2012)
10. Deprelle, T., Groueix, T., Fisher, M., Kim, V., Russell, B., Aubry, M.: Learning elementary structures for 3d shape generation and matching. In: Advances in Neural Information Processing Systems. pp. 7433–7443 (2019)
11. Digne, J.: An analysis and implementation of a parallel ball pivoting algorithm. Image Processing On Line **4**, 149–168 (2014)
12. Edelsbrunner, H., Mücke, E.P.: Three-dimensional alpha shapes. ACM Transactions on Graphics (TOG) **13**(1), 43–72 (1994)
13. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 605–613 (2017)
14. Gao, L., Yang, J., Wu, T., Yuan, Y.J., Fu, H., Lai, Y.K., Zhang, H.: Sdm-net: Deep generative network for structured deformable mesh. ACM Transactions on Graphics (TOG) **38**(6), 1–15 (2019)
15. Genova, K., Cole, F., Vlasic, D., Sarna, A., Freeman, W.T., Funkhouser, T.: Learning shape templates with structured implicit functions. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 7154–7164 (2019)
16. Gkioxari, G., Malik, J., Johnson, J.: Mesh r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9785–9795 (2019)
17. Graham, B., Engelcke, M., van der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 9224–9232 (2018)

18. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: 3d-coded: 3d correspondences by deep deformation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 230–246 (2018)
19. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3d surface generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 216–224 (2018)
20. Guennebaud, G., Germann, M., Gross, M.: Dynamic sampling and rendering of algebraic point set surfaces. In: Computer Graphics Forum. vol. 27, pp. 653–662. Wiley Online Library (2008)
21. Guerrero, P., Kleiman, Y., Ovsjanikov, M., Mitra, N.J.: Pcpnet learning local shape properties from raw point clouds. In: Computer Graphics Forum. vol. 37, pp. 75–85. Wiley Online Library (2018)
22. Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W.: Surface reconstruction from unorganized points. In: Proceedings of the 19th annual conference on Computer graphics and interactive techniques. pp. 71–78 (1992)
23. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 371–386 (2018)
24. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. vol. 7 (2006)
25. Kazhdan, M., Hoppe, H.: Screened poisson surface reconstruction. ACM Transactions on Graphics (ToG) **32**(3), 1–13 (2013)
26. Liao, Y., Donne, S., Geiger, A.: Deep marching cubes: Learning explicit surface representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2916–2925 (2018)
27. Liu, H.T.D., Kim, V.G., Chaudhuri, S., Aigerman, N., Jacobson, A.: Neural subdivision. arXiv preprint arXiv:2005.01819 (2020)
28. Liu, M., Sheng, L., Yang, S., Shao, J., Hu, S.M.: Morphing and sampling network for dense point cloud completion. arXiv preprint arXiv:1912.00280 (2019)
29. Liu, S., Chen, W., Li, T., Li, H.: Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. arXiv preprint arXiv:1901.05567 (2019)
30. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
31. Luo, T., Mo, K., Huang, Z., Xu, J., Hu, S., Wang, L., Su, H.: Learning to group: A bottom-up framework for 3d part discovery in unseen categories. In: International Conference on Learning Representations (2020), `https://openreview.net/forum?id=rkl8dlHYvB`
32. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
33. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 909–918 (2019)
34. Öztireli, A.C., Guennebaud, G., Gross, M.: Feature preserving point set surfaces based on non-linear kernel regression. In: Computer Graphics Forum. vol. 28, pp. 493–501. Wiley Online Library (2009)
35. Pan, J., Han, X., Chen, W., Tang, J., Jia, K.: Deep mesh reconstruction from single rgb images via topology modification networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 9964–9973 (2019)

36. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019)
37. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
38. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems. pp. 5099–5108 (2017)
39. Sharma, G., Goyal, R., Liu, D., Kalogerakis, E., Maji, S.: Csgnet: Neural shape parser for constructive solid geometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5515–5523 (2018)
40. Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S.J., Hoppe, H.: Fast exact and approximate geodesics on meshes. ACM transactions on graphics (TOG) **24**(3), 553–560 (2005)
41. Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning shape abstractions by assembling volumetric primitives. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2635–2643 (2017)
42. Turk, G., Levoy, M.: Zippered polygon meshes from range images. In: Proceedings of the 21st annual conference on Computer graphics and interactive techniques. pp. 311–318 (1994)
43. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.G.: Pixel2mesh: Generating 3d mesh models from single rgb images. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 52–67 (2018)
44. Wen, C., Zhang, Y., Li, Z., Fu, Y.: Pixel2mesh++: Multi-view 3d mesh generation via deformation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1042–1051 (2019)
45. Williams, F., Schneider, T., Silva, C., Zorin, D., Bruna, J., Panozzo, D.: Deep geometric prior for surface reconstruction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 10130–10139 (2019)
46. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Interpretable unsupervised learning on 3d point clouds. arXiv preprint arXiv:1712.07262 (2017)
47. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Ec-net: an edge-aware point set consolidation network. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 386–402 (2018)

Appendix A: More figures of the Reconstructed Meshes

Fig. 10: Reconstructed meshes of the ShapeNet test set. In each pair, the above one is the result of our method, and the below one is the result of the traditional ball-pivoting algorithm.

Appendix B: Pseudo Code of the Remeshing Algorithm

Algorithm 1: Remeshing with Intrinsic-Extrinsic Ratio as Guidance

input : Reference mesh M_R, point cloud P sampled on M_R, IER threshold τ

$M_N.V = P$;
$M_N.F = \emptyset$;
Construct a k-NN graph on P and propose candidate triangle faces;
Calculate the intrinsic-extrinsic ratio for each candidate;
Filter out the incorrect candidates with IER $\geq \tau$;
Sort the remaining candidates with respect to their distance to M_R and the length of their longest edges;
for *each remaining candidate triangle* f_i **do**
    if $\exists$ $f_j \in M_N.F$ *intersects with* f_i ***or*** $\exists$ *edge* $\in M_N$ *has more than two incident faces after adding* f_i **then**
        continue;
    end
    else
        $M_N.F = M_N.F \cup \{f_i\}$;
    end
end
return M_N;

Appendix C: Confusion Matrix of the Candidate Classification

Table 3 shows the confusion matrix of the candidate classification on the ShapeNet test set. Specifically, category 0 indicates the incorrect triangles filtered out by the IER. Both category 1 and 2 are the correct candidates. The candidates of category 1 are closer to the ground truth surface than candidates of category 2. We find that the overall performance of the candidate classification is satisfactory.

Table 3: Confusion matrix of the candidate classification on the ShapeNet test set.

	category 0	category 1	category 2
category 0	89.8%	3.9 %	6.3%
category 1	1.4%	96.5%	2.1%
category 2	20.2%	8.8%	71.0%

Appendix D: Results of Different Sampling Densities

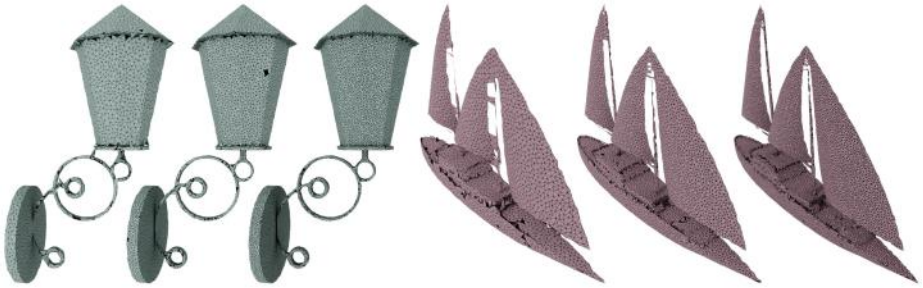

Fig. 11: Qualitative results of different sampling densities. Training on point clouds with 12,800 points (middle), our method can transfer to point clouds with 6,400 (left) and 25,600 points (right).

Table 4: Quantitative results on point clouds with different densities (F-score with two thresholds, Chamfer distance, and normal consistency score). The results are averaged across the eight categories.

#points	F-score(μ) ↑	F-score(2μ) ↑	CD ($\times$100) ↓	normal ↑
6,400	0.814	0.916	0.091	0.949
12,800	0.872	0.959	0.071	0.962
25,600	0.907	0.983	0.062	0.969

To evaluate the transferability of our method across different sampling densities. We trained our models on point clouds with $12,800$ points and test it on point clouds with $6,400$ and $25,600$ points respectively. Fig. 11 shows the qualitative results, from which we find that our method can generalize to different density distributions, and as the resolution of the point clouds increases, the details become more accurate. The quantitative results shown in Table 4 further confirm our arguments.

Appendix E: Results on Virtual Scans

In order to examine the robustness of our method with regard to different distributions of the input point clouds, we test our method on virtual scans of the ShapeNet models. Specifically, we utilize the VCG Lib to mimic the Kinect sensors and randomly select 10 camera poses to scan the models. The scanned point clouds of different views are fused and downsampled to 12,800 points (Poisson-disk sampling) before feeding into our method.

Fig. 12 shows the point clouds and the reconstructed meshes. Although the scanned points clouds are unevenly distributed, our method still reconstructs high-quality meshes, which demonstrates the generalizability of our method.

Please note that due to the limited number of views, the scanned point clouds may not cover the full area of the shape and the reconstructed meshes are thus also incomplete.

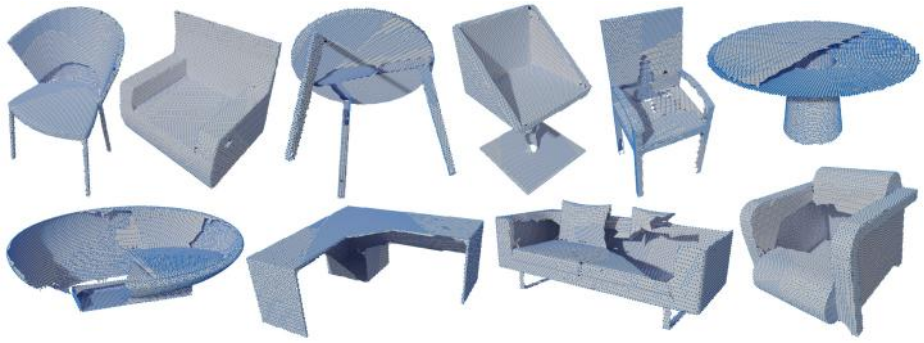

Fig. 12: Input point clouds with 12,800 points and the reconstructed meshes by our method.

Appendix F: Hyper-parameters of the Experiments

Some baseline algorithms require the normals of the point clouds as input. However, meshes in ShapeNet are not perfect manifolds, there are lots of flipped faces and faces that are visible from multiple views. As a result, it's not trivial to determine the consistent directions of the normals (point inward or point outward). We thus employ PCPNet to predict normals for the input point clouds. We then utilize MeshLab to reconstruct the meshes for the three traditional methods. We basically follow the default hyper-parameters of MeshLab. Specifically, the reconstruction depth of PSR is set to be 8, and the grid resolution of Marching Cube (APSS) is set to be 200. Since ball-pivoting algorithms are sensitive to the radius, we tried the auto-guess mode of the MeshLab and also manually selected 3 radii $1\%, 2\%$, and 3% to choose the best radius. For PSR, we also provide outlier removal as post-processing, which filters out all the vertices that cannot find a point in the input point cloud within a radius of 0.02.

For all the learning-based methods, we used our training set (point clouds with 12,800 points) and followed their released hyperparameters to retrain the model. For DeepSDF, we retrained the network category-by-category. Since both positive and negative signed distance samples are required by the DeepSDF, we assume the normal direction is known for each point in order to sample testing signed distance samples. For AtlasNet, we use the "Autoencoder 25 Squares" model. For Deep Geometric Prior (DGP), "radius" is set to be 0.05, "local-epochs" and "global-epochs" is set to be 125, and "upsamples-per-patch" is set to be 64. Since DGP outputs point clouds of millions of points and reconstructing meshes on such point clouds is time-consuming, we directly use the generated

points to calculate the F-score and Chamfer distance, and do not report the normal consistency score.

For the noise experiments, we test on the point clouds of concentric dual spheres. The diameters of the two spheres are 0.933 and 1 respectively. We add a Gaussian noise of a standard deviation of $0.001 \times t$ to point coordinates, where t indicates the level of the noise. For the figure of the main paper, t is set to be 0, 0.8, 1.6, and 3.2 respectively.

Appendix G: Geodesic Distance Regression

In our method, we use the ratio of the geodesic distance and Euclidean distance as guidance to train the network to classify the candidate triangles. Another straightforward idea is that regressing the geodesic distances between pairs of vertices directly with methods such as GeoNet and then use the regressed geodesic distance to classify the candidates. We tried this ablated version to estimate its effectiveness. Specifically, since GeoNet didn't release their source code, we modify our classification network to regress the geodesic distance directly. As we only care about the geodesic distance within a small neighborhood, the training set only contains the pairs between each vertex and its k-nearest neighbors, and the distances are truncated with a threshold of 0.1.

Table 5: Results of the geodesic distance regression. The first row shows the relative error of the predicted distance. The second row shows the accuracy of the candidate classification using the predicted geodesic distance.

category	airplane	cabinet	chair	display	lamp	sofa	table	vessel	average
relative error	137.7%	44.0%	59.8%	47.6%	132.0%	47.5%	49.1%	91.7%	70.7%
accuracy	58.0%	83.2%	61.9%	70.1%	47.4%	71.1%	73.3%	51.6%	65.7%

Table 5 shows the results of our regression version. We find that it's not easy for the network to regress the geodesic distances and the predicted distances are not accurate. Using the predicted geodesic distance to classify the candidate triangles (into 2 categories) also produces poor results. The accuracy of 65.7% is much lower than the accuracy of our original version of 91.8%. In fact, in our original version, the network does not need to regress the geodesic distance. The labels inferred by the ratio of the geodesic distance and the Euclidean distance only serve as the supervision for the network to learn some local priors to recognize those incorrect triangles, which is a more reasonable and easy task.