DMNet: Delaunay Meshing Network for 3D Shape Representation

Chen Zhang[1] Ganzhangqin Yuan[1,2] Wenbing Tao[1,*]

[1]National Key Laboratory of Science and Technology on Multi-spectral Information Processing,
School of Artifical Intelligence and Automation, Huazhong University of Science and Technology, China
[2]TuKe Research

{zhangchen_, gzq_yuan, wenbingtao}@hust.edu.cn

Abstract

Recently, there has been a growing interest in learning-based explicit methods due to their ability to respect the original input and preserve details. However, the connectivity on complex structures is still difficult to infer due to the limited local shape perception, resulting in artifacts and non-watertight triangles. In this paper, we present a novel learning-based method with Delaunay triangulation to achieve high-precision reconstruction. We model the Delaunay triangulation as a dual graph, extract local geometric information from the points, and embed it into the structural representation of Delaunay triangulation in an organic way, benefiting fine-grained details reconstruction. To encourage neighborhood information interaction of edges and nodes in the graph, we introduce a local graph iteration algorithm, which is a variant of graph neural network. Moreover, a geometric constraint loss further improves the classification of tetrahedrons. Benefiting from our fully local network, a scaling strategy is designed to enable large-scale reconstruction. Experiments show that our method yields watertight and high-quality meshes. Especially for some thin structures and sharp edges, our method shows better performance than the current state-of-the-art methods. Furthermore, it has a strong adaptability to point clouds of different densities.

1. Introduction

Surface reconstruction from a given point cloud is a long-standing problem in computer vision and graphics [7, 1, 28, 27, 34, 59, 4, 60, 25, 21, 17, 36, 8]. A widely used framework is to first compute an implicit representation and then extract the resulting surface using Marching Cubes [37]. Implicit methods typically produce a watertight mesh and have the advantage of noise resistance, but present over-smoothing and loss of detail in the face of fine structures [4]. On the other hand, some methods from computational geometry [16, 5, 31, 22, 44, 33] construct explicit meshes by point set triangulation, where the key feature is that the vertices of the output mesh come from the input point cloud. As an advantage, these explicit methods respect the original point set to preserve sharp features and fine structures.

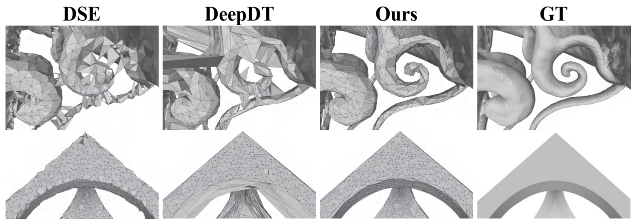

Figure 1: A comparison of our approach with the two state-of-the-art explicit methods on challenging sharp edges and complex thin structures composed of sparse data.

More recently, several learning-based explicit methods with point set triangulation have been proposed, which demonstrate a good performance. A category of methods, such as DSE [51], PointTriNet [52] and IER Meshing [35], learn to generate connected triangles locally in an iterative manner. They can preserve some linear structures to some extent, but ensuring watertightness is a challenge for them. Although DSE reports good results with few non-watertight triangles, it still struggles with inferring triangular connectivity on complex topologies. Another category of method, DeepDT [38], learns the classification of tetrahedrons from Delaunay triangulation. It improves upon traditional approaches that use graph cuts and visibility information, by employing a multi-layer Graph Convolution Network (GCN) [29] to break the limitation of visibility information. DeepDT achieves good results on objects with hundreds of thousands of points, but struggles with reconstructing sparse data, especially for complex thin structures and sharp edges (see Figure 1). There are several important reasons for this. 1) The tangent plane features used

[*]Corresponding author.

between points are inadequate for expressing the local distribution information of the point cloud, which leads to a weak perception of the local geometry. 2) The simple combination of point features without structural representation results in ambiguous tetrahedral features for classification. 3) The employed graph neural network lacks a strong connection to the Delaunay triangulation structure, providing inadequate interaction with local information. Moreover, dealing with large-scale data is also a tricky problem for DeepDT.

Nonetheless, the application of Delaunay triangulation without visibility information remains a topic worth exploring. In contrast to generating connected triangles locally, Delaunay triangulation pre-constructs global candidate triangles, containing a more accurate surface approximation [2] and ensuring the watertightness and non-self-intersection of the mesh. Additionally, Delaunay triangulation possesses an excellent property of adapting to point density, enabling it to strike a balance between resolution, efficiency, and resource occupation. It suggests that further exploration of the potential of Delaunay triangulation for high-precision reconstruction is necessary to address some of the challenges faced by current learning-based explicit approaches, such as 1) high-quality reconstruction of thin structures and sharp edges, 2) good compatibility with data of different densities on a well-trained model, especially for sparse data, and 3) efficient scalability to handle large-scale data.

In this paper, we present Delaunay Meshing Network (DMNet), a completely local approach for surface reconstruction that captures fine details and scales efficiently. In stark contrast to the previous approaches, each tetrahedron and triangular facet in the Delaunay triangulation can be individually encoded in our method. To accurately perceive local geometry, our graph feature encoder captures both their morphological structure information and the neighborhood points distribution information of their vertices. Additionally, an offset position of the vertex is calculated to enable the organic embedding of the vertex features into the structural representation. This enables our method to better capture geometric differences between tetrahedrons inside and outside the local surface for robust labeling. To obtain more comprehensive neighborhood information, we design a variant of the graph neural network called Local Graph Iteration. As a form of local graph structure regularization, it utilizes simple self-attention modules and MLPs to iteratively process one-hop node features, without relying on a global adjacency matrix in GCN. Furthermore, we take into account the connection relationship between nodes and adjacent edges in the Delaunay triangulation structure, incorporating edge features into the information interaction. Compared to GCN, it provides a more powerful local processing capability with only a few iteration layers. To avoid large triangle artifacts, we further propose a shape constraint loss to constrain the shape of the extracted triangles. Benefiting from the local network processing, an octree-based scaling strategy is proposed for separating the dual graph, which allows our approach to handle large-scale real data with millions of points like DTU [24]. Our experiments demonstrate the superiority of our approach across some challenging tasks. In summary, our contributions are as follows:

1) Our graph feature construction integrates multi-geometry information from Delaunay triangulation, allowing our method to preserve rich fine-grained details, especially for thin structures and sharp edges.

2) Local Graph Iteration algorithm is tailored for Delaunay triangulation, promoting ample interaction among neighborhood information. Both it and the geometric constraint loss allow better classification of tetrahedrons.

3) The robust local processing capability of our network allows for a good generalization to data of varying densities, even sparse data. Additionally, the combination with our scaling strategy enables efficient scalability with high precision.

2. Related Work

2.1. Traditional Geometric Mesh Reconstruction

The traditional geometric methods mainly include implicit methods and explicit methods. Implicit methods typically compute a field function to fit the point cloud and then use Marching Cubes to extract isosurfaces [27, 28, 58, 9, 47, 26, 41, 45, 46, 18]. They are suitable for compact point clouds with tight bounding boxes, but lead to expensive voxelization and fail to preserve the given points in the final mesh. [4] discusses in detail the over-smoothing and loss of detail produced by implicit methods. Some explicit methods estimate local surface connectivity and directly connect the sampled points through triangles [57, 6, 30]. They respect the input data, i.e., the vertices of output surface are the points in the input point cloud. For example, Alpha shapes [16] and Ball pivoting [5] utilize the concept of a rolling ball on the surface to derive local triangles. Another strategy is to apply Delaunay triangulation to the input points [32, 44, 23, 62, 33]. L-Method [31] is a classic method using graph cuts and visibility information to label tetrahedrons as inside/outside. By relying on visibility constraints, this family of methods has achieved remarkable performance in solving some challenging tasks. However, they fail to handle point clouds without visibility information.

2.2. Learning-based Mesh Reconstruction

The recent rapid development of deep learning has led to the emergence of more solutions for surface reconstruction

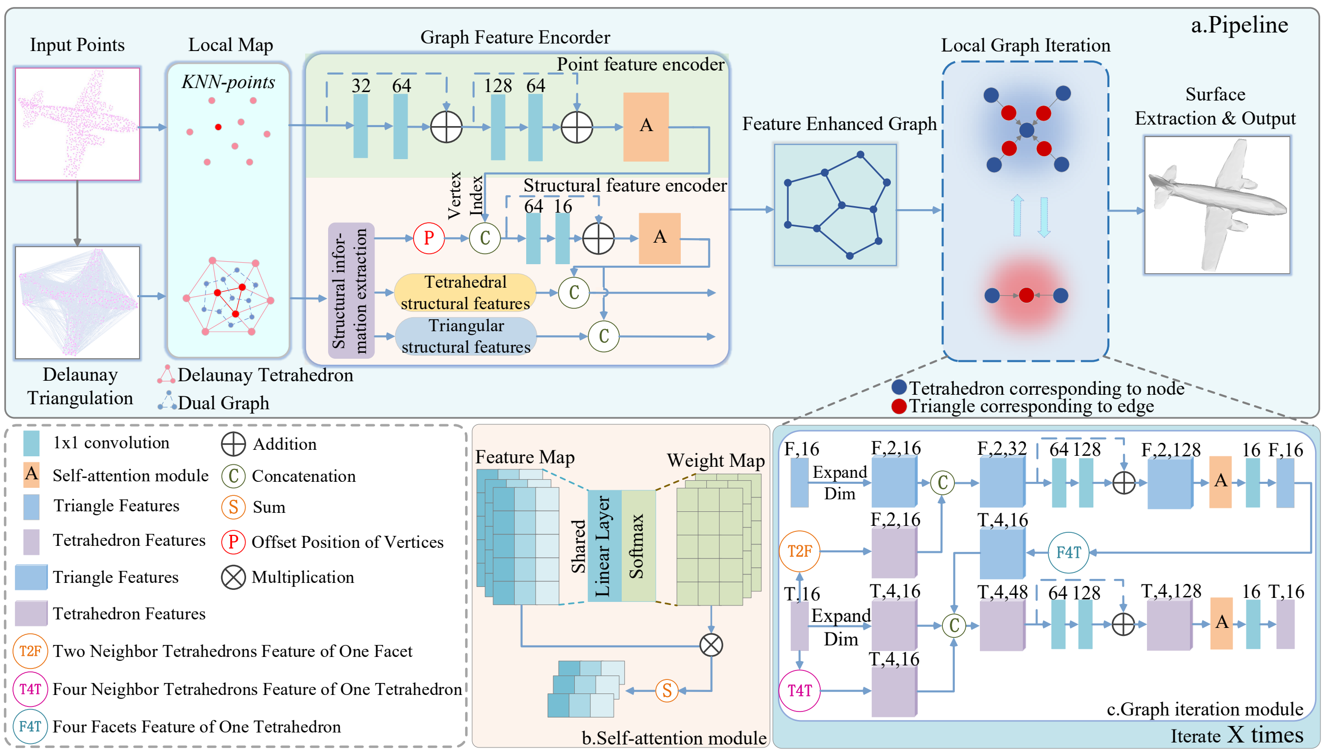

Figure 2: The pipeline of DMNet. Our network contains two main parts, one is the local feature construction of nodes and edges in the dual graph, and the other is the Local Graph Iteration algorithm to exchange neighborhood information.

tasks. Many learning-based methods learn implicit representations from voxels or octrees [12, 48, 20, 49, 36, 40, 39, 3, 14, 15, 19, 53, 55], facing similar problems as traditional implicit methods. For example, ONet [42] and ConvONet [50] learn continuous 3D occupancy functions. SAP [49] employs a differentiable Poisson solver to quickly solve the indicator function. NDC [11] directly predicts the position of mesh vertices, improving the complex dual contouring algorithm. SSRNet [43] and SSRNet+ [61] learn octree vertex classification and takes advantage of the regular division of octrees to achieve batch learning of large-scale data.

Some approaches focus on learning connectivity from point set triangulation and creating explicit meshes directly from input data. PointTriNet [52] uses a local patch-based network for predicting connectivity. IER Meshing [35] selects triangles in the candidate set by predicting the ratio of the Geodesic distance to the Euclidean distance. DSE [51] learns logarithmic maps and applies two-dimensional Delaunay triangulation to pick connected triangles. These methods respect the input points and have the potential to retain rich details. However, the meshes they create are typically non-watertight and lack proper triangles to effectively represent sharp features. Moreover, the expensive geometric calculations limit their application in large-scale data. Building on the improvements of traditional graph cuts, DeepDT [38] provides an approach that uses GCN to label Delaunay tetrahedrons without the need for visibility information. However, its over-reliance on tangent plane features between points and neglect of structural representation make it only applicable to dense point clouds. On the other hand, Sulzer et al. [54] also employ GCN to integrate visibility information, which stands in a different perspective from our approach.

3. Method

Given a point cloud $\mathcal{P} = \{p_i \mid i = 1, ..., N\}$ with N points, where p_i is the coordinate of one point with normal n_i, we construct Delaunay triangulation structure $\mathcal{D}$ for it, where $\mathcal{D}$ is the set of tetrahedrons and the four vertices of each tetrahedron are 3D points in $\mathcal{P}$. At the boundary, there are some infinite tetrahedrons sharing an infinite vertex. Such a setting ensures that each tetrahedron has four neighbor tetrahedrons sharing common triangular facets. We model the Delaunay triangulation structure $\mathcal{D}$ as a dual graph $\mathcal{G}$ with tetrahedrons as nodes and triangles as edges. Our aim is to combine the structural information of Delaunay triangulation and the neighborhood information in the graph to achieve accurate labeling of tetrahedrons without visibility information. The pipeline of our method is illustrated in Figure 2.a.

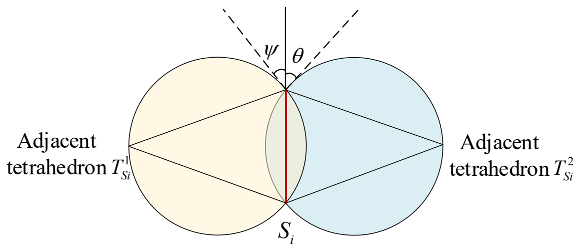
ψ
θ
Adjacent tetrahedron T_{Si}^1
Adjacent tetrahedron T_{Si}^2
S_i

Figure 3: 2D example of additional morphological feature Ang^i for triangle S_i. The red line represents the triangle S_i. The dashed lines represent the tangent planes of the circumscribed spheres of its two adjacent tetrahedrons.

3.1. Graph Feature Extraction

In our approach, the local geometric information of each node and edge in graph $\mathcal{G}$ is encoded separately, which is completely different from previous methods. The combination of neighborhood points distribution information and structural representation is a key insight in our approach.

Point Feature Encoding For each reference point p_i in $\mathcal{P}$, we search its K nearest neighbors $p_i^k, k = 1, ..., K$ with normal n_i^k. The relative position vector $r_i^k = p_i^k - p_i$ can well express the relative geometric direction and the relative geometric distance between p_i and p_i^k. We first take p_i as the center and aggregate r_i^k and n_i^k of the K nearest neighbors as the initial feature of p_i to represent the local point cloud distribution. The normal n_i^k is regularized as equation (1). Afterwards, we apply a feature mapping function F_{map} implemented by a short MLP to process the local geometric information. Finally, a weight matrix W_i generated by a simple self-attention module is used to pool the features. As shown in Figure 2.b, the attention module uses a shared linear layer represented by FC to perform the feature transformation in the same dimension as the input, followed by a softmax operation to compute the weight matrix. The total process can be formulated as follows:

$$H_i^k = F_{map}\left(\left(p_i^k - p_i\right) \oplus n_i^k\right), \quad \left\|n_i^k\right\| = 1 \tag{1}$$

$$W_i^k = Softmax\left(FC\left(H_i^k\right)\right) \tag{2}$$

$$F_i = \sum_{i=1}^{K} H_i^k \odot W_i^k \tag{3}$$

where $\oplus$ represents the concatenation operation and $\odot$ represents the product of elements. $\|\cdot\|$ represents the $L2$-norm of one vector.

Structural Representation After extracting the local features of each point, we further construct structural features of the nodes and edges in $\mathcal{G}$. In this process, we consider morphological structure information and take the relative distribution of vertices as one type of it. For a tetrahedron T_i with four vertices v_i^j $(j = 1, 2, 3, 4)$, we first search for a geometric center position o_i, which can be obtained by taking the average position of v_i^j. An offset distance $od_i^j = o_i - v_i^j$ is calculated as the abstract coordinate of v_i^j in tetrahedron T_i. Then we concatenate it to the indexed feature Fv_i^j from generated points features, and use an attention module and a mapping function G_{map} implemented by a short MLP to automatically aggregate important vertices features for tetrahedron classification. Moreover, to capture more structural features of the tetrahedron, we calculate the longest side length L_{max}^i, the shortest side length L_{min}^i, the volume Vol^i and the radius R^i of the circumscribed sphere of T_i as additional manual features. These features are sufficient to represent the structure of the space occupied by a tetrahedron. The final feature $F\left(T_i\right)$ of T_i is calculated as follows:

$$H_i^j = G_{map}\left(\left(o_i - v_i^j\right) \oplus Fv_i^j\right) \tag{4}$$

$$M_i = \sum_{j=1}^{4} H_i^j \odot W_{ti}^j \tag{5}$$

$$F\left(T_i\right) = M_i \oplus L_{max}^i \oplus L_{min}^i \oplus Vol^i \oplus R^i \tag{6}$$

where $\oplus$ represents the concatenation operation and W_{ti}^j is the weight matrix generated by the attention module. While H_i^j and M_i are intermediate variables. In particular, for the infinite tetrahedrons in the Delaunay triangulation $\mathcal{D}$, we adopt a padding strategy that sets these features to zero.

Similarly, we follow the above manner to construct the features of the edges in graph $\mathcal{G}$. In particular, for the i-th triangle S_i in $\mathcal{D}$, we calculate its longest side length, shortest side length and area as additional manual features. To better represent the relationship between S_i and its two adjacent tetrahedrons, we construct a new feature $Ang^i = 1 - min\left\{\cos\theta, \cos\psi\right\}$, where θ and ψ are the angles between S_i and the circumscribed spheres of its two adjacent tetrahedrons, as shown in Figure 3. It represents a geometric association of one triangle with its adjacent tetrahedrons in shape.

3.2. Local Graph Iteration

The raw node and edge features in graph $\mathcal{G}$ are constructed independently, thus they do not contain sufficient neighborhood graph information. Inspired by GCN using the global adjacency matrix and weight matrix for feature aggregation of one-hop nodes, we design a variant consisting of shared MLPs and self-attention modules to enhance local processing ability. Our graph iteration module is shown in Figure 2.c, where each node can be processed individually and the network parameters are shared across different nodes. In addition, since edges in $\mathcal{G}$ have a physical structure, it is meaningful to calculate their geometric features as part of the graph features. Benefiting from our well-designed edge features, the interaction of edges with adjacent nodes is also considered to enhance the aggregation of neighborhood information, making our method different from the general graph algorithms.

In the l-th graph iteration layer, $F^{l-1}(S_i) \in \mathbb{R}^{1\times C}$ is the previous feature of triangle S_i with two adjacent tetrahedrons T_{Si}^1 and T_{Si}^2. $F^{l-1}(T_{Si}^1) \in \mathbb{R}^{1\times C}$ and $F^{l-1}(T_{Si}^2) \in \mathbb{R}^{1\times C}$ are the features of T_{Si}^1 and T_{Si}^2, respectively. By concatenating $F^{l-1}(S_i)$ in the manner of Equation (7), a new feature $E^l(S_i) \in \mathbb{R}^{2\times 2C}$ is formed. Afterwards, we apply a feature aggregation function G_{agg} consisting of a short MLP and a self-attentive module to pool features. The entire process can be formulated as follows:

$$\begin{aligned} E^l(S_i) = [\, & F^{l-1}(S_i) \oplus F^{l-1}(T_{Si}^1), \\ & F^{l-1}(S_i) \oplus F^{l-1}(T_{Si}^2)\,] \end{aligned} \tag{7}$$

$$F^l(S_i) = G_{agg}\left(E^l(S_i)\right) \tag{8}$$

where $\oplus$ represents a concatenating operation in the second dimension and $[\ ,\ ]$ represents the concatenating operation in the first dimension. The output pooling feature $F^l(S_i)$ of S_i in the l-th layer has the same dimension as $F^{l-1}(S_i)$.

Furthermore, we aggregate the features of neighbor nodes and edges to construct the enhanced node features. For a tetrahedron T_i, we search for its four neighbor tetrahedrons T_i^j $(j = 1, 2, 3, 4)$, which share triangles S_{Ti}^j with T_i. Similar to the above manner of processing edges, we concatenate the features of neighboring nodes and edges to T_i in the form of Equation (9 - 10) and gain a new feature $E^l(T_i) \in \mathbb{R}^{4\times 3C}$. A feature aggregation function is applied once again to pool the feature.

$$F^l\left(T_i^j\right) = F^{l-1}(T_i) \oplus F^l\left(S_{Ti}^j\right) \oplus F^{l-1}\left(T_i^j\right) \tag{9}$$

$$E^l(T_i) = \left[F^l\left(T_i^1\right), F^l\left(T_i^2\right), F^l\left(T_i^3\right), F^l\left(T_i^4\right)\right] \tag{10}$$

$$F^l(T_i) = G_{agg}\left(E^l(T_i)\right) \tag{11}$$

As the number of iteration layers increases, each node is able to capture information from a larger receptive field. However, benefiting from its powerful local processing capability, two layers are sufficient in our experiments and the last layer finally outputs the probability of classifying the tetrahedron as inside/outside.

3.3. Octree Based Scaling Strategy

The inconsistency of tetrahedrons presents a great challenge for the effective scaling of Delaunay triangulation-based methods. However, partitioning the nodes in the form of a dual graph provides a feasible solution. In our method, the structural features that we construct for nodes are specifically designed to be computed around the geometric center of the tetrahedrons. This means that the feature of one tetrahedron is aggregated on its geometric center, allowing us to instantiate our graph model, i.e., the location of one geometric center can be taken as the location of the corresponding node. Benefiting from the fact that our network is fully local, we employ an octree structure to divide the nodes and process the tetrahedrons represented by the nodes inside the same leaf in batches. We set a maximum threshold for the number of nodes inside each voxel of the octree, which can be adjusted according to the size of the device's video memory. The division process is illustrated in Figure 4. To ensure feature integrity, the neighborhood point information of all tetrahedrons and triangles is pre-computed before dividing. Such a scaling strategy allows our DMNet to be capable of handling points on the order of millions.

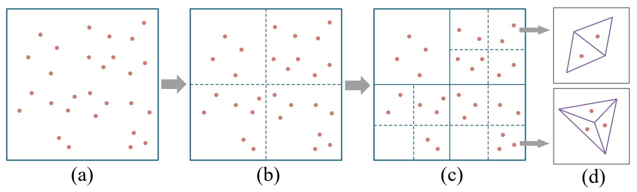

Figure 4: 2D example of an octree dividing the nodes of graph $\mathcal{G}$. (a) Setting the cubic bounding box for all nodes of $\mathcal{G}$. (b and c) Performing division operations. Voxels that contain no nodes of $\mathcal{G}$ or have a number of nodes less than the given threshold are retained, and the division continues for the remaining voxels until the number of nodes in all voxels is less than the threshold. These voxels are referred to as leaves of the octree. (d) Tetrahedrons corresponding to the nodes of $\mathcal{G}$ that are contained within one leaf are grouped into one batch of data.

3.4. Loss Functions

Multi-label Supervision We use a multi-label supervision loss $\mathcal{L}_m$ and a neighborhood smoothing loss $\mathcal{L}_n$ [38] as Equation (12-13) to supervise the labeling of tetrahedrons. We randomly sample N_{ref} reference locations inside each tetrahedron and get their inside/outside labels. $\mathcal{C}_{ij}$ is the cross-entropy loss between the prediction probabilities of a tetrahedron T_i and its j-th sampling label of reference location. $\mathcal{R}_{ij}$ is the cross-entropy between the predicted probabilities of T_i and its j-th neighbor tetrahedron T_i^j. N is the number of tetrahedrons.

$$\mathcal{L}_m = \frac{1}{N \times N_{ref}} \sum_{i=1}^{N} \sum_{j=1}^{N_{ref}} \mathcal{C}_{ij} \tag{12}$$

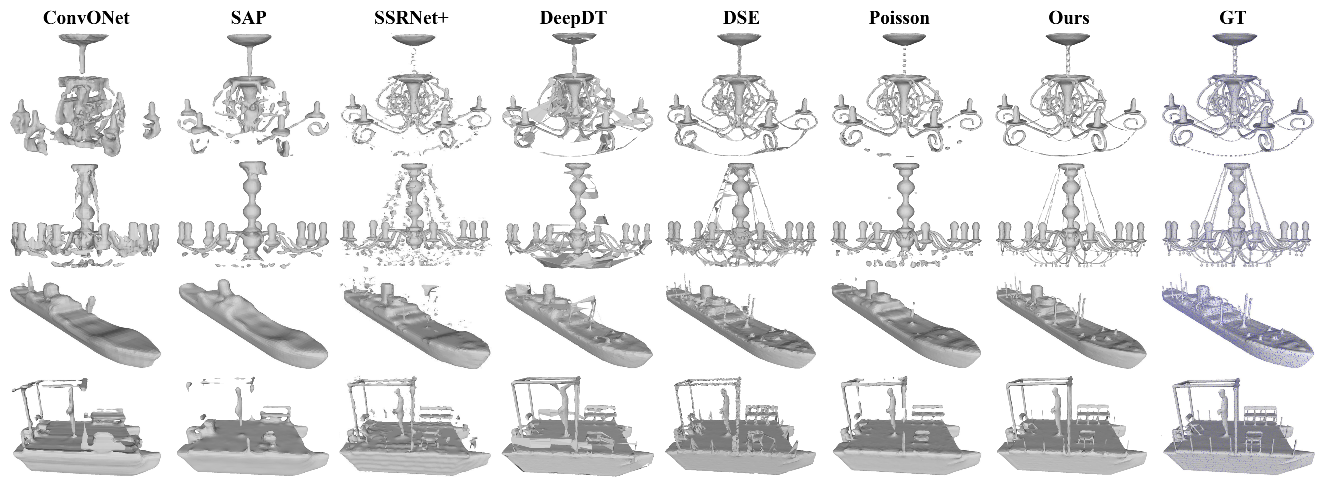

Figure 5: Qualitative comparison on ShapeNet. It can be seen that our method has an outstanding ability to preserve fine-grained details, yielding a better visual representation compared to other methods.

$$\mathcal{L}_n = \frac{1}{4N}\sum_{i=1}^{N}\sum_{j=1}^{4}\mathcal{R}_{ij} \tag{13}$$

Shape Constraint In order to obtain higher-quality meshes, we propose a geometric constraint loss, making our final extracted triangles as uniform in shape as possible and avoiding creating large artifacts. Specifically, the extraction probability $\mathcal{P}_i$ of one triangle $S_i\,(i = 1, .., M)$ can be transformed by the predicted probabilities $\mathcal{P}_i^{T1}$ and $\mathcal{P}_i^{T2}$ of its two neighboring tetrahedrons in a differentiable manner as Equation (14). Then we perform a softmax operation on the extraction probabilities of triangles, using the results as a weight of the longest side length to constrain shape. This loss function is advantageous for the classification of large tetrahedrons, ultimately leading to an improvement in reconstruction accuracy.

$$\mathcal{P}_i = \mathcal{P}_i^{T1}\left(1 - \mathcal{P}_i^{T2}\right) + \mathcal{P}_i^{T2}\left(1 - \mathcal{P}_i^{T1}\right) \tag{14}$$

$$\mathcal{L}_s = \frac{\sum_{i=1}^{M} Softmax(\mathcal{P}_i) \times \mathcal{E}_i}{\overline{\mathcal{E}}} \tag{15}$$

where $\mathcal{E}_i$ is the length of the longest side of S_i, and $\overline{\mathcal{E}}$ is the average length of $\mathcal{E}_i$.

Then our total loss function can be summarized as:

$$\mathcal{L} = \lambda_1\mathcal{L}_m + \lambda_2\mathcal{L}_n + \lambda_3\mathcal{L}_s \tag{16}$$

4. Experiments

4.1. Setup

Dataset and Metric We use three datasets to evaluate our method from different perspectives. The first is ShapeNet [10]. Following ConvONet [50], we use a subset and select the same four metrics including Normal Consistency (NC), Chamfer-L_1 Distance (C-L_1), F-Score (F), and IoU for evaluation. For the threshold of F-Score, we use 1% and 0.3% to make some comparisons more obvious. Additionally, we add the Percentage of Watertight Edges (WE) and the Average Number of Triangles (N.T) to evaluate the watertightness of meshes and the number of triangles contained in meshes, respectively. The second is FAMOUSTHINGI dataset [51] and we use it to perform generalization experiments. Following DSE [51], we use Chamfer Distance (CD), Normal Reconstruction Error (NR) and WE as metrics. Since models in this dataset have a large number of sharp edges, we additionally add Edge Chamfer Distance (ECD) and Edge F-Score (EFS) [13] to evaluate the preservation of sharp edges. The last is DTU [24]. Following SSRNet [43], we use CD, DTU Accuracy, and DTU Completeness [24] as metrics. A detailed description can be found in the supplementary.

Method	C-L_1 ↓	F(1%) ↑	F(0.3%) ↑	NC ↑	WE ↑	IoU ↑	N.T(k)
Alpha shapes [16]	0.058	0.835	0.601	0.880	0.991	-	15.32
Ball pivoting [5]	0.026	0.965	0.754	0.954	0.879	-	17.57
ONet [42]	0.077	0.813	0.338	0.918	**1.0**	0.803	59.91
ConvONet [50]	0.052	0.942	0.368	0.934	**1.0**	0.848	55.95
PointTriNet [52]	0.0215	0.9975	0.783	0.965	0.910	-	19.19
IER Meshing [35]	0.028	0.977	0.746	0.954	0.937	-	20.03
SAP [49]	0.031	0.981	0.675	0.950	**1.0**	0.930	45.54
NDC [11]	0.047	0.947	0.696	0.954	0.992	-	65.79
SSRNet+ [61]	0.030	0.983	0.571	0.958	0.989	-	59.53
DSE [51]	0.0215	0.9968	0.786	0.962	0.986	-	19.99
DeepDT [38]	0.027	0.973	0.747	0.948	0.983	-	18.50
Poisson [27]	0.023	0.9942	0.775	0.966	**1.0**	0.970	38.57
Ours	**0.0206**	**0.9983**	**0.804**	**0.973**	**1.0**	**0.983**	20.24

Table 1: Quantitative comparison on ShapeNet. '-' means that the data is not evaluated due to the limitation of watertight meshes.

	$5K$						$3K$						$1K$					
Method	C-$L_1\downarrow$	F(1%)$\uparrow$	NC$\uparrow$	WE$\uparrow$	IoU$\uparrow$	N.T(k)	C-$L_1\downarrow$	F(1%)$\uparrow$	NC$\uparrow$	WE$\uparrow$	IoU$\uparrow$	N.T(k)	C-$L_1\downarrow$	F(1%)$\uparrow$	NC$\uparrow$	WE$\uparrow$	IoU$\uparrow$	N.T(k)
Alpha shapes [16]	0.065	0.824	0.873	0.990	-	7.93	0.072	0.817	0.865	0.988	-	4.93	0.063	0.810	0.818	0.975	-	1.75
Ball pivoting [5]	0.031	0.947	0.934	0.877	-	8.81	0.036	0.929	0.915	0.877	-	5.31	0.052	0.862	0.872	0.878	-	1.78
ONet [42]	0.079	0.812	0.909	**1.0**	0.797	59.50	0.080	0.811	0.905	**1.0**	0.791	59.07	0.082	0.799	0.892	**1.0**	0.773	57.60
ConvONet [50]	0.052	0.943	0.934	**1.0**	0.849	55.45	0.052	0.940	0.933	**1.0**	0.846	54.92	0.063	0.900	0.917	**1.0**	0.810	53.39
PointTriNet [52]	0.024	0.992	0.952	0.903	-	9.59	0.027	0.981	0.937	0.891	-	5.75	**0.042**	0.906	0.886	0.845	-	1.89
IER Meshing [35]	0.036	0.957	0.900	0.800	-	12.43	0.043	0.925	0.875	0.717	-	9.29	0.063	0.810	0.803	0.737	-	5.50
SAP [49]	0.032	0.977	0.947	**1.0**	0.924	45.73	0.034	0.966	0.940	**1.0**	0.910	46.18	0.069	0.824	0.851	**1.0**	0.727	47.18
NDC [11]	0.049	0.943	0.948	0.986	-	64.92	0.053	0.933	0.938	0.974	-	60.97	0.076	0.822	0.894	0.892	-	34.48
SSRNet+ [61]	0.031	0.982	0.955	0.974	-	57.07	0.032	0.977	0.951	0.953	-	55.02	0.045	0.897	**0.919**	0.851	-	33.74
DSE [51]	0.024	0.991	0.947	0.974	-	9.99	0.027	0.980	0.930	0.957	-	5.99	0.044	0.898	0.894	0.901	-	1.98
DeepDT [38]	0.030	0.966	0.940	0.980	-	9.46	0.034	0.956	0.931	0.978	-	5.69	0.058	0.896	0.893	0.973	-	1.88
Poisson [27]	0.026	0.987	0.956	**1.0**	0.955	19.94	0.031	0.972	0.946	**1.0**	0.935	13.51	0.056	0.875	0.901	**1.0**	0.849	4.59
Ours	**0.022**	**0.996**	**0.965**	**1.0**	**0.977**	10.04	**0.024**	**0.990**	**0.956**	**1.0**	**0.967**	6.02	**0.042**	**0.937**	0.914	**1.0**	**0.912**	2.00

Table 2: Quantitative comparison for data with different densities on ShapeNet. '-' means that the data is not evaluated due to the limitation of watertightness.

Implementation Details Unless otherwise stated, the number of reference locations (N_{ref}) in our approach is set to 3, and the λ_1, λ_2, and λ_3 in loss function are set to 0.9, 0.1 and 1, respectively. In our point encoder, the number of nearest neighbor points searched is set to 32. The learning rate is set to 0.001 and decreases tenfold after 5 epochs.

4.2. Results

Reconstruction of Details ShapeNet contains a rich set of models, and we sample $10K$ points uniformly on each of them. We retrain all learning-based methods, but for ONet and ConvONet, we use their given models because we cannot train a better one.

Table 1 presents the quantitative evaluation results. As an advantage of explicit methods respecting the input points, they typically yield superior results in distance metrics and produce fewer triangles compared to methods based on implicit representation. However, the inability to guarantee a watertight mesh remains a significant limitation. In contrast, our DMNet well inherits the advantages of both explicit methods and Delaunay triangulation, achieving the best results across all performance metrics and ensuring watertight output meshes. The highest IoU value and the best distance values represent the superior ability of our method to reconstruct details.

Figure 5 presents a qualitative comparison of our method with some state-of-the-art methods. Implicit representation based methods, such as Poisson, ConvONet, SAP and SSRNet+, show great shortcomings in perceiving thin structures, while explicit methods tend to better preserve details. However, a significant number of artifacts are also created by these explicit methods. Due to the limited local shape perception, the lines generated by DSE are typically open structures composed of discrete triangles, rather than closed columnar structures. For similar reasons, DeepDT generates many misclassifications of tetrahedrons in the details, resulting in numerous visually corrupting artifacts. In contrast, our approach addresses these drawbacks by focusing on the learning of local structural representation. This allows our approach to robustly distinguish geometric information between tetrahedrons inside and outside the thin structures. Meaningfully, our approach learns to pick some narrow acute triangles to aid in reconstructing smooth columnar structures and edges. As a result, our method reconstructs more complete models and smoother lines without any smoothing post-processing. Furthermore, our method has a fast running time (3.64s), which is one-ninth of DSE (32.8s), and also achieves the best results for non-uniformly sampled data. These are shown in the supplementary.

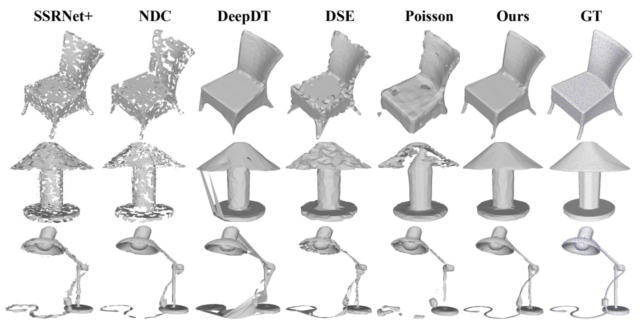

Figure 6: Comparison on sparse $1K$ points. Our results performs better reconstruction accuracy and completeness.

Compatibility with Data of Various Density Adapting point clouds of varying densities is also a great challenge for a learning-based algorithm, especially for sparse point clouds. We sample $5K$, $3K$, and $1K$ points on ShapeNet to evaluate the compatibility with point density variations. Interestingly, all learning-based methods are trained on $10K$ sampled points without retraining in the comparison, which makes it difficult to challenge classical geometric algorithms. Table 2 shows the quantitative comparison result, where our method achieves the best results in almost every

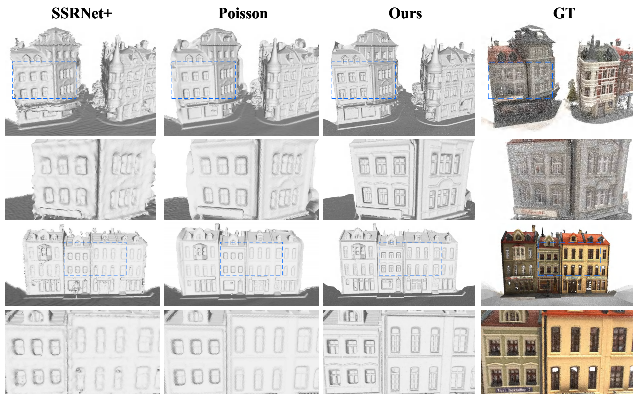

Figure 7: Qualitative comparison results on DTU. For one scene, we show two rows of results, where the lower row is the clear local result of the upper row.

Method	CD ↓	NR ↓	ECD ↓	EFS ↑	WE ↑
Alpha shapes [16]	1.064	17.69	0.854	0.153	0.983
Ball pivoting [5]	0.524	6.59	0.746	0.289	0.743
ConvONet [50]	2.436	24.13	2.210	0.004	**1.0**
PointTriNet [52]	0.331	5.54	0.443	0.338	0.917
IER Meshing [35]	0.343	6.30	0.445	0.335	0.947
SSRNet+ [61]	0.438	9.95	1.105	0.027	0.978
DSE [51]	**0.329**	5.34	0.726	0.342	0.996
DeepDT [38]	0.554	8.61	0.566	0.235	0.972
Poisson [27]	0.347	7.44	5.357	0.015	**1.0**
Ours	0.338	**4.74**	**0.270**	**0.386**	**1.0**

Table 3: Quantitative comparison on FAMOUSTHINGI. The units of CD and ECD are both 10^{-2}.

	DA ↓		DC ↓		CD ↓	
Method	Mean	Var.	Mean	Var.	Mean	RMS
L-Method [31]	0.524	1.612	0.377	0.596	0.745	35.840
DeepDT [38]	0.734	0.634	0.383	0.207	0.293	10.277
SSRNet [43]	0.321	0.285	0.304	0.0888	1.46	4.42
SSRNet+ [61]	0.310	0.236	0.335	0.105	1.29	3.97
Poisson [27]	0.330	0.441	0.345	0.438	1.17	4.49
Ours	**0.272**	**0.114**	**0.259**	**0.054**	**0.025**	**0.363**

Table 4: Quantitative results on DTU. DA = DTU Accuracy, DC = DTU Completeness. CD is in units of 10^{-6}.

metric. The qualitative results on $1K$ points are shown in Figure 6. It can be seen that our DMNet is able to reconstruct a structurally complete model even for sparse data. This demonstrates an outstanding ability to adapt to point density variations, which is difficult to achieve even for geometric methods. More meaningfully, the reconstructed smooth surfaces show that explicit methods have the potential to produce high-quality meshes without requiring any smoothing post-processing, even if very few triangles are used for sparse data (see N.T in Table 2).

Generalization Capability and Edge reconstruction We generalize various methods over FAMOUSTHINGI dataset to further investigate the ability of preserving sharp edges, especially for unknown shapes. We sample $10K$ points uniformly for each model and all learning-based methods are trained on ShapeNet. The comparison results are shown in Table 3 and Figure 1. Satisfactorily, our method exhibits an outstanding performance in preserving edge sharpness, as demonstrated by ECD and EFS values that significantly outperform those of other methods. While Poisson, SSRNet+ and ConvONet exhibit a significant gap with the explicit methods. The inherent smoothness of these implicit methods makes it challenging for them to retain sharpness. More visual comparisons are shown in the supplementary.

Challenging Large-scale Real Data We evaluate the scalability of our method on the DTU dataset, where each scene contains millions of scanned points. We use the first 30 scenes for training, 10 scenes for validation, and the rest for testing. The maximum capacity of an octree leaf in our method is set to $60K$ tetrahedrons. We impose a slight Laplacian smoothing [56] like SSRNet and SSRNet+ in this experiment, which does not increase the details our method preserves, simply to get a better visual effect. We still add DeepDT as a comparison, although it has no scalability. Therefore, for consistency with its paper [38], each point cloud is downsampled to $200K$ points for its training and testing. Table 4 reports the quantitative results, where our method outperforms SSRNet and SSRNet+, the state-of-the-art learning-based methods we know for processing large-scale data. Figure 7 shows the qualitative results. Benefiting from our scaling strategy, our method exhibits exceptional performance in handling scans with millions of points, retaining rich fine-grained details. In contrast, Poisson presents an overly smooth surface that results in loss of many details, such as columns on the windows, due to the inherent smoothness of implicit function. Our excellent results on large-scale real data further illustrate that our method well explores the reconstruction potential of Delaunay triangulation.

4.3. Analysis

Ablation Studies In this section, we evaluate the performance impact of each major module in our approach, as presented in Table 5. Firstly, we consider the performance improvement that the proposed local graph iteration algorithm brings in comparison to GCN [29]. Following the replacement, the performance of our method significantly declines, indicating that edge information in the graph is important and that node-edge interaction aids in effective aggregation of neighborhood information by nodes. Secondly, we construct multi-level geometric features for the graph by combining points, triangles, and tetrahedrons in a fully local

	w/o LGI w/ GCN	feature encoder*	w/o offset distance	w/o manual features	w/o shape constraint	base
C-$L_1 \downarrow$	0.0230	0.0252	0.0221	0.0214	0.0210	**0.0206**
F(0.3%) $\uparrow$	0.776	0.758	0.787	0.798	0.801	**0.804**

Table 5: Ablation experiments of our method on ShapeNet. LGI represents our Local Graph Iteration algorithm. * represents the module replacement with DeepDT.

manner. In contrast, DeepDT focuses on tangent plane features between points. To compare the two manners of graph feature encoding, we replace our method's feature encoder with that of DeepDT. The result shows an even greater performance decline, indicating that focusing on structural representation can capture more local geometric information and contribute to better tetrahedron classification. Thirdly, to assess the significance of the local coordinate transformation in our structural feature encoder, we replace the offset distances of vertices with their original coordinates. The resulting performance degradation demonstrates the importance of this transformation in effectively embedding vertex features into the structural representation. Finally, we remove the designed manual features in our structural encoder and the shape constraint loss, respectively. We observe that each of these changes leads to a degradation in performance.

Limitations We show the experiments for noisy data with two standard deviations including 0.5% and 1% in the supplementary. Although the results demonstrate that our method remains robust in maintaining model integrity in the face of noisy data, it inevitably requires a smoothing post-processing to produce better visual results when dealing with high-intensity noise. This is a common problem for explicit methods since the vertices of output meshes are derived from the input noisy points. It would be a good direction to combine mesh deformation in future work.

5. Conclusion

We propose a novel data-driven method with Delaunay triangulation of point clouds for surface reconstruction, giving an effective solution to preserve fine-grained details and deal with data of different densities. The special feature construction and Local Graph Iteration algorithm make our method have a powerful ability to capture local details. The scaling strategy allows our approach to be applied to the reconstruction of millions of points. Experiments demonstrate the effectiveness of our method in tackling some reconstruction challenges, such as thin structures, sharp edges and large-scale real data. We hope that our method could inspire more learning-based methods to explore the application value of Delaunay triangulation in surface reconstruction, contributing to high-quality reconstruction.

6. Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grants 62176096 and 61991412.

References

[1] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on visualization and computer graphics*, 9(1):3–15, 2003.

[2] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999.

[3] Tristan Aumentado-Armstrong, Stavros Tsogkas, Sven Dickinson, and Allan D Jepson. Representing 3d shapes with probabilistic directed distance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19343–19354, 2022.

[4] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, volume 36, pages 301–329. Wiley Online Library, 2017.

[5] 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.

[6] Jean-Daniel Boissonnat and Bernhard Geiger. Three-dimensional reconstruction of complex shapes based on the delaunay triangulation. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 964–975. International Society for Optics and Photonics, 1993.

[7] Ruud M. Bolle and Baba C. Vemuri. On three-dimensional surface reconstruction methods. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(01):1–13, 1991.

[8] 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.

[9] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, 2001.

[10] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

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

[12] 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.

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

[14] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Advances in Neural Information Processing Systems*, 33:21638–21652, 2020.

[15] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. On the effectiveness of weight-encoded neural implicit 3d shapes. *arXiv preprint arXiv:2009.09808*, 2020.

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

[17] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision*, pages 108–124. Springer, 2020.

[18] Fabien Evrard, Fabian Denner, and Berend Van Wachem. Surface reconstruction from discrete indicator functions. *IEEE Transactions on Visualization and Computer Graphics*, 25(3):1629–1635, 2018.

[19] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020.

[20] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164, 2019.

[21] 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.

[22] M Jancosek and T Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3121–3128, 2011.

[23] Michal Jancosek and Tomas Pajdla. Exploiting visibility information in surface reconstruction to preserve weakly supported surfaces. *International scholarly research notices*, 2014, 2014.

[24] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[25] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020.

[26] Michael Kazhdan. Reconstruction of solid models from oriented point sets. In *Proceedings of the third Eurographics symposium on Geometry processing*, pages 73–es, 2005.

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

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

[29] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[30] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F O'Brien. Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 11–21, 2004.

[31] Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007.

[32] Patrick Labatut, J-P Pons, and Renaud Keriven. Robust and efficient surface reconstruction from range data. In *Computer graphics forum*, volume 28, pages 2275–2290. Wiley Online Library, 2009.

[33] Shiwei Li, Yao Yao, Tian Fang, and Long Quan. Reconstructing thin structures of manifold surfaces by integrating spatial curves. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2887–2896, 2018.

[34] Seng Poh Lim and Habibollah Haron. Surface reconstruction techniques: a review. *Artificial Intelligence Review*, 42(1):59–78, 2014.

[35] 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.

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

[37] 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.

[38] Yiming Luo, Zhenxing Mi, and Wenbing Tao. Deepdt: Learning geometry from delaunay triangulation for surface reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2277–2285, 2021.

[39] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. *arXiv preprint arXiv:2011.13495*, 2020.

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

[41] Josiah Manson, Guergana Petrova, and Scott Schaefer. Streaming surface reconstruction using wavelets. In *Computer Graphics Forum*, volume 27, pages 1411–1420. Wiley Online Library, 2008.

[42] 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.

[43] Zhenxing Mi, Yiming Luo, and Wenbing Tao. Ssrnet: Scalable 3d surface reconstruction network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 970–979, 2020.

[44] Christian Mostegel and Markus Rumpler. Robust surface reconstruction from noisy point clouds using graph cuts. *Technical report*, 2012.

[45] Yukie Nagai, Yutaka Ohtake, and Hiromasa Suzuki. Smoothing of partition of unity implicit surfaces for noise robust surface reconstruction. In *Computer Graphics Forum*, volume 28, pages 1339–1348. Wiley Online Library, 2009.

[46] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. 3d scattered data interpolation and approximation with multilevel compactly supported rbfs. *Graphical Models*, 67(3):150–165, 2005.

[47] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer graphics forum*, volume 28, pages 493–501. Wiley Online Library, 2009.

[48] 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.

[49] 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.

[50] 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.

[51] 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.

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

[53] 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.

[54] Raphael Sulzer, Loic Landrieu, Renaud Marlet, and Bruno Vallet. Scalable surface reconstruction with delaunay-graph neural networks. In *Computer Graphics Forum*, volume 40, pages 157–167. Wiley Online Library, 2021.

[55] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[56] Gabriel Taubin. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, 1995.

[57] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, 1994.

[58] Greg Turk and James F O'brien. Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG)*, 21(4):855–873, 2002.

[59] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016.

[60] Cheng Chun You, Seng Poh Lim, Seng Chee Lim, Joi San Tan, Chen Kang Lee, and Yen Min Jasmina Khaw. A survey on surface reconstruction techniques for structured and unstructured data. In *2020 IEEE Conference on Open Systems (ICOS)*, pages 37–42. IEEE, 2020.

[61] Ganzhangqin Yuan, Qiancheng Fu, Zhenxing Mi, Yiming Luo, and Wenbing Tao. Ssrnet: Scalable 3d surface reconstruction network. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[62] Yang Zhou, Shuhan Shen, and Zhanyi Hu. Detail preserved surface reconstruction from point cloud. *Sensors*, 19(6):1278, 2019.