DCCVT: Differentiable Clipped Centroidal Voronoi Tessellation

Wylliam Cantin Charawi[1] Adrien Gruson[1] Jane Wu[2] Christian Desrosiers[1] Diego Thomas[3]
[1]École de Technologie Supérieure [2]UC Berkeley [3]Kyushu University

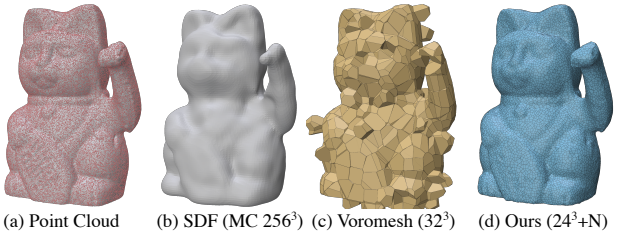
Abstract

While Marching Cubes (MC) and Marching Tetrahedra (MTet) are widely adopted in 3D reconstruction pipelines due to their simplicity and efficiency, their differentiable variants remain suboptimal for mesh extraction. This often limits the quality of 3D meshes reconstructed from point clouds or images in learning-based frameworks. In contrast, clipped CVTs offer stronger theoretical guarantees and yield higher-quality meshes. However, the lack of a differentiable formulation has prevented their integration into modern machine learning pipelines. To bridge this gap, we propose DCCVT, a differentiable algorithm that extracts high-quality 3D meshes from noisy signed distance fields (SDFs) using clipped CVTs. We derive a fully differentiable formulation for computing clipped CVTs and demonstrate its integration with deep learning-based SDF estimation to reconstruct accurate 3D meshes from input point clouds. Our experiments with synthetic data demonstrate the superior ability of DCCVT against state-of-the-art methods in mesh quality and reconstruction fidelity. https://wylliamcantincharawi.dev/DCCVT.github.io/

1. Introduction

High-quality three-dimensional content is essential across a wide range of fields, including AR/VR, digital simulation, healthcare, gaming, and filmmaking. Among the various 3D representations, discrete formats like 3D meshes are widely used due to their simplicity in rendering and manipulation. However, generating accurate 3D content from real-world data, such as 2D images or partial or noisy point clouds, is a highly challenging task that typically demands expert knowledge and considerable manual effort. As a result, extensive research has focused on assisting or fully automating this process. One common approach involves using implicit representations, such as Signed Distance Fields (SDFs), which are optimized from observations and later converted into discrete 3D meshes through a post-processing step. This method is effective because smooth SDFs are well-suited for optimization techniques.

In recent years, significant attention has been devoted to optimizing and regularizing SDFs from multi-view images or 3D point clouds. However, the final step of extracting a 3D mesh is typically handled by simple methods that are decoupled from the optimization process, often resulting in sub-optimal reconstructions. To address this, differentiable versions of Marching Cubes (DMC) [19] and Marching Tetrahedra (DMTet) [32] have been introduced, enabling the integration of mesh extraction directly into the optimization loop. These methods have shown improved mesh reconstruction quality when using SDFs or point clouds as input. Nevertheless, we argue that the underlying algorithms, Marching Cubes and Marching Tetrahedra, are themselves not optimal for high-quality mesh extraction, and thus their differentiable counterparts (DMC and DMTet) remain fundamentally limited.



(a) Point Cloud (b) SDF (MC 256^3) (c) Voromesh (32^3) (d) Ours (24^3+N)

Figure 1. Comparison of 3D mesh reconstruction from an unoriented point cloud. (a) Input point cloud. (b) Input Signed Distance Field (SDF) reconstructed using Marching Cubes at 256^3 resolution. (c) Voromesh baseline with 32^3 sites. (d) Our proposed DCCVT method with 24^3 sites plus near sampling sites, achieving more accurate and regular surface reconstruction.

Marching Cubes operates on a fixed grid where SDF values are sampled at the grid's vertices, and a triangular mesh is extracted along edges where the SDF crosses zero. While this method is fast and efficient, it often suffers from poor tessellation due to the fixed orientation of grid cells, leading to irregular triangles and approximation errors, particularly for complex geometries. As a result, post-processing is typically required to produce a more regular 3D mesh. On the other hand, Marching Tetrahedra offers greater flexibility by using Delaunay tetrahedralization, which allows the tessellation to better adapt to the shape of the implicit surface. However, irregularities in the mesh remain challenging to eliminate completely. Recent studies have shown that Centroidal Voronoi Tessellations (CVT) can produce volumetric and surface discretizations with significantly improved regularity [8]. In addition, CVTs provide provably more ac-

curate approximations of implicit shapes compared to both Marching Cubes and Marching Tetrahedra.

In this paper, we propose DCCVT, a differentiable algorithm for the clipped CVT (CCVT) algorithm [38]. CCVT has proven advantages compared to MC and MTet and we demonstrate that DCCVT also has significant advantages compared to DMC and DMTet. While our primary focus is on 3D mesh reconstruction from input point clouds, our core contribution of differentiable clipped CVT also has broader applicability to other 3D mesh reconstruction pipelines. Our contributions are:

- A Centroidal Voronoi Tesselation extraction from a sparse point cloud;
- An expressive end-to-end optimization of Voronoi site position and their associated SDF values through a clipping operation;
- An iterative upsampling method improving the overall robustness of our method.

2. Related Work

Explicit methods.

Explicit methods directly connect sample points to form a mesh. A representative example is the Ball Pivoting Algorithm (BPA) introduced by Bernardini et al. [4], which forms a triangle whenever a ball of a user-specified radius can touch three points without enclosing any others. This simple heuristic yields watertight meshes and remains widely used in practice. However, BPA suffers from notable limitations, including sensitivity to noise, non-uniform sampling densities, and the need to tune a single ball radius that may not suit all surface regions. In contrast, Poisson Surface Reconstruction [14, 15] formulates mesh recovery as solving a well-conditioned sparse linear system using local basis functions. It does not rely on radius heuristics and is generally more robust to noise, but requires oriented point clouds, which are often unavailable in raw scans. Normal estimation algorithms [1, 2, 11, 17] could be used to estimate point sample orientations, but existing methods still yield noisy predictions.

Deep learning approaches to point set triangulation have also been proposed [22, 25, 31, 35, 44, 45]. DeepDT [22] reconstructs surfaces by predicting in/out labels of tetrahedrons directly from the point cloud and its corresponding Delaunay triangulation. PointTriNet [31] proposes candidate triangles and another classifies whether each triangle should appear in the final triangulation. DMNet [45] and followup work in Zhang and Tao [44] models the Delaunay triangulation as a dual graph and embeds local geometric features into its structure.

Implicit methods. A number of learning-based approaches to surface reconstruction involve predicting a continuous SDF of 3D surfaces directly from unstructured point sets [6, 7, 9, 10, 13, 18, 23, 27, 34, 39, 40, 46]. One of the earliest works in this area was DeepSDF [27], which introduced a latent-conditioned neural network that learns a continuous SDF for an entire shape class, enabling high-quality shape representation, interpolation, and completion from partial/noisy data. Points2Surf [9] presented a patch-based framework that learns an implicit surface from raw point clouds without normals by combining detailed local patches with coarse global SDF sign predictions; this yields more accurate reconstructions and better generalization to unseen shapes. NeuralTPS [6] infers an SDF from a single sparse point cloud without any learned shape priors or normal inputs by leveraging thin-plate-spline surface parameterizations to generate coarse surface samples during training. Zhang et al. [46] proposes a two-stage approach that combines SDF learning with adaptive Delaunay meshing algorithm. The recent Hotspot method [40] can infer a neural SDF within minutes. Besides SDF representations, a number of neural implicit representations have also been proposed for surface reconstruction from point clouds [3, 12, 20, 26, 29, 33, 36, 37, 41, 43].

While implicit representations can capture smooth and accurate surfaces, detail preservation often suffers when applying standard mesh extraction methods such as Marching Cubes or Marching Tetrahedra. Consequently, current research is focusing on integrating mesh extraction into the learning process to preserve fidelity, with differentiable extraction emerging as a key approach.

Differentiable mesh extraction. Liao et al. [19] introduced Deep Marching Cubes, a differentiable variant of Marching Cubes, enabling end-to-end training from point clouds. However, the approach is limited by its requirement for an axis-aligned voxel grid, which constrains detail resolution. To address this, DMTet [32] replaces the grid with a tetrahedral discretization, removing axis alignment constraints and enabling finer reconstructions. Similarly, Wu et al. [42] proposed a differentiable Marching Tetrahedra variant with true gradients for sparse-view 3D human body reconstruction. More recently, Maruani et al. [24] proposed Voromesh, leveraging the regularity of Voronoi diagrams to optimize space tessellation for improved mesh quality. However, most existing differentiable methods treat grid optimization and SDF optimization as separate stages, potentially limiting the optimality of the final reconstruction.

3. Preliminaries

A set of N 3D sites $\{\mathbf{s}_i\}_{i=1}^{N} \subset \mathbb{R}^3$ implicitly represents a tessellation of the 3D space into Voronoi cells $\mathcal{V}(\mathbf{s}_i)$ defined as:

$$\mathcal{V}(\mathbf{s}_i) = \{\mathbf{p} \in \mathbb{R}^3 \mid \|\mathbf{p} - \mathbf{s}_i\| < \|\mathbf{p} - \mathbf{s}_j\|,\ \forall j \neq i\}.$$

The Voronoi diagram is also the dual of the Delaunay tetrahedralization $\mathcal{T} = \{\mathcal{T}_m\}_{m=1}^{M}$.

Each tetrahedron $\mathcal{T}_m \subset \mathcal{T}$ with corresponding sites $\{\mathbf{s}_a, \mathbf{s}_b, \mathbf{s}_c, \mathbf{s}_d\}$ defines one Voronoi vertex $\mathbf{v}_m$ as its circumcenter, which is computed as

$$\mathbf{v}_m = \mathbf{s}_a + \frac{\alpha\,(\mathbf{q}\times\mathbf{r}) + \beta\,(\mathbf{r}\times\mathbf{p}) + \gamma\,(\mathbf{p}\times\mathbf{q})}{2\,\mathbf{p}\cdot(\mathbf{q}\times\mathbf{r})}, \quad (1)$$

where $\mathbf{p} = \mathbf{s}_b - \mathbf{s}_a$, $\mathbf{q} = \mathbf{s}_c - \mathbf{s}_a$, $\mathbf{r} = \mathbf{s}_d - \mathbf{s}_a$ are the local edges, and α, β, γ are their squared norms. The denominator is 2 times the scalar triple product, i.e., $12\,V_{\text{signed}}$, with the tetrahedron volume $V = |\mathbf{p}\cdot(\mathbf{q}\times\mathbf{r})|/6$.

Voronoi cell centroid. In general, a Voronoi diagram initialized with random sites produces an irregular discretization, where sites often form elongated tetrahedra and slivers[1], which can cause significant issues during 3D mesh extraction. To improve the regularity of the tessellation, minimizing the Centroidal Voronoi Tessellation (CVT) energy is a widely used and robust strategy. A Voronoi diagram is said to be in a CVT configuration when all sites coincide with the centroids of their corresponding Voronoi cells. This configuration can be enforced using the Lloyd algorithm [21].

4. Proposed method

We address the problem of reconstructing an accurate, watertight 3D mesh from an unoriented point cloud (Fig. 2a). The surface is modeled as an implicit SDF discretized on a 3D grid (Fig. 2b), from which the mesh is extracted. Unlike prior work that optimizes only the SDF or the sampled site positions, we jointly optimize both the site positions $\{\mathbf{s}_i\}_{i=1}^N$ and their associated SDF values $\{\phi_i\}_{i=1}^N$. Our approach comprises: (1) projecting 0-crossing Voronoi vertices $\{\mathbf{v}_j\}_{j=1}^M$ onto the SDF zero-level set via a robust projection scheme (Sec. 4.1); (2) regularizing the SDF-aware Voronoi diagram to satisfy the Centroidal Voronoi Tessellation (CVT) property (Sec. 4.2); and (3) adaptively refining the discretization through error-driven site insertion (Sec. 4.3). The optimization is fully differentiable, facilitating integration with modern machine learning frameworks. Figure 2 gives an overview of our proposed method. All diagrams are presented in 2D for simplicity.

4.1. Joint optimization

The central idea of our method is to transform the conventional Voronoi diagram into a surface-aware Voronoi diagram, in which Voronoi vertices generated by tetrahedrons intersecting the SDF zero level are positioned close to the vertices of the final 3D mesh extracted by clipping the Voronoi cells. By formulating both the CVT regularization and the vertex projection onto the SDF zero level in a differentiable manner, we jointly optimize the SDF values and the discretization in an end-to-end framework, enabling loss functions to be applied directly to the vertices of the output 3D mesh.

In the remainder of this section, we restrict our attention to tetrahedrons that intersect the zero-level set of the SDF defined as $\mathcal{T}^{(0)}$. They are the elements that determine the vertices of the output 3D mesh. A tetrahedron $\mathcal{T}_j$, defined by the four sites $\{\mathbf{s}_a, \mathbf{s}_b, \mathbf{s}_c, \mathbf{s}_d\}$, is classified as intersecting if the SDF values at its summits satisfy

$$\min(\phi_a, \phi_b, \phi_c, \phi_d) < 0 < \max(\phi_a, \phi_b, \phi_c, \phi_d)\,. \quad (2)$$

For each such tetrahedron, we compute the corresponding Voronoi vertex $\mathbf{v}_j$ using Eq. (1), and subsequently project it onto the SDF zero-level. Likewise, the midpoints of all crossing edges are determined and projected onto the zero-level set. The data loss is then evaluated as the distance between these projected points and the input target points. By expressing the projection process in a differentiable form with respect to both the SDF values and the site positions, we enable joint optimization within a unified framework.

Projection of 0-crossing Voronoi vertices.

Figure 2 (c) illustrates the projection of Voronoi vertices onto the zero-level set of the SDF. Each Voronoi vertex is defined as the circumcenter of its corresponding tetrahedron. However, this circumcenter can sometimes lie far from the tetrahedron itself, making straightforward barycentric interpolation of SDF values and gradients unreliable. Moreover, explicitly searching for the tetrahedron that contains a given Voronoi vertex is computationally prohibitive. To address these issues, we introduce an efficient and robust projection strategy that remains effective even under challenging tetrahedral configurations. Specifically, for each tetrahedron we fit a plane to the zero-level projections of its four summits and project the corresponding Voronoi vertex onto it.

For each $\mathcal{T}_i$ that crosses the zero level set, we first project its associated sites $\mathbf{s}_i'$ onto the zero-level set as

$$\mathbf{s}_i' = \mathbf{s}_i - \frac{\nabla\phi_i}{\|\nabla\phi_i\| + \epsilon}\,\phi_i\,. \quad (3)$$

We then compute the centroid of the projected sites as

$$\bar{\mathbf{s}}_j' = \frac{1}{4}\sum_{i\in\mathcal{T}_j} \mathbf{s}_i', \quad (4)$$

We find the plane, defined by normal vector $\mathbf{n}_j$, which passes by the site centroid $\bar{\mathbf{s}}_j'$ and whose average distance to projected sites $\mathbf{s}_i'$ is minimum. Denoting the centered coordinates as $\Delta\mathbf{s}_i' = \mathbf{s}_i' - \bar{\mathbf{s}}_j'$, this can be done by optimizing the following problem:

$$\arg\min_{\mathbf{n}_j} \frac{1}{4}\sum_{i\in\mathcal{T}_j} \Big(\underbrace{\frac{\mathbf{n}_j}{\|\mathbf{n}_j\|}\cdot\Delta\mathbf{s}_i'}_{\text{distance to plane}}\Big)^2. \quad (5)$$

[1] Flattened and elongated Voronoi cells

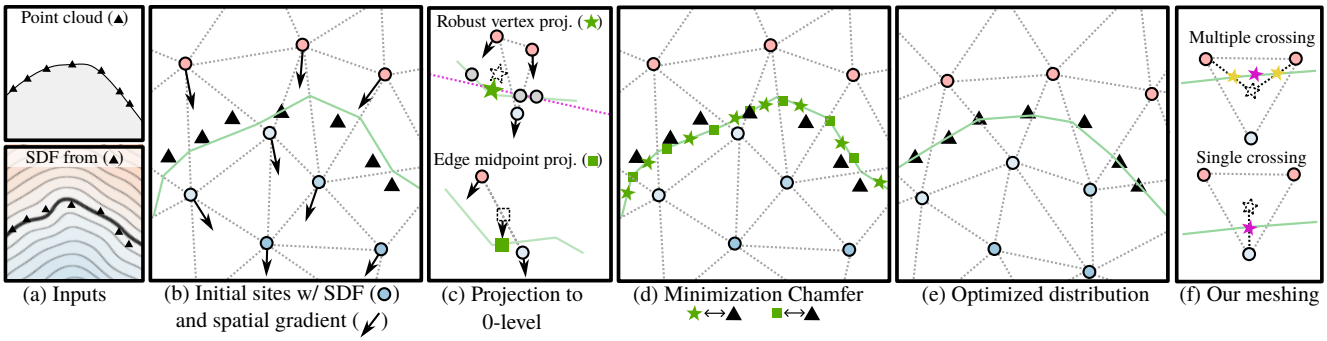


Figure 2. Overview of our proposed method. Starting from an unoriented 3D point cloud and an inferred SDF estimation from the point cloud (a), we initialize Voronoi sites (b), which yield a discretized SDF (zero-level shown in green). We then project zero-crossing Voronoi vertices (green stars) and edge midpoints (green squares) onto the SDF zero-level and minimize the Chamfer distance to the input point cloud (c–d). This optimization produces an improved site distribution and SDF representation (e), from which we extract the final mesh using our Voronoi-based meshing strategy (f), with Voronoi vertices shown in purple.

This normal vector $\mathbf{n}_j$, which defines the orientation of the fitted plane passing through the centroid $\mathbf{c}_j$, corresponds to the first eigenvector[2] of the centered site covariance matrix. Finally, we project the Voronoi vertex $\mathbf{v}_j$ onto this fitted plane as

$$\mathbf{v}'_j \;=\; \mathbf{v}_j - [(\mathbf{v}_j - \mathbf{c}_j)\cdot \mathbf{n}_j]\;\mathbf{n}_j\,. \tag{6}$$

This projection ensures that the Voronoi vertex lies on the zero-level set of the SDF using only site-based information. See supplemental material for the full derivation (**??**) and comparative experiments with using a naive Newton projection.

Projection of 0-crossing edge midpoints. In addition to the Voronoi vertices, we compute the midpoints of tetrahedrons edges that cross the 0 level-set (Fig. 2c). These edge midpoints provide an additional constraint, ensuring that the Voronoi faces remain well aligned with both our SDF representation and the target point cloud. For each edge corresponding to a pair of adjacent sites $(\mathbf{s}_i, \mathbf{s}_j)$ that crosses the zero-level set, we approximate the SDF value and the gradient vector of the midpoint $\mathbf{b}_{ij}$ as follows.

$$\phi(\mathbf{b}_{ij}) \approx \tfrac{1}{2}(\phi_i + \phi_j),\;\; \nabla\phi(\mathbf{b}_{ij}) \approx \tfrac{1}{2}(\nabla\phi_i + \nabla\phi_j) \tag{7}$$

Finally, we project the edge midpoint $\mathbf{b}_{ij}$ onto the zero-level set of the SDF using the Newton step method (Eq. (8)).

$$\mathbf{b}'_{ij} \;=\; \mathbf{b}_{ij} - \frac{\nabla\phi(\mathbf{b}_{ij})}{\|\nabla\phi(\mathbf{b}_{ij})\| + \epsilon}\;\phi(\mathbf{b}_{ij})\,, \tag{8}$$

Spatial SDF gradient. To compute the spatial gradient of the SDF at each site, we assume a constant gradient within each tetrahedron, denoted by $\nabla\phi^{(j)}$. Using a first-order approximation, the SDF at site $i \in \mathcal{T}_j$ is estimated as

$$\widehat{\phi}_i \;\approx\; \bar{\phi}_j \;+\; (\mathbf{s}_i - \bar{\mathbf{s}}_j)^\top \nabla\phi^{(j)}. \tag{9}$$

[2]Eigenvector corresponding to the smallest eigenvalue.

The gradient $\nabla\phi^{(j)}$ is then obtained by minimizing the mean squared difference between the actual and estimated SDF values over all sites:

$$\arg\min_{\nabla\phi^{(j)}} \;\frac{1}{4}\sum_{i\in\mathcal{T}_j}\left(\widehat{\phi}_i - \phi_i\right)^2. \tag{10}$$

As detailed in **??** inside the supplementary materials, this can be done efficiently by solving a linear system.

Then, the spatial gradient at site $\mathbf{s}_i$ is obtained as the volume-weighted average of the gradients of its incident tetrahedron:

$$\nabla\phi_i \;=\; \frac{1}{\sum_{j\in\mathcal{D}_\mathbf{s}} V_j}\sum_{j\in\mathcal{D}_\mathbf{s}} V_j \nabla\phi^{(j)}, \tag{11}$$

where $\mathcal{D}_\mathbf{s}$ is the set of tetrahedrons adjacent to $\mathbf{s}$, V_j is the volume of tetrahedron j, and $\nabla\phi^{(j)}$ is its local gradient defined in **??**.

4.2. Regularization

The data loss guiding our optimization is the Chamfer distance between the original point cloud and the set of reconstructed vertices and edge midpoints, defined as

$$\mathcal{L}_{\text{CD}} \;=\; \min_{\text{dist}}\left(\{\|\mathbf{p}-\mathbf{v}_i\|^2, \|\mathbf{p}-\mathbf{b}_{ij}\|^2\}\right) \tag{12}$$

where $\mathbf{v}_i$ are Voronoi vertices and $\mathbf{b}_{ij}$ are edge midpoints. This loss encourages both the Voronoi vertices and edges to lie close to the input point cloud, ensuring that the extracted mesh accurately represents the input data. In addition to this main loss, we introduce several regularization losses to promote well-distributed sites and smooth SDF variation.

CVT Regularization. A surface-aware Centroidal Voronoi Tessellation (CVT) loss is computed using vertices *projected* onto the zero-level set (Eq. (6)). Vertices from tetrahedrons that do not intersect the zero-level set are left unchanged. During optimization, a site might deviate from

this centroid, resulting in irregular Voronoi cells. To encourage regularity, we define a CVT loss that penalizes the Euclidean distance of each site $\mathbf{s}_i$ from its centroid $\mathbf{c}_i$ as

$$\mathcal{L}_{\text{CVT}} = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{s}_i - \mathbf{c}_i\| \,. \tag{13}$$

Eikonal Regularization. A fundamental property of SDFs is that they satisfy the Eikonal equation, which requires the gradient of the SDF to have unit norm at every point. This property is crucial for ensuring a consistent distance field and is extensively used in our multiple projection steps. To regularize the SDF gradient, we adapt the definition by Wu et al. [42] and define our Eikonal loss as

$$\mathcal{L}_{\text{Eik}} = \frac{1}{4M}\sum_{j=1}^{M}\sum_{i=1}^{4} V_j \left(\left\|\nabla\phi_{j,i}\right\|^2 - 1\right)^2, \tag{14}$$

where M is the number of tetrahedrons and $\nabla\phi_{j,i}$ is the gradient of the SDF at summit i of tetrahedron j. This encourages $\|\nabla\phi\| \approx 1$ throughout the domain, preserving the SDF property in a volume-weighted discrete form.

Motion by mean curvature (MbMC) Regularization. As shown by Wu et al. [42], the SDF can be regularized by incorporating a motion by mean curvature (MbMC) term, which encourages the SDF to evolve according to its mean curvature. This can be achieved by adding a term to the loss function that penalizes deviations from the mean curvature flow. For this they define a smeared-out Heaviside function H as

$$H(\hat{\phi}) = \begin{cases} 0, & \hat{\phi} < -\epsilon_H \\ \frac{1}{2} + \frac{\hat{\phi}}{2\epsilon_H} + \frac{1}{2\pi}\sin\left(\frac{\pi\hat{\phi}}{\epsilon_H}\right), & -\epsilon_H \le \hat{\phi} \le \epsilon_H \\ 1, & \hat{\phi} > \epsilon_H \end{cases} \tag{15}$$

where ϵ_H is equal to the mean edge distance between sites, excluding the 5% longer edges. By reusing the weight matrix $\mathbf{W}_j$ defined in **??** when solving for the tetrahedrons gradients, we can compute the mean curvature at each tetrahedron as

$$\nabla\mathcal{H}^{(j)} = \mathbf{W}_j \begin{bmatrix} H(\phi_A) - \bar{H}_j \\ H(\phi_B) - \bar{H}_j \\ H(\phi_C) - \bar{H}_j \\ H(\phi_D) - \bar{H}_j \end{bmatrix}, \tag{16}$$

where $\bar{H}_j$ is the mean of the Heaviside function values at the four sites of tetrahedron $\mathcal{T}_j$, similarly to the mean SDF value defined in **??**. The MbMC loss is then defined as

$$\mathcal{L}_{\text{H}} = \frac{1}{M}\sum_{j=1}^{M} V_j \left\|\nabla\mathcal{H}^{(j)}\right\|, \tag{17}$$

where M is the number of tetrahedrons and V_j is the volume of tetrahedron $\mathcal{T}_j$.

Loss. The total loss of our optimization is

$$L = \mathcal{L}_{CD} + \lambda_{\text{CVT}}\mathcal{L}_{CVT} + \lambda_{\text{Eik}}\mathcal{L}_{Eik} + \lambda_{\text{H}}\mathcal{L}_{H}\,, \tag{18}$$

where $\lambda_{\text{CVT}} = 0.1, \lambda_{\text{Eik}} = 0.02, \lambda_{\text{H}} = 0.1$ are the weights of the respective losses.

4.3. Upsampling

One key aspect of our method is the ability to represent complex shapes with a limited number of Voronoi sites. Nevertheless, similar to prior approaches [24, 32], our proposed method achieves higher mesh quality when more active sites are available (i.e., sites that have tetrahedrons crossing the zero-level set of the SDF) to improve the mesh quality. To this end, we employ an adaptive upsampling strategy that refines the reconstruction domain by inserting new sites according to local SDF features. The decision to insert additional sites is guided by a scoring function based on local spacing and curvature.

Local Feature Computation. We compute local features for each Voronoi site $\mathbf{s}_i$ to guide the upsampling process. The first feature is the local spacing ρ_i, which measures the distance to the nearest neighbor site which is defined as

$$\rho_i = \min_{j\in\mathcal{N}(i)} \|\mathbf{s}_i - \mathbf{s}_j\|\,, \tag{19}$$

where $\mathcal{N}(i)$ is the 1-ring neighborhood of site i. This feature identifies regions where the actual density is low.

The second feature is the curvature proxy κ_i, which measures the average variation of the SDF gradient in the neighborhood of site i. This is computed as:

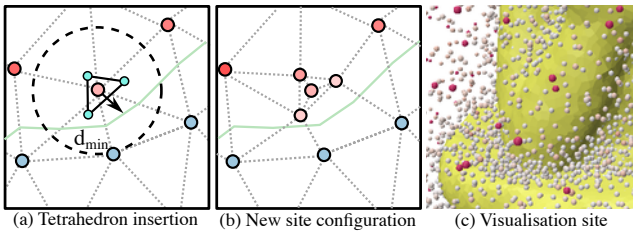
$$\kappa_i = \frac{\alpha_\kappa}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \left\|\widehat{\nabla}\phi_i - \widehat{\nabla}\phi_j\right\|^2 + (1-\alpha_\kappa), \tag{20}$$

where $\widehat{\nabla}\phi = \nabla\phi/(\|\nabla\phi\| + \epsilon)$ is the unit SDF gradient and $\alpha_\kappa = 0.8$ is to give a non zero score on flat regions. This feature captures the local curvature of the SDF, which is essential for identifying regions that require more sites.

Score computation and candidate selection. We compute a score for each Voronoi site $\mathbf{s}_i$ based on the local spacing and curvature features as follows:

$$\mathcal{S}_i = \left(\frac{\rho_i}{\tilde{\rho}}\right)\left(\frac{\kappa_i}{\tilde{\kappa}}\right), \quad i \in \mathcal{T}^{(0)}, \tag{21}$$

where $\mathcal{T}^{(0)}$ denotes the set of zero-crossing tetrahedron, $\tilde{\rho}$ and $\tilde{\kappa}$ are the median local spacing and curvature across all active sites, respectively. This score is designed to balance the uniform site spacing and curvature. Other sites producing a cell not crossing the zero-level receive a score of zero.



(a) Tetrahedron insertion (b) New site configuration (c) Visualisation site distribution

Figure 3. After selecting a site, we compute the minimal distance $\mathrm{d}_{\min}$ and insert a tetrahedron aligned with the SDF gradient (a). This preserves the local site connectivity (b). Our upsampling strategy results in a non-uniform site distribution (c).

We select sites candidates proportional to their score, by importance sampling proportionally to the score. A candidate site can be only selected once per upsampling iteration.

Scheduling. We perform upsampling at regular intervals until 80% of the optimization epochs, adding 10% more sites at each step relative to the current number of sites. In our experiments, we allow a maximum of 10 upsampling steps. To ensure a fair comparison with non-progressive upsampling methods, we cap the total number of sites.

Tetrahedral Insertion. One way to insert new sites is to randomly sample a point over a hemisphere centered at the candidate site and oriented along the spatial gradient of the SDF and based on its SDF distance. However, adding a single point can disturb the local connectivity between the Voronoi sites. Instead, we propose to insert a small tetrahedral structure around each selected site candidate. This approach allows us to maintain the local connectivity and regularity of our previous Voronoi tessellation. As shown in Fig. 3, for each selected site $\mathbf{s}_i$, we spawn 4 new sites forming a regular tetrahedron aligned with the spatial SDF gradient. Let $\{\mathbf{d}_k\}_{k=1}^4$ be the canonical tetrahedral directions and $\mathbf{T}_i \in \mathbb{R}^{3\times3}$ the local frame at $\mathbf{s}_i$, then:

$$\mathbf{s}'_{i,k} = \mathbf{s}_i + \frac{\rho_i}{4}\mathbf{T}_i\mathbf{d}_k, \quad k = 1,\dots,4\,, \tag{22}$$

where ρ_i is the local spacing defined in Eq. (19) and $\mathbf{s}'_{i,k}$ are the new sites to be inserted. We scale the regular tetrahedron using 1/4 of the local spacing to avoid adding new site which will cross the zero-level set. For each new site, we compute the SDF by using the SDF value at the original site $\mathbf{s}_i$ and the spatial gradient $\nabla\phi_i$ as follows:

$$\phi(\mathbf{s}'_{i,k}) \approx \phi(\mathbf{s}_i) + \nabla\phi_i^\top(\mathbf{s}'_{i,k} - \mathbf{s}_i). \tag{23}$$

4.4. Mesh extraction

The final vertex positions can be computed using either Eq. (6) or Eq. (8) for each vertex belonging to a tetrahedron intersecting the zero-level set, based on the optimized site positions and SDF values. However, because our SDF model relies on a linear approximation, Voronoi vertices are not guaranteed to lie exactly on the zero-level set. To ensure a consistent and watertight mesh, we adopt a simpler interpolation procedure inspired by Marching Tetrahedra [32].

In this method, we first identify the active tetrahedrons that are crossing the zero-level set. For each active tetrahedron, we compute their associated (unprojected) vertices and their SDF values via barycentric interpolation. We then identify the edge formed by an active vertex and its associated sites that are crossing the zero-level set. Finally we interpolate the vertex position along this edge at the zero-level set of the SDF. If a vertex results from multiple intersecting edges, we average all interpolated positions to determine its final location (Fig. 2f).

5. Results

We assess the effectiveness of our method in reconstructing detailed 3D meshes from unoriented point clouds using the publicly available Thingi32 dataset, a subset of the Thingi10K dataset [47] containing numerous high-resolution ground-truth meshes.

5.1. Experimental setup

In all experiments, we generate input point clouds by uniformly sampling 9.6k points from each ground-truth 3D mesh. The Voronoi sites are initialized on grids of varying resolutions, with a small random perturbation of 0.005 added to their positions to avoid coplanarity. Both the input point clouds and the initial site positions are then normalized to lie within the cube $[-1, 1]^3$.

SDF Initialization. For the SDF representation, we adopt the state-of-the-art neural network Hotspot [40], where each model is overfitted to the input point cloud. We use two versions of the trained model: (1) a fully converged model trained for 10k iterations and (2) an unconverged model trained for 500 iterations. The former is used to initialize a precise SDF representation, while the latter is used to evaluate our method's performance under an imprecise SDF representation. Note that we study shape ID 398259 separately in Fig. 6, as HotStop failed to produce a reliable SDF representation for this case.

Baselines. We compare our method against Voromesh [24] and Marching Tetrahedra (MTet) [32] as baselines. MTet is driven solely by the Chamfer distance computed from the extracted vertices. Voromesh minimizes a plane-based distance function and only optimizes the site positions. Which is why small extrusions can be seen on Figure 1 (c). In contrast, both MTet and our method jointly optimize the site positions and their associated SDF values. All methods are re-implemented in PyTorch [28] and initialized with the same sites. All methods use Adam optimizer [16] with a learning rate of 5×10^{-4} and β coefficients set to $(0.8, 0.99)$. All experiments are executed for a fixed 1k iterations.

Unlike Voromesh, both MTet and our method require a

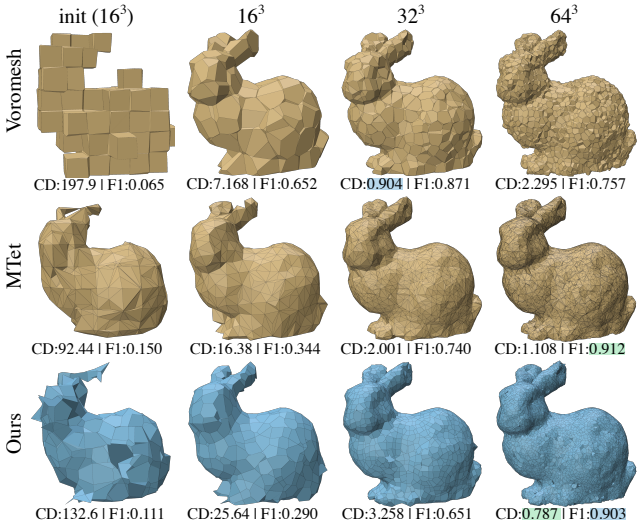


Figure 4. Optimized results for different resolution of sites with an accurate SDF representation.

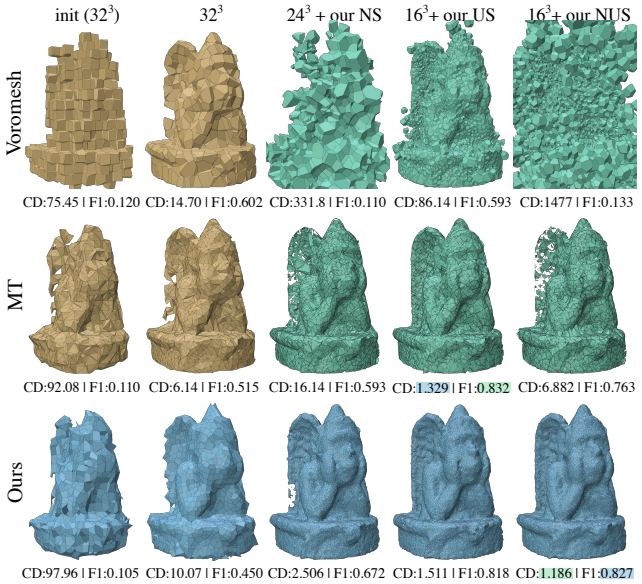


Figure 5. Results for different upsampling methods on an inaccurate SDF representation. Every optimization uses the same number of sites.

tetrahedral representation of the Voronoi sites, obtained via Delaunay triangulation. We use gDel3D [5] to accelerate this computation on the GPU. The triangulation is recomputed from scratch at every iteration. Exploring incremental updates to the triangulation as the Voronoi sites evolve remains an interesting direction for future work.

Centroid approximation. Computing the exact centroid is time consuming. To accelerate the process, we use an approximate centroid computation. Instead of evaluating the exact centroid of each Voronoi cell, we simply average the positions of the projected and un-projected vertices associated with the cell. In practice, we did not observe any noticeable difference in the final results.

Evaluation metrics. We assess mesh quality using three metrics: Chamfer Distance squared (CD), F1 score, and Normal Consistency (NC). To compute these metrics, we sample 1M points and apply face-to-point projection with FCPW [30]. The Chamfer distance measures how closely the reconstructed mesh aligns with the input point cloud, while the F1 score evaluates the ability of the mesh to capture the original shape. We set the F1 threshold to 0.003 to ensure sensitivity to fine details. For readability, all reported Chamfer distances are scaled by a factor of 10^5. In the tables and figures, we use a green color to highlight the best metric and blue color for second best.

5.2. Analysis

Converged SDF example. Figure 4 shows the results obtained by all methods for different initial site grid resolutions, and without upsampling. All methods, except Voromesh, remain stable across different grid resolutions for the initial site distribution. At higher resolutions, Voromesh receives too few projected target points per face, which leads to unstable face orientations. Although our 64^3 configuration introduces some surface noise, the extracted mesh remains predominantly uniform thanks to our CVT regularization.

Unconverged SDF example. Figure 5 illustrates a more challenging scenario where the SDF representation is inaccurate. In this case, the initial, unoptimized representation exhibits large missing regions due to the SDF errors. As shown in the second column of Figure 5, both our method and MTet [32] are able to recover from this inaccurate SDF at a 32^3 resolution. In contrast, Voromesh fails to recover, as it requires an accurate occupancy information for its final extraction.

Non-uniform site distribution and upsampling. Figure 5, other columns, present the results of three site placement strategies: (a) a naive approach, referred to as *near sampling*, which places a proportion of the sites close to uniformly subsampled input point cloud, (b) our tetrahedral insertion upsampling strategy (Sec. 4.3), and (c) a combination of these two strategies. In this combination, we start with a uniform grid of 16^3, add 16^3 near sampling sites, and then apply our iterative upsampling strategy. In all cases, these alternative placement strategies result in a total of 32^3 sites.

The near sampling strategy is effective only in our method, as we can efficiently move sites during optimization. In contrast, MTet suffers from this non-uniform site distribution, while Voromesh is unable to handle it due to instabilities in its target point projection. Our upsampling strategy improves performance by adaptively placing

Method	Converged			Unconverged		
	CD ↓	F1 ↑	NC ↑	CD ↓	F1 ↑	NC ↑
Voro 32^3	10.96	0.628	0.922	14.70	0.602	0.902
MTet 32^3	6.032	0.516	0.898	6.144	0.515	0.894
MTet U	2.539	0.682	0.916	2.443	0.682	0.917
Ours U	2.521	0.677	0.928	2.601	0.672	0.925
MTet N	7.400	0.636	0.873	16.14	0.593	0.838
Ours N	2.346	0.674	0.935	2.573	0.673	0.932
MTet NU	5.051	0.660	0.893	10.02	0.619	0.864
Ours NU	2.326	0.684	0.930	2.276	0.680	0.930

Table 1. Average score over the 31 meshes of Thingi32 over the MTet and VoroMesh 32^3 baseline versus our method. We include our near sample (N), upsampling (U) and combination (NU) approach for MTet and Ours for completeness.

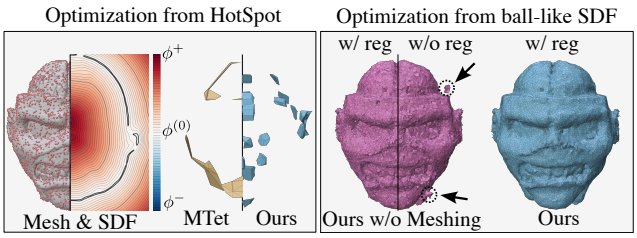


Figure 6. SDF interior/exterior issue causes mesh extraction issues for both MTet and Ours. Optimization from ball-like SDF works.

sites in regions with the highest reconstruction error proxy (Eq. (21)). Overall, combining near sampling with our upsampling strategy yields the best reconstruction results.

Average performance on Thingi32. Table 1 reports the average performance of all methods on the Thingi32 dataset, excluding mesh ID 398259 due to its inaccurate SDF representation with HotSpot. Our method consistently outperforms both Voromesh and MTet with fixed resolutions, showing the benefits of our site placement strategies and optimization process. As shown in Fig. 5, our upsampling strategy also improves the performance of MTet, while our method remains robust to all adaptive site placement strategies thanks to the CVT formulation and its connection to projected vertices. On average, the combination of near sampling and upsampling yields the best overall results. In terms of efficiency, our current implementation requires about 5 minutes for optimization with 32^3 sites on an RTX 3090 GPU—faster than MTet (9 minutes) but slower than Voromesh (1 minute). Leveraging approximations or more efficient optimization strategies could significantly reduce this computational overhead.

Simple SDF initialization. Figure 6 shows that the HotSpot SDF representation for mesh ID 398259 fails to correctly capture the interior–exterior relationship of the shape, due to the sparsity of sampled points. This poor initialization produces an extremely thin mesh volume, causing all methods that rely solely on zero-crossing information to fail. In contrast, since our method also optimizes the SDF representation, we can initialize from a simpler analytical SDF, such as a sphere, and still recover the target mesh. We note, however, that in this challenging case floating artifacts may appear when no regularization is applied to the SDF as visible in the purple mesh.

Method	CD ↓	F1 ↑	NC ↑
Ours 32^3	9.496	0.459	0.878
w/o CVT Reg.	11.68	0.433	0.835
w/o Meshing	13.23	0.257	0.860
w/o Midpoint	11.63	0.441	0.851

Table 2. Ablation study of our method with 32^3 grid resolution over the 31 shape of Thingi32.

5.3. Ablation study

Table 2 presents the ablation study for our different design choices using a 32^3 site resolution and a converged SDF initialization. We observe that the CVT regularization (Sec. 4.2) plays a crucial role in achieving high reconstruction quality, as also noted in earlier results. Our meshing strategy (Sec. 4.4), compared to directly using the projected sites from Eq. (3), yields better performance. Furthermore, projecting the midpoints (Eq. (7)) is important for improving face orientation regularity.

6. Conclusion and Future Work

We introduced DCCVT, a differentiable clipped centroidal Voronoi tessellation for 3D mesh reconstruction from point clouds. By projecting vertices onto the zero-level set while enforcing a CVT configuration, our method produces regular discretizations of optimized shapes. Furthermore, our framework supports different site placement strategies, enabling more accurate shape representations. Finally, our approach is capable of reconstructing meshes from various SDF representations, including unconverged or analytical initializations.

A promising future research direction is to combine DCCVT with deep learning frameworks to increase robustness when handling shapes with missing regions. Specifically, integrating visual foundation models could provide strong shape priors, allowing the system to plausibly fill gaps in the input point cloud and correct topological holes.

Acknowledgment This work was supported by JSPS/KAKENHI JP23H03439 and AMED JP24wm0625404 at Kyushu University, and by the NSERC Discovery Grant RGPIN-2022-03182. J. W. was supported by NSF and UC President's Postdoctoral Fellowships.

References

[1] Yizhak Ben-Shabat and Stephen Gould. Deepfit: 3d surface fitting via neural network weighted least squares. In *European conference on computer vision*, pages 20–34. Springer, 2020. 2

[2] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. Nesti-net: Normal estimation for unstructured 3d point clouds using convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10112–10120, 2019. 2

[3] Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. Digs: Divergence guided shape implicit neural representation for unoriented point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19323–19332, 2022. 2

[4] 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, 2002. 2

[5] Thanh-Tung Cao, Ashwin Nanjappa, Mingcen Gao, and Tiow-Seng Tan. A gpu accelerated algorithm for 3d delaunay triangulation. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2014. 7

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

[7] Gene Chou, Ilya Chugunov, and Felix Heide. Gensdf: Two-stage learning of generalizable signed distance functions. *Advances in Neural Information Processing Systems*, 35: 24905–24919, 2022. 2

[8] Qiang Du and Desheng Wang. Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International Journal for Numerical Methods in Engineering*, 56:1355 – 1373, 2003. 1

[9] 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. 2

[10] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 2

[11] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer graphics forum*, pages 75–85. Wiley Online Library, 2018. 2

[12] 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. 2

[13] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1251–1261, 2020. 2

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

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

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015. 6

[17] Jan Eric Lenssen, Christian Osendorfer, and Jonathan Masci. Deep iterative surface normal estimation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 11247–11256, 2020. 2

[18] Shengtao Li, Ge Gao, Yudong Liu, Ming Gu, and Yu-Shen Liu. Implicit filtering for learning neural signed distance functions from 3d point clouds. In *European Conference on Computer Vision*, pages 234–251. Springer, 2024. 2

[19] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *CVPR*, 2018. 1, 2

[20] Yaron Lipman. Phase transitions, distance functions, and implicit neural representations. *arXiv preprint arXiv:2106.07689*, 2021. 2

[21] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 3

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

[23] Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. *arXiv preprint arXiv:2011.13495*, 2020. 2

[24] Nissim Maruani, Roman Klokov, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. Voromesh: Learning watertight surface meshes with voronoi diagrams. In *ICCV*, 2023. 2, 5, 6

[25] Nissim Maruani, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. Ponq: a neural qem-based mesh representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3647–3657, 2024. 2

[26] 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. 2

[27] 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. 2

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019. 6

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

[30] Rohan Sawhney. Fcpw: Fastest closest points in the west, 2021. 7

[31] Nicholas Sharp and Maks Ovsjanikov. Pointtrinet: Learned triangulation of 3d point sets. In *European conference on computer vision*, pages 762–778. Springer, 2020. 2

[32] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *NeurIPS*, 34, 2021. 1, 2, 5, 6, 7

[33] Tianchang Shen, Zhaoshuo Li, Marc Law, Matan Atzmon, Sanja Fidler, James Lucas, Jun Gao, and Nicholas Sharp. Spacemesh: A continuous representation for learning manifold surface meshes. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 2

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

[35] Sanghyun Son, Matheus Gadelha, Yang Zhou, Zexiang Xu, Ming C Lin, and Yi Zhou. Dmesh: A differentiable representation for general meshes. *CoRR*, 2024. 2

[36] 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. 2

[37] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *Proceedings of the IEEE international conference on computer vision*, pages 2088–2096, 2017. 2

[38] Li Wang, Franck Hétroy-Wheeler, and Edmond Boyer. On volumetric shape reconstruction from implicit forms. In *European Conference on Computer Vision*, 2016. 2

[39] Zixiong Wang, Pengfei Wang, Peng-Shuai Wang, Qiujie Dong, Junjie Gao, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Wenping Wang. Neural-imls: Self-supervised implicit moving least-squares network for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):5018–5033, 2023. 2

[40] Zimo Wang, Cheng Wang, Taiki Yoshino, Sirui Tao, Ziyang Fu, and Tzu-Mao Li. Hotspot: Signed distance function optimization with an asymptotically sufficient condition. In *CVPR*, 2025. 2, 6

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

[42] Jane Wu, Diego Thomas, and Ronald Fedkiw. Sparse-view 3d reconstruction of clothed humans via normal maps. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2025. 2, 5

[43] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields. *arXiv preprint arXiv:2106.05187*, 2021. 2

[44] Chen Zhang and Wenbing Tao. Learning meshing from delaunay triangulation for 3d shape representation. *International Journal of Computer Vision*, 133(6):3413–3436, 2025. 2

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

[46] Chen Zhang, Wentao Wang, Ximeng Li, Xinyao Liao, Wanjuan Su, and Wenbing Tao. High-fidelity lightweight mesh reconstruction from point clouds. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 11739–11748, 2025. 2

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