PoNQ: a Neural QEM-based Mesh Representation

Nissim Maruani
Inria, Université Côte d'Azur
nissim.maruani@inria.fr

Maks Ovsjanikov
LIX, École Polytechnique, IP Paris
maks@lix.polytechnique.fr

Pierre Alliez
Inria, Université Côte d'Azur
pierre.alliez@inria.fr

Mathieu Desbrun
Inria Saclay - Ecole Polytechnique
mathieu.desbrun@inria.fr

Abstract

Although polygon meshes have been a standard representation in geometry processing, their irregular and combinatorial nature hinders their suitability for learning-based applications. In this work, we introduce a novel learnable mesh representation through a set of local 3D sample Points *and their associated* Normals *and* Quadric *error metrics (QEM) w.r.t. the underlying shape, which we denote PoNQ. A global mesh is directly derived from PoNQ by efficiently leveraging the knowledge of the local quadric errors. Besides marking the first use of QEM within a neural shape representation, our contribution guarantees both topological and geometrical properties by ensuring that a PoNQ mesh does not self-intersect and is always the boundary of a volume. Notably, our representation does not rely on a regular grid, is supervised directly by the target surface alone, and also handles open surfaces with boundaries and/or sharp features. We demonstrate the efficacy of PoNQ through a learning-based mesh prediction from SDF grids and show that our method surpasses recent state-of-the-art techniques in terms of both surface and edge-based metrics.*

(a) PoNQ over input (b) PoNQ mesh (c) Poisson mesh [26]

Figure 1. **PoNQ representation.** Quadric error metric (QEM) matrices are fitted and visualized on a given shape (left). Note that their aspect ratios (green to blue) capture the underlying surface information: pancakes for flat regions, cigars for sharp edges and balls for corners. Our mesh extraction outputs a watertight (or, optionally, open, see bottom) and non-self-intersecting mesh that preserves salient features of the input shape (center) and is more concise and faithful than current implicit approaches (right).

1. Introduction

In recent years, learning-based methods have shown great promise as an efficient means to handle ill-posed shape processing tasks with complex priors [20, 23, 59]. Yet, despite the slew of advanced architectures to analyze or process 3D datasets, there is a lack of learnable 3D representations that can capture ridges and corners, while guaranteeing valid output meshes representing real 3D shapes for even the most basic learning tasks.

Early works on shape representation for learning predominantly relied on implicit volumetric representations [59, 60], which, while eventually yielding a mesh through extraction, posed significant challenges. Notably, their training can be costly due to the volumetric nature of these representations. Moreover, the resulting mesh surfaces often lack detail, tending to generate overly smoothed shapes even where sharp features are expected. In response to these limitations, recent advancements have formulated explicit representations that effectively encode shapes through strategically chosen sample points and other geometric entities such as normals, from which more accurate meshes can be derived [35, 41]. While these representations offer higher generalizability and greater control over the generated surfaces, they often cannot guarantee important properties such as *watertightness* (i.e., the mesh must be the boundary of a non-degenerated volume), or the *absence of* *self-intersections* (i.e., the surface is not just immersed, but embedded in $\mathbb{R}^3$). Only very recently an approach has been proposed that enforces these properties [35]. However, it can suffer from overly complex output meshes (presence of many spurious polygonal facets) and, at times, wrong occupancy guesses create dents on the resulting shapes.

In this work, we propose PoNQ (short for point-normal-QEM), a new learnable 3D representation which encodes a shape through discrete points and other *local* geometric quantities to ensure efficient training and sharp reconstructions. As we demonstrate, a *global* mesh can be extracted from our PoNQ representation through a robust approach that leverages the available geometric data so as to better capture ridges and thin structures. Our key contributions are as follows:

- our neural representation is the first to exploit the quadric error metric (QEM), which has been instrumental in classical geometry processing tasks; consequently, PoNQ excels at capturing sharp features and boundaries, preserving the intricate details that are often lost in existing representations (see Fig. 1);
- PoNQ meshes are guaranteed to be watertight and free of self-intersections, thus broadening their applicability and utility in downstream applications;
- our PoNQ representation can also easily reduce the element count of the output shapes while preserving sharp features due to its reliance on QEM;
- finally, our neural representation outperforms state-of-the-art methods in mesh reconstruction from SDF grids as measured with both surface and edge-based metrics.

2. Related Work

We begin with a review of related works, including popular methods in shape learning, but also covering relevant parts of shape reconstruction and shape simplification.

Shape Reconstruction. The literature abounds with shape reconstruction methods (typically from input pointsets), each distinguished by their unique priors – necessary to regularize the ill-posed nature of the task – and output representations. In so-called continuous approaches, the reconstructed shape is the fixed-point of a projection operator [2], an algebraic point set surface (APSS [18]), or the isolevel of a 3D distance-like scalar field [26, 27], to mention a few popular approaches. Approaches from the latter category (often referred to as implicit methods) further require a mesh extraction from the scalar field, denoted isosurfacing, typically through Marching Cubes (MC [33], or its improved neural variant NMC [10]), Dual Contouring (DC [25], or its neural variant NDC [8]), or through the recently-proposed Reach-for-the-Spheres (RTS) technique [46] that leverages geometric properties of a signed distance field to improve isosurfacing.

Combinatorial methods, popular in Computational Geometry, rely instead on 3D Delaunay triangulations or their duals, Voronoi diagrams. The popular Crust approach, for instance, proceeds by local filtering of (facets of) 3D Delaunay triangulations [3]. However, it requires dense point samples to correctly capture thin features, and cannot handle sharp edges. Its Powercrust [4] and Tight Cocone [13] variants add filtering of the triangulation to improve various properties, like the capture of holes in the reconstructed surface, but they share the same limitations. A global graph-cut extraction of the faces of the Delaunay triangulation of the input samples [29] was also shown to be more robust to noisy inputs, just like a recent Delaunay-based approach that alternates between filmsticking and sculpting over a 3D triangulation [56].

Implicit Field Learning. Most pioneering 3D learning methods [36, 39, 42] represented a shape as a scalar field (very often, a signed distance field in fact), later converted into a mesh via Marching Cubes [33] or directly rendered. These methods need to process the global shape, limiting their generalizability beyond ShapeNet [7]. More recent implicit-based approaches [16, 49] showed that neural networks can be overfitted to single-shape Signed Distance Fields (SDF), and as such, they cannot learn from multiple shapes. Adding point-based convolutions [6], hash tables [37], octree structures [32, 52, 57] or kernel methods [22] have also been proposed to further refine these implicit methods, while the use of unsigned distance fields was leveraged to represent surfaces with boundaries [11, 19, 61]. Although using continuous fields provides topological guarantees, the lack of control of the final locations of mesh vertices yields a distinctively "blobby" aspect that smooths out sharp features and small details. Furthermore, implicit fields are trained over the whole *volume* around and inside the shape, while most applications are only interested in the reconstructed *surface*.

Explicit Shape Learning. To overcome the limitations of implicit fields and offer better control over the elements of the output mesh, end-to-end differential isosurfacing approaches were proposed [31, 45, 48]. Yet these approaches still rely on regular grids and often do not guarantee intersection-free output meshes. Methods that best fit a set of canonical geometric primitives [9, 12, 58] or an explicit mesh [38] to an input shape manage to produce concise meshes and preserve sharp features if these primitives (planes, quadric patches, etc) are diverse enough to describe the input mesh. However, they are rapidly compute-intensive even for a limited number of primitives. Deforming template shapes [17, 51] or regular tetrahedral meshes [14, 47] often constrains the output mesh genus and connectivity, and cannot guarantee intersection-free meshes. Another line of contributions approaches the design of learnable 3D representations from a different standpoint: provided a known reconstruction algorithm, what *geometric estimates* should the neural network predict in order to extract a mesh off of it? Methods relying upon the popular Poisson surface reconstruction algorithm to extract a mesh [41, 43] for instance inherit the topological advantages of an implicit surface representation, but they also suffer from their limitations by exhibiting overly-smoothed sharp features and a large amount of mesh elements. The Neural Marching Cubes (NMC [10]) and Neural Dual Contouring (NDC [8]) methods reduce some of these limitations, at the price of losing topological guarantees. Recently, VoroMesh [35], computes a Voronoi diagram so that a subset of the 3D Voronoi facets best fit the input mesh, guaranteeing watertight results; but VoroMeshes are littered with small faces with spurious normal orientations, causing both visual artifacts and large mesh element counts. A few learning methods rely on a Delaunay triangulation instead [34, 44, 50, 62], but they are limited to the case of a fixed input pointset. We recap in Tab. 1 some typical properties of explicit learnable 3D representations.

	Arbitrary connectivity	Sharp features	Watertight, no self-int.	Open surfaces
NDC [8]		✓		✓
DMT [47]		✓		
SAP [41]			✓	
DPF [43]	✓		✓	
VoroMesh [35]	✓	✓	✓	
PoNQ	✓	✓	✓	✓

Table 1. Properties of explicit learnable 3D representations

Quadric Error Metrics. Finally, we review a workhorse of surface approximation. In the context of mesh decimation, Garland and Heckbert [15] introduced a Quadric Error Metric (QEM), encoded as a 4×4 matrix Q, which evaluates the sum of squared distances to a (possibly large) set of tangent planes. The use of QEM for their application was particularly appropriate: the sum Q of two matrices Q_i and Q_j represent the sum of the squared distances to the union of the tangent planes used for Q_i and those used for Q_j – a very economical proxy to all these tangent planes. Consequently, they proposed to use these QEM matrices to perform an efficient mesh decimation: the updated matrices encode the squared distances of all nearby removed facets, thus offering a very fast and low-memory evaluation of the distance to the original mesh throughout the decimation. After a few rounds of simplifications, the ϵ-isovalue of QEM error near a vertex v_i (i.e., the set of 3D positions x such that $[x,1]^t Q_i [x,1] = \epsilon$) thus encodes the local region around v_i of the initial surface: it will look like a *pancake* if the original region was flat, an elongated ellipsoid if the region was a sharp feature (and the ellipsoid will precisely match the alignment of the sharp feature), or a sphere if the original region was near a corner. While QEM was adapted to deal with colors and texture coordinates [21], spherical distances [53], mesh filtering [30] and even variational mesh reconstruction from 3D pointsets [63], the authors of [54] recently suggested a probabilistic version of QEM as a potential representation for learning-based tasks, while [1] used QEM as a loss function. As we will see next, we will use a QEM matrix per discrete sample point in our learnable *representation* for 3D shapes instead, to better learn the shape of local regions.

3. Method

We now delve into our PoNQ representation by first offering a more in-depth introduction to QEM and how we use it in Sec. 3.1. We then describe our representation in Sec. 3.2 along with its use in learning tasks, and finally introduce our PoNQ mesh extraction approach in Sec. 3.3.

3.1. Motivation: Quadric Error Metrics

Given a sample point $s_k \in \mathbb{R}^3$ and a normal $n(s_k) \in \mathbb{R}^3$, the signed distance from a given location x to the plane passing through s_k that is normal to $n(s_k)$ is given simply as $d_{s_k,n(s_k)}(x) = (x - s_k)^t n(s_k)$. The squared distance can thus be rewritten as $d_{s_k,n(s_k)}(x)^2 = x^t A_k x - 2b_k^t x + c_k$, where 3×3 matrix A_k, vector b_k, and scalar c_k are:

$$A_k = n(s_k)n(s_k)^t, \quad b_k = A_k s_k, \quad c_k = s_k^t A_k s_k. \quad (1)$$

The sum of squared distances to a *set of planes* thus yields:

$$\begin{aligned}\mathrm{QEM}(x) &= \sum_k d^2_{s_k,n(s_k)}(x)\\ &= x^t \underbrace{\Big(\sum_k A_k\Big)}_{:=A} x - 2\underbrace{\Big(\sum_k b_k\Big)}_{:=b}^t x + \underbrace{\Big(\sum_k c_k\Big)}_{:=c} \quad (2)\\ &= [x,1]^t \underbrace{\begin{pmatrix} A & -b \\ -b^t & c \end{pmatrix}}_{:=Q} [x,1]. \quad (3)\end{aligned}$$

The summed distances to a fine sampling $\{s_k\}_k := S$ of a small surface patch can then be concisely encoded by the QEM matrix $Q \in \mathbb{R}^{4\times4}$ from Eq. (3). Moreover, Q can also indicate the best possible position to represent this surface patch with a single vertex: assuming the submatrix A to be invertible (which is true in the general case [54]), the quadratic QEM function reaches its unique minimum at a location $v^* = A^{-1}b$. We leverage these quantities as part of a powerful 3D *representation* suitable for machine learning.

3.2. PoNQ representation

The PoNQ representation consists in a set $\mathbf{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}$ of points, augmented with their local normals $\mathbf{N} = \{\mathbf{n}_i \in \mathbb{R}^3\}$ and quadrics $\mathbf{Q} = \{\mathbf{Q}_i \in \mathbb{R}^{4\times4}\}$ – hence its name. In order to remove possible ambiguities, we will use a bold font to refer to the different quantities involved in the PoNQ representation, i.e., what will be *optimized* or *learned*.

3.2.1 QEM-based representation via optimization

In a pure optimization-based setting, we want PoNQ to fit a watertight and non-self-intersecting input shape finely discretized by samples $s_k \in S$ with their local normals $n(s_k)$. We initialize the points $\mathbf{P}$ on a regular grid around the input shape (other initializations work equally well), and optimize their position to minimize the bi-directional Chamfer Distance CD:

$$\mathrm{CD}(\mathbf{P}, S) = \frac{1}{|\mathbf{P}|} \sum_{\mathbf{p}_i \in \mathbf{P}} \min_{s_k \in S} \|\mathbf{p}_i - s_k\|^2 \tag{4}$$

$$+ \frac{1}{|S|} \sum_{s_k \in S} \min_{\mathbf{p}_i \in \mathbf{P}} \|\mathbf{p}_i - s_k\|^2, \tag{5}$$

The resulting point set will thus lie close to (and spread out over) the target surface. These points define a partition of $\mathbb{R}^3$ into Voronoi cells $V(\mathbf{p}_i)$ for which any location $x \in V(\mathbf{p}_i)$ has $\mathbf{p}_i$ as its closest point from $\mathbf{P}$, see [5].

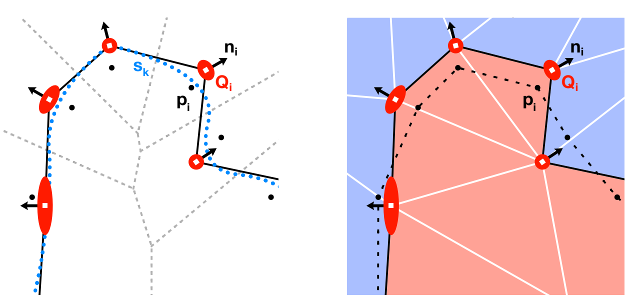
Once the positions are optimized, we enrich each point $\mathbf{p}_i \in \mathbf{P}$ with a normal $\mathbf{n}_i$ and a QEM matrix $\mathbf{Q}_i$, where $\mathbf{n}_i$ represents the average of the sample normals $n(s_k)$ for all the samples s_k contained within $V(\mathbf{p}_i)$ (see Fig. 2), i.e.,

$$\mathbf{n}_i = \frac{1}{|S \cap V(\mathbf{p}_i)|} \sum_{s_k \in S \cap V(\mathbf{p}_i)} n(s_k). \tag{6}$$

Similarly, $\mathbf{Q}_i$ is assembled using Eqs. (2) and (3) using the tangent planes implied by each sample (and its normal) within $V(\mathbf{p}_i)$, i.e., $\mathbf{Q}_i$ is the QEM matrix such that

$$[x, 1]^t \, \mathbf{Q}_i \, [x, 1] = \sum_{s_k \in S \cap V(\mathbf{p}_i)} d^2_{s_k, n(s_k)}(x). \tag{7}$$

Note that these additional variables are thus proxies for the local geometry around each point $\mathbf{p}_i$; points with no input samples within their Voronoi cell are simply discarded. As explained in Section 2, each matrix $\mathbf{Q}_i$ implies, through its sub-matrices $\mathbf{A}_i$ and $\mathbf{b}_i$, an optimal location $\mathbf{v}^* := \mathbf{A}_i^{-1}\mathbf{b}_i$ with respect to the input surface, hinting at the fact that storing $\mathbf{v}^*$ instead of $\mathbf{b}_i$ is a possible alternative, which we will use in the learning context in the next section. We then rely on $\mathbf{Q}$ and $\mathbf{N}$ to extract the connectivity of the optimal vertices $\mathbf{v}^*$, which will be explained in Sec. 3.3.



(a) *Optimized PoNQ point set* (b) *PoNQ meshing*

Figure 2. 2D illustration of PoNQ: (a) a sampled ground-truth shape S (blue dots) is represented by PoNQ as points $\mathbf{p}_i$ (whose Voronoi diagram (dotted lines) partitions the input samples), along with normals $\mathbf{n}_i$ and quadrics $\mathbf{Q}_i$ encoding the shape within each Voronoi cell. (b) The PoNQ mesh (black solid lines) is the boundary of the union of labeled tetrahedra from the Delaunay triangulation of the QEM-optimal vertices, providing a better fit than simply interpolating the points (black dotted lines).

3.2.2 Learning with PoNQ

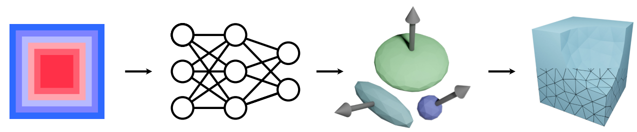


Figure 3. Overview of our learning pipeline with PoNQ.

We demonstrate the benefits of PoNQ in a learning context by applying it to reconstruction from Signed Distance Fields (SDF) (see Fig. 3). As shown in several recent works [8, 10, 35], this task is especially interesting as it enables the training of a *local* model that can truly generalize to novel shapes outside the training set.

We use an architecture similar to NDC [8], consisting of several convolutional neural networks (CNN). Unlike NMC [10] (which uses eight points per voxel) and NDC (which uses only one), we can use an arbitrary number P of predicted points per cell since our representation does not depend on a regular grid — in practice, we found that $P = 4$ can represent sub-voxel details nicely without requiring too large networks. We use a shared 5-layer encoder that converts the $(N+1)^3$ input SDF grid into a N^3-sized grid of 128 features. These features are processed by five separate 6-layer decoders (which do not share weights) to predict a PoNQ, i.e., $P \times N^3$ points $\mathbf{p}_i$, along with their associated local normals $\mathbf{n}_i$ and QEM matrices $\mathbf{Q}_i$, to which we add a set $\mathbf{O}$ of N^3 binary "occupancies" $\mathbf{o}_i$ to mark the voxels containing the surface at inference time.

To stabilize the learning process and avoid having to perform matrix inversions, we do not store QEM matrices directly but only use the quadratic form $\mathbf{A}_i$ and the QEM-optimized vertex location $\mathbf{v}^*$ instead, from which one can reconstruct $\mathbf{Q}_i = \begin{pmatrix} \mathbf{A}_i & -\mathbf{A}_i\mathbf{v}_i^* \\ -(\mathbf{A}_i\mathbf{v}_i^*)^t & 0 \end{pmatrix}$. Furthermore, we store instead of $\mathbf{A}_i$ a 3×3 upper triangular matrix $\mathbf{U}_i$ such that $\mathbf{A}_i = \mathbf{U}_i\mathbf{U}_i^t$ is the reversed Cholesky decomposition of $\mathbf{A}_i$, guaranteeing that $\mathbf{A}_i$ remains invertible.

We supervise our training with a collection of watertight shapes, each converted into a dense sampling S. We preprocess the data by computing the ground-truth occupancy set O_{gt} of the m voxels containing samples, to which we restrict our loss terms (except for L_{occ}, which checks how well the trained occupancy set matches the ground-truth one). At training time, we apply the sum of the following losses and backpropagate their gradients w.r.t. the bold variables:

$$L_{\text{CD}} = \text{CD}(\mathbf{P}, S) \quad (\textit{see Eq. (5)}) \tag{8}$$

$$L_{\mathbf{n}} = \frac{1}{m}\sum_{i=1}^{m} \sum_{s_k \in S \cap V(\mathbf{p}_i)} \|\mathbf{n}_i - n(s_k)\|^2 \tag{9}$$

$$L_{\mathbf{A}} = \frac{1}{m}\sum_{i=1}^{m} \sum_{s_k \in S \cap V(\mathbf{p}_i)} \|\mathbf{U}_i\mathbf{U}_i^t - n(s_k)n(s_k)^t\|^2 \tag{10}$$

$$L_{\mathbf{v}^*} = \frac{1}{m}\sum_{i=1}^{m} \sum_{s_k \in S \cap V(\mathbf{p}_i)} \left(n(s_k)^t(\mathbf{v}_i^* - s_k)\right)^2 \tag{11}$$

$$L_{\text{reg}} = \frac{1}{m}\sum_{i=1}^{m} \|\mathbf{v}_i^* - p_i\|^2 \;\; (\mathbf{p}_i \text{ assumed fixed here}) \tag{12}$$

$$L_{\text{occ}} = \|\mathbf{O} - O_{\text{gt}}\|^2 \tag{13}$$

In the order in which they are listed above, these losses were designed to: help spread around the pointset $\mathbf{P}$ and best fit the inputs; enforce that each normal is the mean normal of the local sample normals; enforce that the quadratic forms $\mathbf{A}_i$ correspond to the proper submatrix of the QEM of the local samples and normals around point $\mathbf{p}_i$; enforce, similarly, that $\mathbf{v}_i^*$ minimizes the sum of squared distance to the local tangent planes around point $\mathbf{p}_i$; regularize (with a very small coefficient) the positions of the optimal points $\mathbf{v}_i^*$ in flat regions — as in this case, any point on this flat region is optimal in theory, so we force it to stay close to $\mathbf{p}_i$; and make sure that we match the ground-truth occupancy set. See §2.3 of the supplementary material for ablation studies.

3.3. Meshing our representation

Given a PoNQ representation (either optimized or produced by a trained network), one can easily extract a mesh by combining two approaches from computational geometry [3, 29] to ensure robustness. Note that we do not even use the pointset $\mathbf{P}$ which only served to build a shape-adapted partition: we construct a mesh whose vertices are the QEM-optimal positions $\mathbf{v}_i^*$ as they capture features best. We present here a concise overview of our meshing method; see our supplemental material for details.

Pre-processing We first compute the Delaunay tetrahedralization of the optimal vertices $\mathbf{v}_i^*$ deriving from the QEM matrices $\mathbf{Q}$. The next two steps will tag each tetrahedron as either *inside* or *outside* based on local information, so that *our final PoNQ mesh will be simply the triangle mesh forming the inside/outside boundary* — ensuring watertightness and no self-intersections by design.

Tagging obvious inside/outside tetrahedra. We leverage the circumcenter criterion put forth in the Crust algorithm [3]. In our case, each vertex $\mathbf{v}_i^*$ and its assigned normal $\mathbf{n}_i$ define an oriented plane: we tag a tetrahedron as *outside* (resp., *inside*) if both its circumcenter and barycenter are determined to be in the *outside* (resp., *inside*) half-space of each of the four vertices. Considering that the shape is contained within the convex hull of the $\mathbf{v}_i^*$ and that each vertex of the Delaunay tetrahedralization must be part of the final surface allows us to further tag a series of tetrahedra where there is no ambiguity; see our supplemental material for a detailed rationale.

Figure 4. Optimization-based results (32^3).

Tagging remaining tetrahedra. Delaunay-based meshing approach (like *Crust* [3]) require a dense point sampling (formally, an $\epsilon-$sampling), which is not compatible with our desire to deal with thin structures, sharp features and corners — and this is the main reason why our earlier phase can end up not providing a tag for *every* tetrahedra. To finish our tetrahedron tagging based on the ones we already have, we use a graph cut approach, inspired by (but simpler than) an existing spectral graph partitioning [29]. For each Delaunay triangle T, we compute a likelihood score $S(T)$ that evaluates how confident we are that this triangle is to appear on the final output mesh: this triangle score evaluates the fitness of T based on the local normals $\mathbf{N}$ and quadric matrices $\mathbf{Q}$ as explained in the supplemental material. We now tag the remaining undetermined tetrahedra through a *minimum cut* of the Voronoi graph (in which each dual of a tetrahedron is a node, and each dual of a Delaunay triangle T is an edge with weight $S(T)$) using the already-tagged "inside" ones as a source and the "outside" one as a drain.

Surface extraction. We extract the final *PoNQ mesh* as the triangle mesh forming the boundary between the inside and outside tetrahedra (see Fig. 2). As mentioned above, this automatically enforces by design the fact that our mesh is watertight and intersection-free.

4. Experimental Results

We implemented PoNQ for both optimization and learning tasks in Python, using PyTorch [40], SciPy [55] and libigl [24] — our code is available on our project page. All timings were computed on a single workstation with 54 cores, a NVidia A6000 GPU and 512 GB of RAM.

4.1. Optimization-based 3D Reconstruction

For our tests in optimization-based 3D reconstruction, we use the 30 watertight shapes from the Thingi10k [64] dataset chosen in VoroMesh [35] as it is arguably the most related and most recent neural representation with which to compare. We consider three grid resolutions res $\in [32^3, 64^3, 128^3]$, and sample $1024 \times \text{res}^{2/3}$ surface samples. As described in Sec. 3.2.1, we use the chamfer distance as a loss to optimize the points for 400 epochs with the Adam optimizer before computing the mean normals and quadrics with our GPU-based implementation. We then extract the PoNQ representation as explained in Sec. 3.3.

Baselines. Besides VoroMesh, we also compare our results with Shape As Points (SAP) [41] and Dynamic Point Field (DPF) [43] as these three point-based neural representations all guarantee watertight outputs. We add a variant of DPF, trained with the Chamfer distance only (i.e., without the image-based loss), which we denote $\text{DPF}_{\text{chamfer}}$. For each method, we use the same number of optimized points.

Metrics. We use the most common surface-based metrics, i.e., chamfer distance (CD), F-score (F1, with a threshold of 0.003), and normal consistency (NC). In order to assess the reconstruction quality of sharp edges, we sample 10^5 points on the edges featuring a dihedral angle larger than $\frac{\pi}{6}$, and compute the edge-chamfer distance (ECD) and edge-F-score (EF1, with a threshold of 0.005) between the ground truth and the reconstruction samples (see supplemental material for details). We also report the number of triangles and faces of the extracted 3D models. Finally, we provide timings of the optimization step for all methods, as it is systematically the most time-consuming phase; see supplemental material for additional timings.

Results. As SAP and DPF both rely on Poisson surface reconstruction, they smooth out sharp edges and small details. DPF does not optimize the extracted field, but rather leverages local information from which the mesh is assembled without supervision, leading to faster convergence and better scores than SAP for all metrics. As the evaluation point cloud is sampled from the reconstructed Poisson mesh (rather than the predicted points), the image-based loss does not significantly improve results, and $\text{DPF}_{\text{chamfer}}$ is faster than DPF with comparable performance.

VoroMesh exhibits finer details and better sharpness overall. With the same number of points as SAP, DPF and PoNQ, it has the best overall surface fitting scores due to the large number of Voronoi vertices and faces. However many of these faces have spurious normals and create undesired sharp edges, thus impacting the NC, ECD, and EF1 scores.

Qualitatively, PoNQ is better at dealing with sharp features than SAP and DPF and does not generate surface artifacts like VoroMesh (see Figure 4). Quantitatively, our method yields the best normal consistency and sharp edge fitting scores (see Tab. 2). In terms of surface fitting, it matches the considered baselines at resolution 128^3, and comes close second behind VoroMesh at resolutions 32^3 and 64^3, but with significantly lower face counts. We will discuss in Sec. 4.3 how one can further lower the face count of PoNQ meshes at very little cost on the scores (see Fig. 7), setting PoNQ apart even more prominently.

Method	Grid Size	CD ↓ ($\times 10^{-5}$)	F1 ↑	NC ↑	ECD ↓ ($\times 10^3$)	EF1 ↑ ($\times 10^3$)	# V ($\times 10^3$)	# F ($\times 10^3$)	Time (s)
SAP [41],	32^3	6.475	0.589	0.894	0.235	0.058	1.6	3.2	51.2
DPF [43]	32^3	2.256	0.724	0.935	0.147	0.115	5.8	11.6	45.5
$\text{DPF}_{\text{chamfer}}$ [43]	32^3	2.077	0.717	0.933	0.160	0.104	5.7	11.3	2.5
VoroMesh [35]	32^3	**0.802**	**0.919**	0.957	0.257	0.242	5.9	11.8	2.0
PoNQ	32^3	0.972	0.892	**0.961**	**0.106**	**0.447**	2.3	4.6	2.6
SAP [41],	64^3	1.912	0.858	0.949	0.119	0.267	7.0	13.9	106.3
DPF [43]	64^3	0.795	0.909	0.971	0.104	0.435	22.6	45.1	51.5
$\text{DPF}_{\text{chamfer}}$ [43]	64^3	0.797	0.907	0.970	0.094	0.415	24.1	48.3	4.0
VoroMesh [35]	64^3	**0.645**	**0.938**	0.975	0.251	0.249	23.4	46.9	4.1
PoNQ	64^3	0.655	0.936	**0.981**	**0.061**	**0.645**	10.1	20.1	4.1
SAP [41],	128^3	0.671	0.934	0.978	0.060	0.619	28.8	57.7	191.3
DPF [43]	128^3	0.644	0.938	0.986	0.089	0.665	90.0	180.0	71.0
$\text{DPF}_{\text{chamfer}}$ [43]	128^3	0.644	0.938	0.986	0.086	0.664	97.3	194.5	17.7
VoroMesh [35]	128^3	**0.634**	**0.939**	0.982	0.264	0.213	91.3	182.6	36.3
PoNQ	128^3	0.637	**0.939**	**0.988**	**0.039**	**0.795**	42.3	84.6	17.9

Table 2. Optimization-based results. Quantitative comparisons of Chamfer distance (CD), F1 score, and normal consistency (NC) on the Thingi30 dataset for three different grid resolutions.

4.2. Learning-based 3D reconstruction

4.2.1 Reconstruction from SDF

We now assess the behavior of our PoNQ representation in the learning-based task of 3D shape reconstruction from SDF grids. We train and evaluate our method on the CAD shapes of the ABC dataset [28]. We also assess the generalizability of our method on the free-form shapes of the Thingi10k [64] dataset, without any fine-tuning. For fairness, we use the train/test split provided in VoroMesh [35]: 3,843/962 in the training and testing set for ABC, with 30 watertight validation shapes for Thingi10k. We train our network for 600 epochs while increasing the number of sampled points and decreasing the learning rate and regularization – see supplemental material for additional details.

Baselines. We compare our method against the three most closely related baselines: NMC [10], NDC [8] and VoroMesh [35] using the authors' code and their best pretrained model (we also discuss comparisons with RTS [46] and DMTet [47] in the supplementary material).

Metrics. We use the same evaluation metrics that were already mentioned in Sec. 4.1.

Results. Our key results are reported in Tabs. 3 and 4, where PoNQ outperforms state-of-the-art methods *on every resolution, dataset, and metric while guaranteeing watertight output meshes that are devoid of self-intersections*. We note that NMC and NDC both rely on a regular-grid based meshing. As a result, they fail to capture thin structures and exhibit aliasing artifacts (see Figs. 5 and 6), which impacts their surface, normal and sharp-edge fitting scores.

VoroMesh does not rely on a regular grid, so its ability to capture thin surfaces leads to better results than NMC

(a) NDC (b) NMC (c) VoroMesh (d) PoNQ (e) Gr. Truth

Figure 5. Learning results (top: 32^3; bottom: 64^3) on ABC.

Method	Grid Size	CD ↓ ($\times 10^{-5}$)	F1 ↑	NC ↑	ECD ↓	EF1 ↑	Watertight ↑ no self-int.	# V ($\times 10^3$)	# F ($\times 10^3$)
NDC [8]	32^3	66.004	0.787	0.941	0.445	0.658	44%	1.3	2.6
NMC [10]	32^3	60.755	0.833	0.954	0.350	0.693	26%	9.7	19.3
VoroMesh [35]	32^3	2.228	0.835	0.941	0.802	0.232	**100%**	10.0	20.0
PoNQ-lite	32^3	3.539	0.810	0.953	0.296	0.658	**100%**	1.3	2.6
PoNQ	32^3	**1.514**	**0.852**	**0.964**	**0.184**	**0.713**	**100%**	5.1	10.2
NDC [8]	64^3	2.211	0.882	0.975	0.223	0.855	23%	5.5	11.0
NMC [10]	64^3	2.138	0.891	**0.980**	0.254	0.854	18%	42.8	85.5
VoroMesh [35]	64^3	1.219	0.886	0.966	0.796	0.207	**100%**	38.4	76.9
PoNQ-lite	64^3	1.074	0.888	0.978	0.128	0.858	**100%**	5.5	10.9
PoNQ	64^3	**0.886**	**0.892**	**0.980**	**0.109**	**0.866**	**100%**	21.2	42.3
NDC [8]	128^3	1.889	**0.896**	0.983	0.095	**0.947**	14%	22.1	44.2
NMC [10]	128^3	1.888	**0.896**	**0.984**	0.349	0.859	12%	175.9	351.9
VoroMesh [35]	128^3	1.069	0.894	0.974	0.792	0.189	**100%**	149.1	298.2
PoNQ-lite	128^3	1.007	**0.896**	**0.984**	**0.043**	0.933	**100%**	21.9	43.8
PoNQ	128^3	**0.920**	**0.896**	**0.984**	0.191	0.878	**100%**	85.7	171.2

Table 3. Results on ABC with our network trained on ABC.

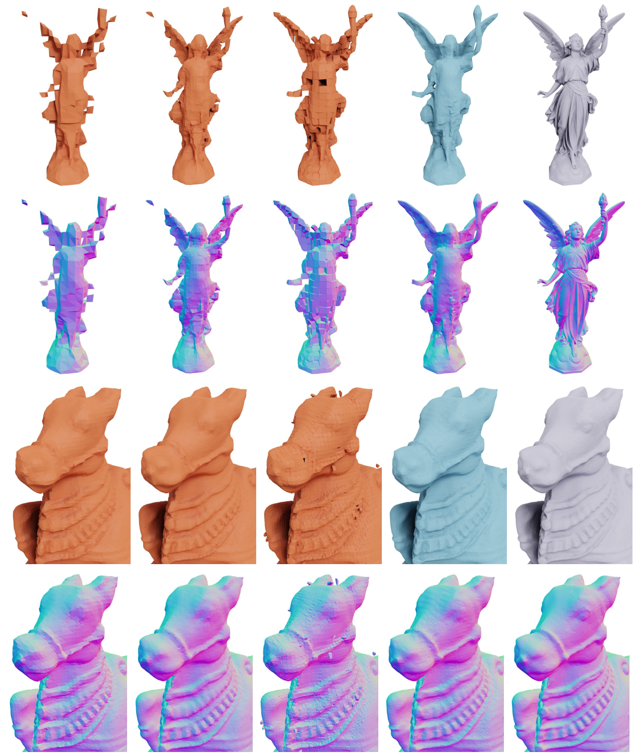
and NDC for the complex ABC dataset or on low-resolution models of Thingi30. However, as noticed in Tabs. 3 and 4, it suffers from local surface artifacts leading to sharp edges and spurious normals, resulting in worst NC, ECD and EF1. Moreover, its two-stage training implies that its encoder is not trained for occupancy prediction; as a result, it can mislabel Voronoi generators, leading to either missing parts or floating volumes around the shapes (see Figs. 5 and 6).

In contrast, our method is able to capture fine details due to our use of QEM, which helps to capture surface characteristics, and it does not visually exhibit any of the artifacts of other approaches due to our PoNQ mesh being extracted from a Delaunay tetrahedralization. Ultimately, PoNQ leads to more faithful reconstructions while guaranteeing 100% watertight and intersection-free results.

4.3. Additional extensions

We conclude this section with other examples leveraging the unique nature of our neural representation.

Surfaces with boundary. With minor modifications, PoNQ can output surfaces with boundaries as well (Fig. 1 and supplementary material). To optimize or train our representation for this case, we simply compute a *boundary sampling* S_b of the input surface boundaries, and duplicate it



(a) NDC (b) NMC (c) VoroMesh (d) PoNQ (e) G. Truth

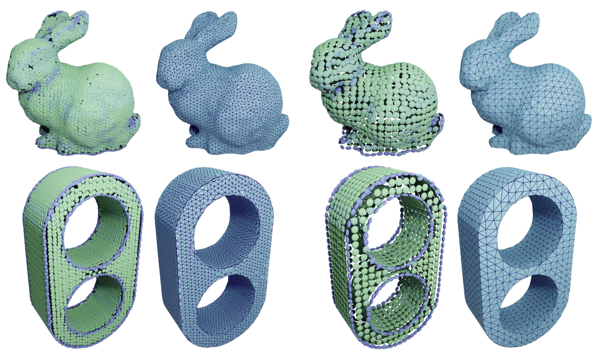
Figure 6. Learning results (top: 32^3; bottom: 128^3) on Thingi30.

Method	Grid Size	CD ↓ ($\times 10^{-5}$)	F1 ↑	NC ↑	ECD ↓	EF1 ↑	Watertight ↑ no self-int.	# V ($\times 10^3$)	# F ($\times 10^3$)
NDC [8]	32^3	6.390	0.745	0.920	0.172	0.245	40%	1.3	2.6
NMC [10]	32^3	5.188	0.796	0.936	0.148	0.271	0%	8.7	17.3
VoroMesh [35]	32^3	2.825	0.758	0.902	0.263	0.156	**100%**	9.9	19.9
PoNQ-lite	32^3	1.705	0.754	0.934	0.154	0.270	**100%**	1.3	2.6
PoNQ	32^3	**1.344**	**0.810**	**0.942**	**0.137**	**0.314**	**100%**	4.8	9.7
NDC [8]	64^3	0.849	0.908	0.961	0.106	0.441	3%	5.4	10.8
NMC [10]	64^3	0.776	0.923	0.969	0.115	0.467	0%	36.8	73.6
VoroMesh [35]	64^3	1.021	0.906	0.939	0.259	0.192	**100%**	39.4	78.9
PoNQ-lite	64^3	0.769	0.914	0.968	**0.090**	0.495	**100%**	5.3	10.6
PoNQ	64^3	**0.758**	**0.924**	**0.971**	0.100	**0.511**	**100%**	19.9	39.9
NDC [8]	128^3	0.650	0.937	0.980	0.065	0.644	0%	22.0	44.1
NMC [10]	128^3	0.642	**0.939**	**0.984**	0.131	0.574	0%	151.7	303.3
VoroMesh [35]	128^3	0.731	0.932	0.959	0.260	0.198	**100%**	157.2	314.5
PoNQ-lite	128^3	0.644	0.938	**0.984**	**0.055**	**0.699**	**100%**	21.7	43.3
PoNQ	128^3	**0.641**	**0.939**	**0.984**	0.123	0.592	**100%**	80.8	161.8

Table 4. Results on Thingi30 with our network trained on ABC.

to create a sampling S_b' where we just rotate all the normals by $\pi/2$ around the boundary. Changing S into $S \cup S_b \cup S_b'$ will thus enforce that all the QEM matrices on the boundaries will generate elongated ellipsoids aligned with the local boundary. Meshing can proceed as before; but the output mesh will automatically close the holes. We further cull any triangle T of the extracted mesh for which the anisotropy of the QEM of each of its vertices is above 40%. In practice, we measure the anisotropy through the ratio $r_i = \lambda_2/\lambda_1$ of the two largest eigenvalues of the matrix $\mathbf{A}_i$. Obviously, the surface no longer bounds a volume since we cut holes in the original PoNQ mesh extraction, but it remains devoid of self-intersections. Note that this PoNQ variant cannot handle special cases such as a single square sheet.

PoNQ-lite. While using $P=4$ predicted points per voxel yielded the best performances in our learning-based tests as mentioned in Sec. 3.2.2, we can trivially provide a "lite" version of our output PoNQ with a single point per cell using the same network: due to our reliance on QEM matrices, one can simply sum the P matrices $\mathbf{Q}_i$ within a cell to directly create a single QEM matrix per cell, from which is derived a new optimal position $\mathbf{v}_i^*$ for each cell. The normals $\mathbf{n}_i$ are also trivially averaged into the new one. Sharp features are still well preserved due to our use of quadric error metrics (see Fig. 7), and these PoNQ-lite meshes in fact outperform NDC for an equal level of element count (see Tabs. 3 and 4), proving the superiority of grid-free methods. If even coarser meshes are desired, one can also construct, at nearly no cost, a whole hierarchy of PoNQ meshes by first applying PoNQ-lite, and then merging each group of eight cells into a single larger one to form a twice-coarser grid (2×2 average pool; see Fig. 8).



(a) PoNQ (b) PoNQ-lite

Figure 7. For a network trained on ABC with $P = 4$ predicted QEM matrices per cell (left), one can sum these matrices – i.e., via average pooling – per cell to produce a more compact mesh (right) that still naturally preserves the detected sharp features.

Although PoNQ outperforms PoNQ-lite on almost all metrics, the latter slightly pulls ahead on ECD and EF1 on Thingi30 at 128^3 resolution. This opens the door to a series of further investigations, out of scope for this paper: one could potentially simplify a PoNQ output *adaptively* depending on the contents of the cell.

5. Discussion

The QEM matrices, encoded via points v_i^* and SPD matrices A_i are essential to our work: besides their role in capturing sharp features and second-order shape properties, they (a) disambiguate meshing compared to just point+normals to achieve state-of-the-art results (see supplementary material), (b) allow direct simplification through quadric collapses (PoNQ-lite, Fig. 7 & 8), and (c) allow open boundaries (Fig. 1). Yet, compared to SAP or VoroMesh, one may wonder if the added quadric information are worth a higher network size. In fact, producing QEM information does not significantly affect network size since 80% of the weights are concentrated in the shared encoder. Moreover, the task of fitting points, QEM and normals is quite straightforward and does not require overly large networks (e.g., VoroMesh requires 8.4M parameters while PoNQ only requires 2.7M).

Another possible perceived limitation is that our PoNQ mesh extraction may create non-manifold vertices or edges despite being watertight – for instance, a 1-ring of a vertex may contain two *non-adjacent* tetrahedra both labeled as *inside*. However, one can simply duplicate these few non-manifold elements and connect them to their neighboring mesh elements to enforce manifoldness [35]. In addition, final results can be affected by two types of flaws: wrong estimates of geometric quantities (CNN failure, noise in the motor plate in Fig. 5), or tetrahedra mislabelling (reconstruction failure, unwanted link elbow-body Fig 6).

There are also exciting aspects of PoNQ we have not explored yet. For example, our meshing of open boundaries can potentially be further refined to extract directly the surface through a tagging of boundary edges, instead of relying on a potentially brittle final score-based filtering. A nice extension to our work would be to integrate the PoNQ representation in a differentiable rendering pipeline. Finally, the naturally multiscale nature of PoNQ through average pooling is also bound to be exploitable in a number of contexts.

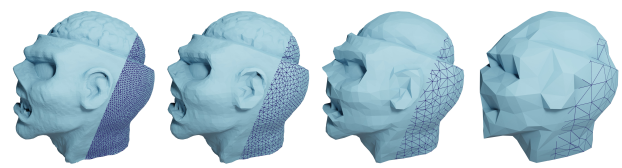


Figure 8. Learning-based results. From PoNQ-lite (128^3), to power-of-two simplifications down to 16^3 via average pooling.

6. Conclusion

We proposed a novel learnable 3D shape representation, coined PoNQ which combines the power of the quadric error metric (QEM) originally devised for mesh decimation and insights from computational geometry. PoNQ relies on points, normals, and QEM matrices to represent local geometric information, which are later leveraged to construct a triangle mesh that is guaranteed to be the boundary of a volume and devoid of self-intersections. We demonstrated the representation power of PoNQ through optimization-based tasks and learning-based reconstruction experiments, showing significant improvement upon previous 3D representations. We thus believe that PoNQ is poised to find many applications and extensions in neural shape processing.

Acknowledgments. Work supported by 3IA Côte d'Azur (ANR-19-P3IA-0002), ERC Starting Grant 758800 (EXPROTEA), ERC Consolidator Grant 101087347 (VEGA), ANR AI Chair AIGRETTE, Ansys, Adobe Research, and a Choose France Inria chair.

References

[1] Nitin Agarwal, Sung-eui Yoon, and M. Gopi. Learning Embedding of 3D models with Quadric Loss, July 2019. arXiv:1907.10250 [cs]. 3

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. Point set surfaces. In *Proceedings Visualization, 2001. VIS '01.*, pages 21–537, San Diego, CA, USA, 2001. IEEE. 2

[3] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, pages 415–421, Not Known, 1998. ACM Press. 2, 5

[4] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, SMA '01, pages 249–266, New York, NY, USA, May 2001. Association for Computing Machinery. 2

[5] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3):345–405, Sept. 1991. 4

[6] Alexandre Boulch and Renaud Marlet. POCO: Point Convolution for Surface Reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6292–6304, New Orleans, LA, USA, June 2022. IEEE. 2

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository, Dec. 2015. arXiv:1512.03012 [cs]. 2

[8] Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics*, 41(4):1–13, July 2022. 2, 3, 4, 6, 7

[9] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. BSP-Net: Generating Compact Meshes via Binary Space Partitioning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 42–51, Seattle, WA, USA, June 2020. IEEE. 2

[10] Zhiqin Chen and Hao Zhang. Neural marching cubes. *ACM Transactions on Graphics*, 40(6):1–15, Dec. 2021. 2, 3, 4, 6, 7

[11] Thor V. Christiansen, Jakob Andreas Bærentzen, Rasmus R. Paulsen, and Morten R. Hannemose. Neural Representation of Open Surfaces. *Computer Graphics Forum*, 42(5):e14916, 2023. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14916. 2

[12] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. CvxNet: Learnable Convex Decomposition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 31–41, Seattle, WA, USA, June 2020. IEEE. 2

[13] Tamal K. Dey and Samrat Goswami. Tight cocone: a watertight surface reconstructor. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 127–134, Seattle Washington USA, June 2003. ACM. 2

[14] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning Deformable Tetrahedral Meshes for 3D Reconstruction. In *Advances in Neural Information Processing Systems*, volume 33, pages 9936–9947. Curran Associates, Inc., 2020. 2

[15] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*, pages 209–216, Not Known, 1997. ACM Press. 3

[16] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3789–3799. PMLR, Nov. 2020. ISSN: 2640-3498. 2

[17] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A Papier-Mache Approach to Learning 3D Surface Generation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 216–224, Salt Lake City, UT, USA, June 2018. IEEE. 2

[18] Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. In *ACM SIGGRAPH 2007 papers*, page 23, San Diego California, July 2007. ACM. 2

[19] Benoît Guillard, Federico Stella, and Pascal Fua. MeshUDF: Fast and Differentiable Meshing of Unsigned Distance Field Networks. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, Lecture Notes in Computer Science, pages 576–592, Cham, 2022. Springer Nature Switzerland. 2

[20] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, Dec. 2021. 1

[21] Hugues Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *Proceedings Visualization '99 (Cat. No.99CB37067)*, pages 59–510, San Francisco, CA, USA, 1999. IEEE. 3

[22] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural Kernel Surface Reconstruction. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4369–4379, Vancouver, BC, Canada, June 2023. IEEE. 2

[23] Zhangjin Huang, Yuxin Wen, Zihao Wang, Jinjuan Ren, and Kui Jia. Surface Reconstruction from Point Clouds: A Survey and a Benchmark, May 2022. arXiv:2205.02413 [cs]. 1

[24] Alec Jacobson and Daniele Panozzo. libigl. 5

[25] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 339–346, New York, NY, USA, July 2002. Association for Computing Machinery. 2

[26] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. *Poisson Surface Reconstruction*. The Eurographics Association, 2006. Accepted: 2014-01-29T08:14:02Z ISSN: 1727-8384. 1, 2

[27] Michael Kazhdan and Hugues Hoppe. Screened poisson

surface reconstruction. *ACM Transactions on Graphics*, 32(3):1–13, June 2013. 2

[28] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A Big CAD Model Dataset For Geometric Deep Learning, Apr. 2019. arXiv:1812.06216 [cs]. 6

[29] 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, Nice France, July 2004. ACM. 2, 5

[30] Hélène Legrand, Jean-Marc Thiery, and Tamy Boubekeur. Filtered Quadrics for High-Speed Geometry Smoothing and Clustering. *Computer Graphics Forum*, 38(1):663–677, Feb. 2019. 3

[31] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep Marching Cubes: Learning Explicit Surface Representations. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, Salt Lake City, UT, June 2018. IEEE. 2

[32] 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 *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1788–1797, Nashville, TN, USA, June 2021. IEEE. 2

[33] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, Aug. 1987. 2

[34] Yiming Luo, Zhenxing Mi, and Wenbing Tao. DeepDT: Learning Geometry From Delaunay Triangulation for Surface Reconstruction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(3):2277–2285, May 2021. Number: 3. 3

[35] Nissim Maruani, Roman Klokov, Maks Ovsjanikov, Pierre Alliez, and Mathieu Desbrun. VoroMesh: Learning Watertight Surface Meshes with Voronoi Diagrams. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14565–14574, 2023. 1, 2, 3, 4, 6, 7, 8

[36] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space, Apr. 2019. arXiv:1812.03828 [cs]. 2

[37] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022. 2

[38] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter Battaglia. PolyGen: An Autoregressive Generative Model of 3D Meshes. In *Proceedings of the 37th International Conference on Machine Learning*, pages 7220–7229. PMLR, Nov. 2020. ISSN: 2640-3498. 2

[39] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, Long Beach, CA, USA, June 2019. IEEE. 2

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, Dec. 2019. arXiv:1912.01703 [cs, stat]. 5

[41] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape As Points: A Differentiable Poisson Solver. In *Advances in Neural Information Processing Systems*, volume 34, pages 13032–13044. Curran Associates, Inc., 2021. 1, 3, 6

[42] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional Occupancy Networks, Aug. 2020. arXiv:2003.04618 [cs]. 2

[43] Sergey Prokudin, Qianli Ma, Maxime Raafat, Julien Valentin, and Siyu Tang. Dynamic Point Fields. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7964–7976, 2023. 3, 6

[44] Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy Mitra, and Maks Ovsjanikov. Learning Delaunay Surface Elements for Mesh Reconstruction. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22–31, Nashville, TN, USA, June 2021. IEEE. 3

[45] Edoardo Remelli, Artem Lukoianov, Stephan Richter, Benoit Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. MeshSDF: Differentiable Iso-Surface Extraction. In *Advances in Neural Information Processing Systems*, volume 33, pages 22468–22478. Curran Associates, Inc., 2020. 2

[46] Silvia Sellán. Reach For the Spheres: Tangency-Aware Surface Reconstruction of SDFs. *ACM Transactions on Graphics*, 2023. 2, 6

[47] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 6087–6101. Curran Associates, Inc., 2021. 2, 3, 6

[48] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible Isosurface Extraction for Gradient-Based Mesh Optimization. *ACM Transactions on Graphics*, 42(4):1–16, Aug. 2023. 2

[49] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020. 2

[50] Raphael Sulzer, Loic Landrieu, Renaud Marlet, and Bruno Vallet. Scalable Surface Reconstruction with Delaunay-Graph Neural Networks. *Computer Graphics Forum*, 40(5):157–167, 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14364. 3

[51] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Vari-

ational Autoencoders for Deforming 3D Mesh Models. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5841–5850, 2018. 2

[52] Jia-Heng Tang, Weikai Chen, Jie Yang, Bo Wang, Songrun Liu, Bo Yang, and Lin Gao. OctField: Hierarchical Implicit Functions for 3D Modeling, Nov. 2021. arXiv:2111.01067 [cs]. 2

[53] Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. Sphere-Meshes: shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics*, 32(6):1–12, Nov. 2013. 3

[54] Philip Trettner and Leif Kobbelt. Fast and Robust QEF Minimization using Probabilistic Quadrics. *Computer Graphics Forum*, 39(2):325–334, May 2020. 3

[55] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, Mar. 2020. Number: 3 Publisher: Nature Publishing Group. 5

[56] Pengfei Wang, Zixiong Wang, Shiqing Xin, Xifeng Gao, Wenping Wang, and Changhe Tu. Restricted Delaunay Triangulation for Explicit Surface Reconstruction. *ACM Transactions on Graphics*, 41(5):1–20, Oct. 2022. 2

[57] Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual Octree Graph Networks for Learning Adaptive Volumetric Shape Representations. *ACM Transactions on Graphics*, 41(4):1–15, July 2022. arXiv:2205.02825 [cs]. 2

[58] Ji Wu, Huai Yu, Wen Yang, and Gui-Song Xia. QuadricsNet: Learning Concise Representation for Geometric Primitives in Point Clouds, Sept. 2023. arXiv:2309.14211 [cs]. 2

[59] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum*, 41(2):641–676, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14505. 1

[60] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry Processing with Neural Fields. In *Advances in Neural Information Processing Systems*, volume 34, pages 22483–22497. Curran Associates, Inc., 2021. 1

[61] Congyi Zhang, Guying Lin, Lei Yang, Xin Li, Taku Komura, Scott Schaefer, John Keyser, and Wenping Wang. Surface Extraction from Neural Unsigned Distance Fields. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22531–22540, 2023. 2

[62] 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 14418–14428, 2023. 3

[63] Tong Zhao, Laurent Busé, David Cohen-Steiner, Tamy Boubekeur, Jean-Marc Thiery, and Pierre Alliez. Variational Shape Reconstruction via Quadric Error Metrics. In *ACM SIGGRAPH 2023 Conference Proceedings*, Aug. 2023. 3

[64] Qingnan Zhou and Alec Jacobson. Thingi10K: A Dataset of 10,000 3D-Printing Models, July 2016. arXiv:1605.04797 [cs]. 6