Variational Shape Reconstruction via Quadric Error Metrics

Tong Zhao, Laurent Busé, David Cohen-Steiner, Tamy Boubekeur, Jean-Marc Thiery, Pierre Alliez

Variational Shape Reconstruction via Quadric Error Metrics

Tong Zhao
Inria, Université Côte d'Azur
Sophia Antipolis, France
LTCI, Télécom Paris
Palaiseau, France
tong.zhao@inria.fr

Laurent Busé
Inria, Université Côte d'Azur
Sophia Antipolis, France
laurent.buse@inria.fr

David Cohen-Steiner
Inria, Université Côte d'Azur
Sophia Antipolis, France
david.cohen-steiner@inria.fr

Tamy Boubekeur
Adobe Research
Paris, France
boubek@adobe.com

Jean-Marc Thiery
Adobe Research
Paris, France
jthiery@adobe.com

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

ABSTRACT

Inspired by the strengths of quadric error metrics initially designed for mesh decimation, we propose a concise mesh reconstruction approach for 3D point clouds. Our approach proceeds by clustering the input points enriched with quadric error metrics, where the generator of each cluster is the optimal 3D point for the sum of its quadric error metrics. This approach favors the placement of generators on sharp features, and tends to equidistribute the error among clusters. We reconstruct the output surface mesh from the adjacency between clusters and a constrained binary solver. We combine our clustering process with an adaptive refinement driven by the error. Compared to prior art, our method avoids dense reconstruction prior to simplification and produces immediately an optimized mesh.

CCS CONCEPTS

• **Computing methodologies** → **Mesh geometry models**; **Point-based models**.

KEYWORDS

Surface reconstruction, 3D point cloud, quadric error metrics, clustering, concise mesh reconstruction

ACM Reference Format:
Tong Zhao, Laurent Busé, David Cohen-Steiner, Tamy Boubekeur, Jean-Marc Thiery, and Pierre Alliez. 2023. Variational Shape Reconstruction via Quadric Error Metrics. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings (SIGGRAPH '23 Conference Proceedings), August 6–10, 2023, Los Angeles, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3588432.3591529

Author version

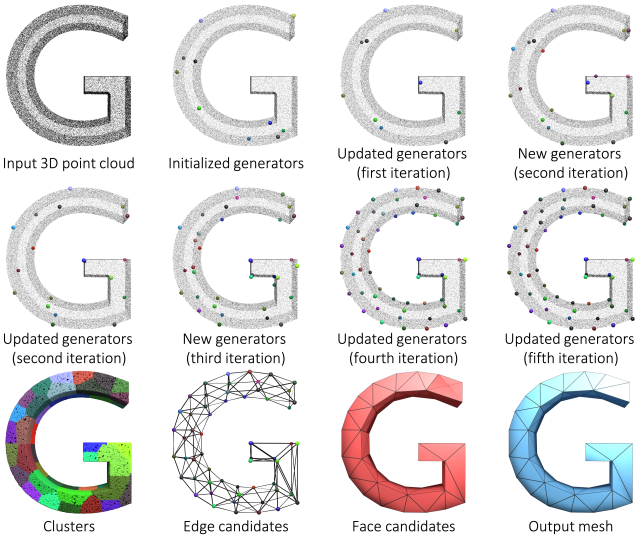


Figure 1: Variational shape reconstruction. The clustering of the points is randomly initialized, then alternates partitioning and generator updating. Some generators relocate to sharp features after one iteration. New generators are then added. The clustering converges after five iterations. A set of candidate edges is derived from the adjacency between clusters, and candidate facets (red) are generated. The output mesh is reconstructed via a constrained binary solver that selects a subset of the red facets.

1 INTRODUCTION

Mesh reconstruction consists in finding a mesh that piecewise approximates a point sampling of a 3D surface well. In addition to the inherent ill-posed nature of the reconstruction problem, several dilemmas witness the difficulty of this problem, such as interpolation vs. approximation, or greedy vs. variational approaches. In this paper, we focus on concise mesh reconstruction from 3D point clouds that are unstructured and come without oriented per-sample normal vector. In addition, we aim to produce concise surface triangle meshes through a low-memory footprint process, see Figure 1. One possible solution to the problem is to perform dense mesh reconstruction followed by mesh decimation but the transient memory consumption is large. If one seeks additional features such

as noise resilience or hole filling, one may resort to implicit reconstruction followed by isosurfacing and mesh decimation. However, this sequence adds another inconsistency since the bias induced by implicit reconstruction is unknown by the decimation step, resulting in a suboptimal complexity-distortion tradeoff. Proceeding coarse-to-fine has in general a lower memory footprint, but approaches such as greedy Delaunay refinement result in suboptimal approximations. This calls for a coarse-to-fine variational approach.

Another challenge comes from sharp features. Nothing is really sharp in the physical world when looking at fine scales. Nevertheless, a physical fillet (rounded edge) becomes sharp when zooming out at coarse scales. In a surface triangle mesh, any edge or vertex that is not flat is a sharp feature, but edges and vertices are also used to approximate smooth parts. In other words, the sharpness of a feature depends on the chosen scale or approximation tolerance error. As we wish to proceed coarse-to-fine, this rules out approaches based on early sharp feature detection. Instead, we favor a variational approach where sharp features emerge from an optimization process.

2 RELATED WORK

We first review the shape reconstruction problem, with a focus on generating concise meshes as output. We then review the pioneering work on quadric error metrics and its variants, and a few variational approaches.

2.1 Shape reconstruction

We focus next on key aspects sought after in our problem statement: conciseness and recovery of sharp features. We refer to books or surveys for a more complete review on shape reconstruction [Berger et al. 2017; Dey 2006].

Conciseness. Hoppe et al. [1994] pioneered a piecewise smooth reconstruction approach that inputs a dense reconstructed mesh, and interleaves decimation and optimization to fit a subdivision surface. Digne et al. [2014] formulate shape reconstruction as an optimal transport problem between a discrete measure (points seen as Dirac masses) and a dense simplicial complex seen as the support of a piecewise-constant measure, which is then decimated. Besides being compute-intensive, these fine-to-coarse approaches contradict our desire to proceed coarse-to-fine with low memory cost. The literature is sparser on coarse-to-fine mesh-based approaches. Sahillioğlu et al. [2010] start from the visual hull mesh of objects, and perform smoothing and restructuring operations. Delaunay refinement is another instance of coarse-to-fine paradigm [Campos et al. 2013], but for both above approaches the meshes are too isotropic to offer the level of conciseness sought after.

Sharp features. The literature is abounded by methods trying to detect sharp features, but we seek for an approach that *recovers* sharp features by optimizing vertex locations such that sharp features are formed where they offer the best approximation. Several non-linear regression approaches have been proposed to find such optimal locations: e.g., robust implicit moving least-squares surfaces (MLS) [Öztireli et al. 2009], or robust MLS [Fleishman et al. 2005]. Another approach consists in consolidating the point clouds before reconstruction, by removing outliers, reducing noise and relocating some points onto sharp features. Avron et al. [2010] first solve a global sparse minimization problem to consolidate normal orientations, then consolidate point locations before running ball pivoting [Bernardini et al. 1999]. Xu et al. [2022] pioneered a multistage approach that removes noise, identifies feature lines via discrete optimal transport and interpolates the consolidated point cloud. Consolidating the mesh after reconstruction is also an option. Attene et al. [2003] proposed a so-called *edge sharpener*. Wang et al. [2013] first remove outliers and noise, before inputting the consolidated point cloud to Poisson surface reconstruction [Kazhdan et al. 2006]. Sharp features are then recovered by applying bilateral filtering to the reconstructed surface mesh. Yadav et al. [2017] proposed another two-stage mesh denoising algorithm. All these methods are powerful but operate on dense meshes. Several learning-based methods [Liu et al. 2021; Matveev et al. 2022] have been recently proposed for extracting sharp feature graphs from raw point clouds.

2.2 Quadric Error Metrics

Garland and Heckbert [1997] pioneered a mesh decimation approach that applies a sequence of edge collapse operators sorted by so-called *quadric error metrics* (QEM). QEMs capture weighted sums of squared errors to the supporting planes of triangle facets (see Section 3). This powerful idea has become one of the most popular approaches for mesh decimation. Hoppe [1999] extended this approach to deal with colors and textures. Deng et al. [2011] utilized quadric error metrics to build cages for shape editing. Thiery et al. [2013] defined a spherical quadric error metric in 4D for extreme shape approximation. Salinas et al. [2015] contributed a structure-aware variant that adds to the quadrics a set of plane quadrics detected from the input mesh. Legrand et al. [2019] proposed a mesh smoothing and clustering approach via filtering quadrics: a field of filtered quadrics is defined on the input mesh by diffusing quadrics via a bilateral filter made of a spatial kernel and a QEM-based range kernel. The mesh vertices are connected to form a tree structure according to a relevance score estimated from the filtered quadric field, and the final clusters are generated by pruning the branches. Trettner and Kobbelt [2020] extended the QEM approach by defining probabilistic quadrics in closed form, in order to deal with uncertainty in the input meshes and improve resilience to noise and mesh quality.

2.3 Variational approaches

Cohen-Steiner et al. [2004] introduced the variational shape approximation (VSA) approach, which formulates the task as a discrete, variational partitioning problem with planar proxies. VSA optimizes clusters of connected mesh triangles by minimizing a one-sided error metric (L^2 or $L^{2,1}$) formulated between triangles and planar proxies. The number of clusters is adjusted via four operators (*merge*, *split*, *add*, *teleport*). The final output is an anisotropic polygonal mesh derived from the clustering partition. Wu and Kobbelt [2005] extended VSA to deal with higher order proxies such as spheres and cylinders. Skrodzki et al. [2020] adapted VSA to 3D point sets, and added a *switch* operator to guarantee convergence. Yu and Lafarge [2022] devised a pliant approach for partitioning a 3D point cloud into planar parts. From an initial partition, a multi-objective

function is optimized through five types of operators (*insert*, *exclude*, *transfer*, *merge*, *split*) to obtain a balance between fidelity, complexity and coverage. The optimized partition is then inputted to a piecewise-planar reconstruction approach [Bauchet and Lafarge 2020]. The above clustering approaches are powerful, but partitioning solves only half of the problem, as it remains to mesh the optimized partition which is in general composed of polygonal elements. Intersecting planes is an option, but it is numerically unstable, as already observed by Zimmer et al. [2012] for planar panelization of freeform surfaces. Meshing is also hampered by concave elements of the partition. Resorting to a volumetric partition through a kinetic-based approach is a reliable alternative, but is compute-intensive [Bauchet and Lafarge 2020]. Departing from partitioning with planar elements, we adopt a dual approach that partitions with "conical" elements, i.e. vertices with their adjacent planar elements where the vertex locations are computed through integrating quadrics over clusters.

2.4 Positioning and Contributions

The original QEM approach, designed for greedy mesh decimation, is fast and very effective. We observe that it may be seen as a clustering algorithm over the input mesh vertices, where the objective is to minimize the maximum (over the clusters) of the sum of squared distances from the optimal vertices to the set of planes of the clusters. This maximum is minimized in a greedy hierarchical way using a priority queue for decimation operators, and the output mesh is the dual, i.e., nerve of the clustering: one vertex per cluster, and one edge per pair of neighboring clusters.

We utilize the strengths of QEM to design a novel coarse-to-fine variational reconstruction algorithm. Our starting point is also QEM, but we replace the greedy fine-to-coarse clustering by a variational expectation-minimization (EM) approach. The objective function is no longer the maximum cluster error but rather the sum of errors over the clusters. Our method may also be seen as a form of dual to the VSA approach initially designed for mesh partitioning; instead of considering squared distances from data points to planar cluster proxies, we consider squared distances from tangent planes at data points to clusters estimated cone points. Our experiments show that, especially for extreme approximations, our approach outperforms the state-of-the-art. In addition, our method takes point clouds as input, allowing to address concise mesh reconstruction. For this harder problem, taking the nerve of the clustering as output mesh may not suffice anymore, and we output instead a mesh computed by a specific binary solver. Our main technical contributions are:

- A novel variational partitioning method designed to cluster the input points based on quadric error metrics. The generators are optimal points minimizing the sum of quadric error metrics for the clusters.
- The resulting partitioning tends to equidistribute the errors among clusters, yielding clusters with anisotropy in accordance to the local geometry.

3 BACKGROUND

We briefly review quadric error metrics (QEM) pioneered by Garland and Heckbert [1997] for mesh decimation, and some other variants such as the probabilistic QEM recently introduced by Trettner and Kobbelt [2020].

Given a triangle t, we consider its supporting plane π and denote by n a unit normal vector of π and by p an arbitrary point on π. The *plane quadric associated to* t is the 4×4 symmetric matrix

$$Q_t = \begin{bmatrix} nn^T & -p^T nn^T \\ -nn^T p & (p^T n)^2 \end{bmatrix} = \begin{bmatrix} A & -b \\ -b^T & c \end{bmatrix}. \tag{1}$$

Given a query point $q \in \mathbb{R}^3$, the corresponding error function is defined as $f(Q_t, q) = \tilde{q}^T Q_t \tilde{q}$, where $\tilde{q} = (q^T, 1)^T$ are the homogeneous coordinates of q. This function captures the squared distance from point q to plane π. The *probabilistic QEM* incorporates Gaussian noise for both the unit normal vector n and the position p on the plane π. In this case, A, b and c are replaced by their corresponding expectations $\mathbb{E}(A)$, $\mathbb{E}(b)$ and $\mathbb{E}(c)$.

A diffused quadric is assigned to each vertex of the input surface triangle mesh. Given a vertex v of a surface mesh, its *diffused quadric* Q_v is defined as the sum of the weighted quadrics of its 1-ring neighboring facets, i.e.

$$Q_v = \sum_{t_i \in \text{ neighbors(v)}} \frac{1}{3} a_{t_i} Q_{t_i}, \tag{2}$$

where a_{t_i} denotes the area of triangle t_i. Each plane quadric is thus evenly distributed on the three vertices of its triangle. From a geometric viewpoint, summing quadrics is equivalent to taking the union of the planes. Such an initialization guarantees that the vertices are the optimal minimizers of their diffused quadrics, while it is in general not the case during decimation. Equation (2) is extended to deal with boundaries by adding quadrics corresponding to planes that are orthogonal to boundary edges.

When an edge e connecting v_1 and v_2 is collapsed, the quadric of the new vertex v_e is computed as: $Q_{v_e} = Q_{v_1} + Q_{v_2}$, i.e., the collapsed vertex receives the contribution from all weighted planes of the input mesh adjacent to v_1 and v_2. The optimal location of v_e is computed by solving for point p_e^* that minimizes QEM $f(Q_{v_e}, p_e)$, i.e. the point which realizes the smallest sum of squared distances to all assigned planes. If all planes are similar then all points on the plane minimize $f(Q_{v_e}, p_e)$; if all planes intersect in a line, all points on this line minimize $f(Q_{v_e}, p_e)$; if all planes intersect at one point, the intersection point is the only minimizer of $f(Q_{v_e}, p_e)$. In other words, QEM has the virtue of placing points onto sharp creases or corners, while an infinite number of optimal point locations exist for planes or for linear sharp creases. Such an ill-posed problem is solved via singular value decomposition (SVD) [Lindstrom 2000], which computes the closest point in the optimal set from a given point - commonly the collapsed edge midpoint. There is thus an intrinsic relation between incremental decimation via QEM and fine-to-coarse vertex clustering. After initialization, each vertex is considered as an individual cluster. An edge collapse operation merges two clusters into a new cluster associated with the new optimal vertex. During decimation, the evolution of clusters can be tracked with a tree structure.

4 APPROACH

Figure 2 provides an overview of our approach. The input is a 3D point cloud sampled on a closed surface, and the output is a surface triangle mesh. A QEM is first estimated for each point.

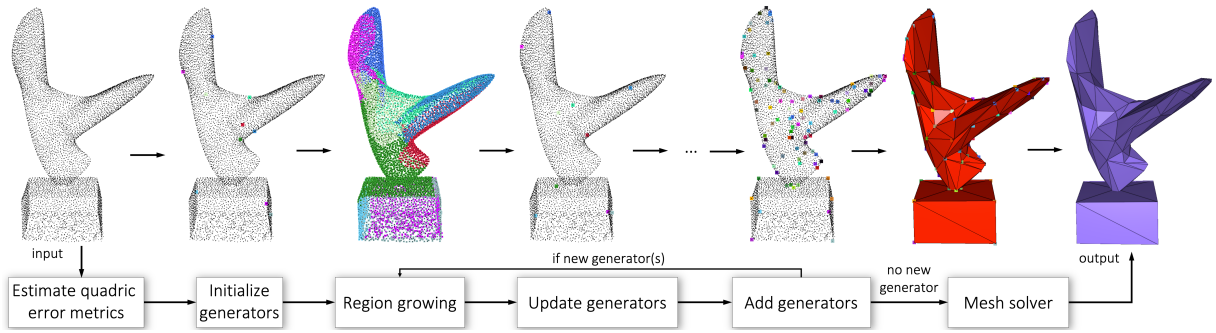


Figure 2: Overview: variational reconstruction via quadric error metrics.

Initial generators of clusters are chosen randomly from the input 3D points. The clustering is performed by alternating partitioning via region growing with updating the generators, until a maximum error tolerance is met for each cluster. Refinement is performed by adding batches of new generators where the errors are large. The output mesh is extracted by solving a constrained binary problem.

4.1 Quadric estimation

The original QEM approach designed for mesh decimation utilizes the mesh triangles to initialize a quadric per vertex, by summing plane quadrics on the 1-ring adjacent facets of each vertex. We first estimate a planar quadric per input point, based on a local normal and area estimation, and then compute a diffused quadric per point in order to capture its local geometry.

Denote by $\mathcal{P}$ the input 3D point cloud. Denote by p_i a point of $\mathcal{P} \in \mathbb{R}^3$ with normal $n_i \in \mathbb{R}^{3\times 1}$. We assume that normal n_i is either read from the input data, or estimated using a local normal estimation method. Normal orientations are not required. We compute the plane quadric of p_i as:

$$Q_{p_i} = \left(n_i^x, n_i^y, n_i^z, -n_i \cdot p_i^T\right)^T \cdot \left(n_i^x, n_i^y, n_i^z, -n_i \cdot p_i^T\right). \tag{3}$$

To compute the diffused quadrics, we first need to define a neighborhood and support area for each point. We choose by default the k-Nearest Neighbor (KNN) graph for both computations. The support area a_i for point p_i is estimated as:

$$a_{p_i} = \frac{1}{2k^2} \cdot \left(\sum_{p_j \mid (p_j, p_i) \in \mathrm{KNN}(\mathcal{P})} \|p_i - p_j\| \right)^2. \tag{4}$$

The diffused quadrics for a point p_i is then computed as:

$$Q_{v_i} = \sum_{p_j \mid (p_j, p_i) \in \mathrm{KNN}(\mathcal{P})} a_{p_j} \cdot Q_{p_j}. \tag{5}$$

Such a quadric reflects an approximation of the local geometry and point density. Figure 3 depicts the estimated diffused quadrics centered at each vertex of two point clouds, using 3D ellipsoids. Each ellipsoid represents a quadric error isosurface, that is the locus of points with equal errors. The center of the quadric realizes the minimum error. Intuitively, we can classify ellipsoids into three main types: pancakes, cigars and balls, corresponding to smooth areas, curved creases and corners.

The above estimation assumes that the points are sampled on a smooth surface. The initialization step is flexible and can also take into account additional geometric information that would be available as input, e.g. information on sharp features via multiple

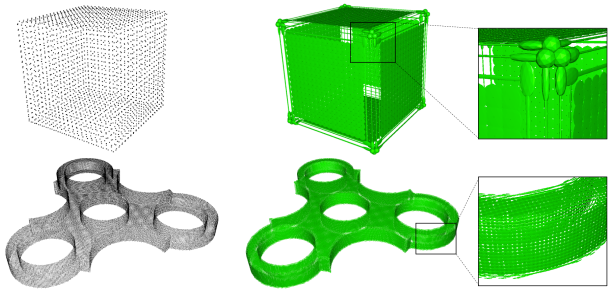


Figure 3: Quadric ellipsoids on two input point clouds.

local plane quadrics. However, the strength of our method is to avoid the ill-posed detection step, and yet to recover sharp creases even when using diffused quadrics that mollify the sharp features existing in the sampled physical object.

4.2 Initialization

Clustering of the input points is initialized by randomly selecting a handful of input points as initial cluster generators $\{c_i | 1 \le i \le m\}$. The generators coincide with the input points after initialization, but this is no longer the case later as they will be updated as optimal QEM 3D points.

4.3 Clustering

The clustering step operates by alternating partitioning with the generator updating until the clusters stabilize or a maximum number of iterations is reached.

Partitioning. This proceeds by region growing. More precisely, growing progresses on the KNN graph greedily one point at a time, with a priority queue that minimizes a cost E combining QEM and Euclidean distance. The cost of adding a point p_i to the cluster l_j is:

$$E(p_i, l_j) = [c_j, 1]^T \cdot Q_{v_i} \cdot [c_j, 1] + \lambda \cdot \|p_i - c_j\|^2, \tag{6}$$

where c_j denotes the current generator of cluster l_j and λ denotes a coefficient for regularizing region growing on planar areas. λ is set by default to a small value such that the coarse-to-fine refinement focuses mainly on minimizing the QEM error. There are three reasons for using such a Euclidean distance term: 1) to regularize partitioning of flat or straight areas by competing growing fronts, so that we obtain a Voronoi-like partition instead of noisy adjacency frontiers, 2) to obtain evenly distributed vertices on flat areas or along straight creases, when these areas are densely populated by generators, and 3) to provide enough edge candidates for meshing

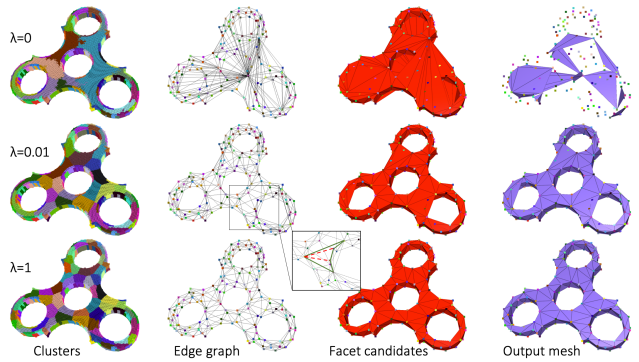
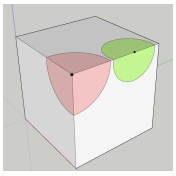


Figure 4: Partitioning and λ coefficient. On flat areas the QEM errors equate zero. This yields a clustering with noisy adjacency frontiers, and a deficient adjacency between clusters for the meshing step. Increasing λ improves the configuration, including for concave clusters.

concave clusters. To make the two metrics compatible, we set λ to be k times the squared average spacing of the input point cloud, because Q_{v_i} is computed as the sum of plane quadrics weighted by support areas. See Figure 4.

Once a point is labeled as l_j, all its unlabeled neighbors in the KNN graph are pushed to the priority queue with label j as new candidates. Such a partitioning proceeds until the queue is empty.

Note that in Equation (6), $Q_{v_i} \cdot [c_j, 1]$ measures an error that is oriented, i.e. realized by the generator point c_j of cluster l_j for the quadric of the candidate point to growing. Consider a 3D point set sampled on a perfect cube, with little noise. If a cluster has a generator point on a corner C of the said cube, then the error would be very small for the (almost flat pancake) quadrics of the candidate points sampled on the three faces adjacent to C, as well as for the (thin cigar) quadrics of the candidate points on the three creases adjacent to C (see inset figure, pink area). A similar behavior is observed when a cluster has a generator point on a crease point of the cube, but with two adjacent faces and creases (see inset figure, green area). Such a behavior is also similar for the tip of a cone, hence our reference to conical elements in Section 2.



Generator updating. In this step, the optimal generator c_j^* of each cluster l_j is recomputed as:

$$c_j^* = \underset{p \in \mathbb{R}^3}{\arg\min} [p, 1]^T \cdot Q_{c_j} \cdot [p, 1]. \tag{7}$$

More specifically, c_j^* is obtained by solving the linear system below:

$$Ac^* = \begin{pmatrix} Q^{11} & Q^{12} & Q^{13} & Q^{14} \\ Q^{12} & Q^{22} & Q^{23} & Q^{24} \\ Q^{13} & Q^{23} & Q^{33} & Q^{34} \\ 0 & 0 & 0 & 1 \end{pmatrix} c^* = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{8}$$

When A is not invertible, c^* is computed by a SVD solver.

Note that the next clustering iteration, and in particular its partitioning step that proceeds by region growing over the KNN graph, requires each updated generator c_j^* to be associated with an input point in $\mathcal{P}$. So each generator c_j^* is associated to its nearest input point in cluster l_j. Only the optimal points c^* are later used for constructing the output surface triangle mesh. Clustering terminates when there is no change between the old and new set of generators, or when a user-selected maximum number of iterations is reached.

4.4 Batch splitting

The maximum QEM error is monitored for all points of a cluster, once clustering terminates. We operate a splitting operator that inserts a new generator in the cluster realizing the maximum error. More specifically, for each cluster l_j, we find the point labeled j that maximizes Q_{c_j}, i.e.

$$p_{\max}(l_j) = \underset{p_i \in l_j}{\arg\max} [p_i, 1]^T \cdot Q_{c_j} \cdot [p_i, 1]. \tag{9}$$

If the corresponding QEM exceeds a maximum user-defined tolerance, then $p_{\max}(l_j)$ is added as a new candidate for cluster splitting. We then greedily compute an independent set of clusters to be split, enforcing that two adjacent clusters cannot be split in the same splitting batch to avoid over-refining. As it is not intuitive to define a meaningful QEM tolerance, we take a Euclidean distance tolerance parameter and convert it to a QEM scale according to the relationship between these two metrics (see Section 3). Alternating between partitioning and batch splitting stops when all clusters satisfy the user-defined tolerance error or when a maximum number of iterations is reached.

4.5 Meshing

The current partition of clusters yields a set of vertices that are optimal in the QEM sense. It remains to connect these vertices by finding a set of triangle facets. We first derive a graph of edges from the adjacency between clusters, from which we compute facet candidates and then solve a constrained binary program (CBP) to extract the output surface triangle mesh.

Edge graph. After partitioning, a graph of edge candidates is derived from the adjacency between clusters in the KNN graph of the input points.

Facet candidates. Given the above edge graph, we search for 3-cycles in the graph by selecting all triplets of vertices that are mutually connected by an edge. This yields a set of triangle facet candidates, possibly overlapping in some areas. It remains to select a subset of these facets.

Mesh extraction. From the set of facet candidates, we wish to keep the ones that fit well, cover well the input 3D point cloud and favor a 2-manifold mesh. Building upon the PolyFit approach [Nan and Wonka 2017], we minimize an objective function rewarding a data fitting term F_f and a data coverage term F_c, under a 2-manifold constraint. More specifically, we assign a binary label b_{f_i} for each facet candidate f_i and a binary label b_{e_i} for each edge e_i. Label one indicates that the facet or edge is kept for the final mesh, and label zero indicates that it is discarded. We then minimize:

$$\max_{\{b_{f_1}, \ldots, b_{f_n}\}} \sum_{i=1}^{n} b_{f_i} \cdot \Big(F_f(f_i) + F_c(f_i) \Big) \tag{10}$$

$$\text{s.t.} \quad 2b_{e_i} - \sum_{f_j \text{ around } e_i} b_{f_j} = 0, \quad \forall e_i. \tag{11}$$

Figure 2 (right) illustrates facet filtering: some quads overly covered by four red facets are later triangulated with two triangles in the output mesh. When the 2-manifold property is not sought after, we can deactivate the constraints and search instead for a balance between high fitting, high coverage and low complexity of output. We can then reconstruct non-manifold meshes with boundaries (see Figure 13) by simply selecting facets such that $F_f(f_i)+F_c(f_i)-1 > 0$.

Fitting term. For each facet candidate, we find all input points whose distance to the triangle facet is smaller than a tolerance error ϵ and compute the fitting term as:

$$F_f(f_i) = \sum_{p_j | d(p_j,f_i)<\epsilon} \left(1 - \frac{d(p_i,f_j)}{\epsilon}\right). \tag{12}$$

Coverage term. Coverage prevents large triangles from covering empty areas. For each facet candidate, the ϵ-selected input points are projected onto its supporting plane. We then construct a 2D alpha shape with alpha set to s times the average spacing of $\mathcal{P}$ and compute the ratio between the total area of the triangles of the alpha shape, and the area of the facet. s is by default set to 5.

$$F_c(f_i) = \min\left(1, \frac{\text{area}\big(\alpha(\{p_j | d(p_j,f_i) < \epsilon\})\big)}{\text{area}(f_i)}\right). \tag{13}$$

Manifold constraint. To favor 2-manifold output meshes, a constraint is added for each edge e_i such that if it is kept by the solver ($e_i = 1$), then exactly two triangles adjacent to e_i must be kept, and zero adjacent triangles otherwise (Equation (11)).

5 EXPERIMENTS

We implemented our approach in C++, using several libraries: CGAL [The CGAL Project 2021] for geometric computations, Eigen [Guennebaud et al. 2010] for linear algebra and SCIP [Bestuzheva et al. 2021] for solving binary linear problems. All experiments are conducted on a MacBook Pro with a 2.9GHz Quad-Core Intel i7 CPU and 16GB memory, running on a single core.

Figure 5 validates our concise reconstruction approach on a 3D point set sampled on a capsule made up of an open cylinder and two half-spheres. The clustering algorithm generates elongated clusters on parabolic areas and isotropic clusters on the spherical areas, that translate into skinny and isotropic triangles, respectively.

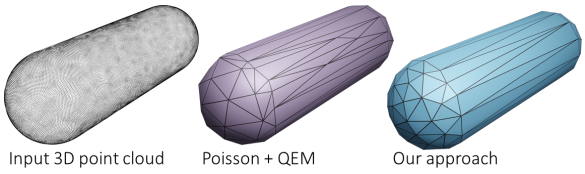


Figure 5: Reconstructing a capsule. Left: input 3D point cloud. Middle: Poisson+QEM with 83 vertices. Right: our reconstruction with 83 vertices.

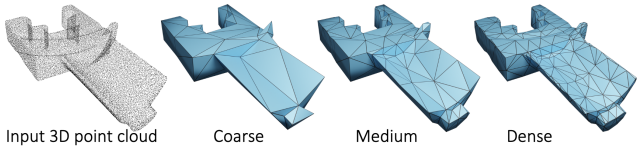


Figure 6: Reconstructing a blade. Left: input 3D point cloud. Right: reconstructions with increasing mesh complexity.

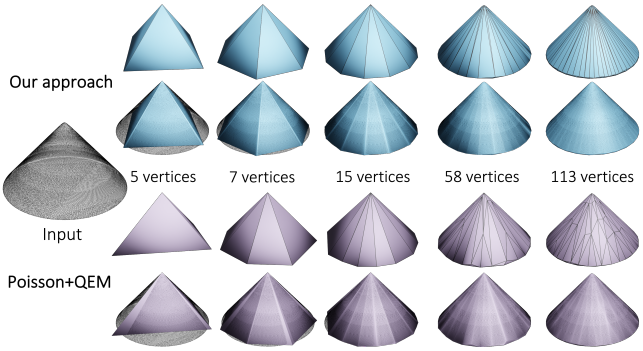


Figure 7: Reconstructing a smoothed circular cone. Left: input 3D points. Top: our reconstruction with increasing resolution. Meshes are shown with or without the input points. Bottom: Poisson reconstruction followed by QEM-based mesh decimation.

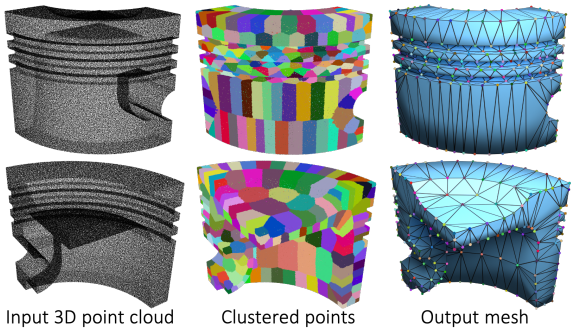


Figure 8: Reconstructing a piston. Left: input 3D point cloud. Middle: input points after clustering. Right: output mesh, and vertices colored by their corresponding clusters. The vertices are preferably located on sharp corners and creases.

Figure 1 illustrates our approach on a point cloud sampled on a "G" letter, containing curved sharp creases. The variational clustering approach tends to equidistribute the errors among clusters where possible, translating into evenly placed vertices along the creases. Figure 6 shows three concise reconstructions from a point cloud sampled on a mechanical part. See our supplementary videos.

Figure 7 highlights the difference between variational partitioning and dense Poisson reconstruction followed by greedy QEM decimation [Garland and Heckbert 1997; Kazhdan and Hoppe 2013], when dealing with smooth creases. Creases should be reconstructed as sharp edges at coarse scales, and as smooth edges at finer scales. The input point set is sampled on a circular cone without a boundary, with a smoothed tip and crease at the bottom. As well as being low-memory, our reconstruction is more symmetric than the greedy approach, reflecting the error equidistribution among clusters. The tip remains as a unique isolated corner for 15 vertices, then only the smoothed tip and crease start being refined.

Figure 8 shows our method at work on more diverse sharp features, including sharp creases subtending small angles. The cluster anisotropy reflects the local geometry and the generators are located on sharp corners, creases and curved areas. Other reconstructions with two resolutions are shown Figure 14.

Figure 9 plots timings and peak memory consumption against number of input points or complexity of the output mesh. The

Table 1: Maximum distances from 3D point clouds to output meshes. P+Q: Poisson followed by QEM-based decimation.

Model	Capsule	Cone (#v:5)	Cone (#v:7)	Cone (#v:15)	Cone (#v:58)	Cone (#v:113)	Hilbert	Hilbert (0.5%)	Hilbert (1%)
Max (P+Q)	0.07489	0.3836	0.1078	0.06092	0.01455	0.008757	1.6725	1.7233	1.5873
Max (Ours)	0.03256	0.2561	0.08905	0.01721	0.007488	0.004118	0.1729	1.0565	2.0487
Model	**Bunny**	**Part**	**Hand**	**Elephant (Middle)**	**Elephant (Right)**	**Mother (Middle)**	**Mother (Right)**	**Rocker arm (Middle)**	**Rocker arm (Right)**
Max (P+Q)	0.00328	2.145	0.01345	0.04299	0.01088	4.8591	1.3958	0.002321	0.001147
Max (Ours)	0.002055	2.1063	0.008587	0.0232	0.007387	4.5793	1.1293	0.001209	0.0005845

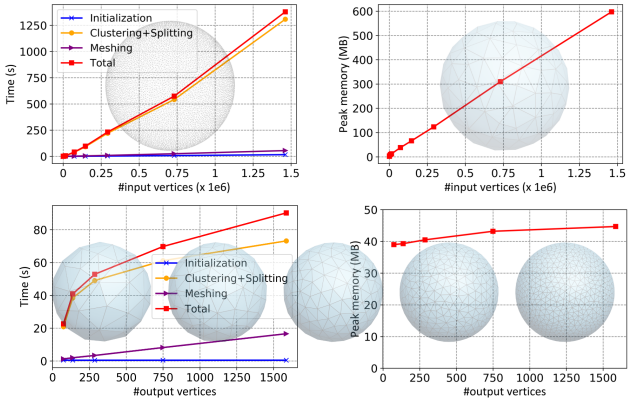


Figure 9: Timing and peak memory. Top: the number of input points increases from 732 to 1M, while the complexity of the output mesh is constant (150 vertices). Bottom: the number of input points is constant (73k points) while the complexity of the reconstructed mesh increases.

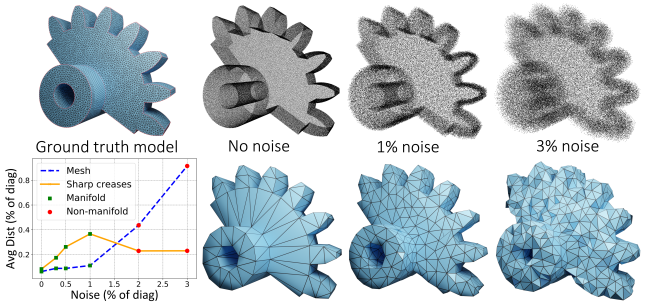


Figure 10: Robustness to noise.

input point cloud is uniformly sampled on a sphere. The memory consumption is only relative, as our current implementation is interactive and the rendering part consumes a large memory fraction of the whole program.

Table 1 reports maximum Euclidean distances from 3D point clouds to output meshes, for comparing with Poisson followed by QEM-based decimation. Blue cells indicates smaller values.

Figure 17 and 18 show visual comparisons. We compare our method with Poisson reconstruction, Poisson followed by QEM-based decimation and VSA, kinetic shape reconstruction [Bauchet and Lafarge 2020], partitioning followed by kinetic shape reconstruction [Yu and Lafarge 2022], an interpolant Delaunay-based approach [Rakotosaona et al. 2021], a recent feature-aware shape reconstruction RFEPS [Xu et al. 2022], and RFEPS followed by QEM-based decimation and VSA. Note that the last two approaches and our approach make no assumption about normal orientation, while the others require oriented normals. We can match the exact same mesh complexity only for Poisson followed by QEM decimation, as the other methods deal with planar shapes before meshing. For the Stanford bunny, the differences between the two meshes with the same complexity are subtle but noticeable by comparing the shading of the triangles with the one of the dense Poisson reconstruction (e.g., nose, crease of the neck and tail). The ear is discretized differently, with a vertex located on a saddle point.

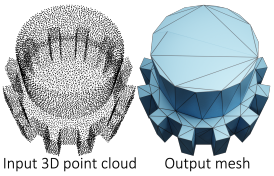
Figure 10 evaluates robustness to noise on dense point clouds sampled on a CAD model, altered with random uniform noise with different magnitude–up to 3% of the bounding box diagonal. We measure: (1) the average distance from a densely sampled reconstructed mesh to the ground truth, (2) the average distance from densely sampled ground truth sharp creases to the reconstructed mesh and (3) the manifoldness of the output mesh. Figure 16 illustrates robustness to noise on a genus 120 model: reconstruction degrades gracefully when noise increases. Figure 11 compares our method with kinetic shape reconstruction [Bauchet and Lafarge 2020], and with Poisson reconstruction followed by QEM decimation, on a piecewise-planar Hilbert cube with increasing random uniform noise. Figure 12 shows results on a point cloud generated by photogrammetry, with a large amount of noise and outliers.

Figure 15 illustrates the stability of the clustering step and approximation errors with respect to four different random initializations. Some salient features such as the saddle in the palm are present in the four reconstructions.

5.1 Limitations

Over-refinement. Batch splitting steps may over-refine some areas, despite the independent set, so that some straight creases may be overly discretized. A richer set of operators (e.g., join or switch) may help but can lead to looping between refinement and coarsening of the partitioning. We leave this for future work.

Fold-overs. The constraints of the binary solver guarantee that each edge is adjacent to either 0 or 2 facets, making the output mesh orientable and free of non-manifold edges. Nevertheless, it does not prevent from having two selected faces around an edge nearly co-planar with opposite orientations (see inset figure - the output mesh contains a fold-over). A larger value for λ parameter (distance for region growing) helps to reduce the problem but does not solve it entirely. In addition, we cannot get rid of non-manifold vertices.



Boundaries. Clustering does not place generators on boundary curves, unless boundary points are first detected and orthogonal quadrics are added there. This contradicts our initial desire to avoid resorting to a detection step. Our approach is designed to deal only with point clouds sampled on closed objects. The solver can fill some holes via the 2-manifold constraints, but only when there are enough candidate facets filling these holes. Deactivating the

constraints is feasible (see Figure 13) but the 2-manifold property is no longer favored.

6 CONCLUSION

We proposed a variational shape reconstruction method leveraging the strengths of two concepts: quadric error metrics (QEM) and variational partitioning. Our approach can deal with unoriented 3D point clouds as input and is designed for the reconstruction of concise triangle meshes. Most previous variational approaches optimize planar elements before determining vertices and then a connectivity to form triangles or polygons. As the optimized errors often refer to infinite planes and the elements may be concave, determining vertices and meshing is difficult. We adopt a dual approach and optimize the placement of optimal cone points of clusters, that are later connected to form the output triangle mesh. Intuitively, this brings us one step closer to the final discretization, with improved consistency that translates into lower approximation errors and better recovery of sharp features and symmetries.

In future work, we plan to extend our approach so that it can discover open boundaries during clustering, while favoring the 2-manifold property. We will also explore alternative solvers for the final mesh extraction steps, so that users can better trade complexity for distortion, or balance with other objectives such as the shape of the mesh elements. Finally, we will extend this approach in order to reconstruct piecewise-smooth surfaces with curved Bézier triangles.

ACKNOWLEDGMENTS

This work has been supported by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002.

REFERENCES

Marco Attene, Bianca Falcidieno, Jarek Rossignac, and Michela Spagnuolo. 2003. Edge-Sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Aachen, Germany) *(SGP '03)*. Eurographics Association, Goslar, DEU, 62–69.

Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. 2010. l1-Sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics* 29, 5 (2010), 1–12.

Jean-Philippe Bauchet and Florent Lafarge. 2020. Kinetic shape reconstruction. *ACM Transactions on Graphics* 39, 5 (2020), 14 pages. https://doi.org/10.1145/3376918

Matthew Berger, Andrea Tagliasacchi, Lee M. Seversky, Pierre Alliez, Gaël Guennebaud, Joshua A. Levine, Andrei Sharf, and Claudio T. Silva. 2017. A survey of surface reconstruction from point clouds. *Computer Graphics Forum* 36, 1 (2017), 301–329.

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. 1999. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics* 5, 4 (1999), 349–359.

Ksenia Bestuzheva, Mathieu Besançon, et al. 2021. *The SCIP optimization suite 8.0.* Technical Report. Optimization Online. http://www.optimization-online.org/DB_HTML/2021/12/8728.html

Ricard Campos, Rafael Garcia, Pierre Alliez, and Mariette Yvinec. 2013. Splat-based surface reconstruction from defect-laden point sets. *Graphical Models* 75, 6 (Nov. 2013), 346–361.

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. *ACM Transactions on Graphics* 23, 3 (2004), 905–914.

Zheng-Jie Deng, Xiao-Nan Luo, and Xiao-Ping Miao. 2011. Automatic cage building with quadric error metrics. *Journal of Computer Science and Technology* 26 (2011), 538–547.

Tamal K. Dey. 2006. *Curve and surface reconstruction: Algorithms with mathematical analysis.* Cambridge University Press, Cambridge.

Julie Digne, David Cohen-Steiner, Pierre Alliez, Fernando de Goes, and Mathieu Desbrun. 2014. Feature-preserving surface reconstruction and simplification from defect-laden point sets. *Journal of Mathematical Imaging and Vision* 48, 2 (2014), 369–382.

Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. 2005. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics* 24, 3 (2005), 544–552.

Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, Los Angeles, 209–216.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org

Hugues Hoppe. 1999. New quadric metric for simplifiying meshes with appearance attributes. In *Proceedings of the Conference on Visualization '99: Celebrating Ten Years* (San Francisco, California, USA) *(VIS '99)*. IEEE Computer Society Press, Washington, DC, USA, 59–66.

Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. 1994. Piecewise smooth surface reconstruction. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. ACM, Orlando, 295–302.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Symposium on Geometry Processing*, Alla Sheffer and Konrad Polthier (Eds.). The Eurographics Association, Cagliari, Sardinia, 10 pages.

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson surface reconstruction. *ACM Transactions on Graphics* 32, 3, Article 29 (jul 2013), 13 pages.

Hélène Legrand, Jean-Marc Thiery, and Tamy Boubekeur. 2019. Filtered quadrics for high-speed geometry smoothing and clustering. *Computer Graphics Forum* 38, 1 (2019), 663–677.

Peter Lindstrom. 2000. Out-of-core simplification of large polygonal models. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, New Orleans, 259–262.

Yujia Liu, Stefano D'Aronco, Konrad Schindler, and Jan Dirk Wegner. 2021. PC2WF: 3D wireframe reconstruction from raw point clouds. *arXiv preprint arXiv:2103.02766* (2021).

Albert Matveev, Ruslan Rakhimov, Alexey Artemov, Gleb Bobrovskikh, Vage Egiazarian, Emil Bogomolov, Daniele Panozzo, Denis Zorin, and Evgeny Burnaev. 2022. DEF: Deep estimation of sharp geometric features in 3D shapes. *ACM Transactions on Graphics* 41, 4 (2022), 1–22.

Liangliang Nan and Peter Wonka. 2017. Polyfit: Polygonal surface reconstruction from point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, Venice, 2353–2361.

Cengiz Öztireli, Gael Guennebaud, and Markus Gross. 2009. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum* 28, 2 (2009), 493–501.

Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy Mitra, and Maks Ovsjanikov. 2021. Learning Delaunay surface elements for mesh reconstruction. In *CVPR - IEEE Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 22–31.

Yusuf Sahillioglu and Yücel Yemez. 2010. Coarse-to-fine surface reconstruction from silhouettes and range data using mesh deformation. *Computer Vision and Image Understanding* 114, 3 (2010), 334–348.

David Salinas, Florent Lafarge, and Pierre Alliez. 2015. Structure-aware mesh decimation. *Computer Graphics Forum* 34, 6 (2015), 211–227.

Martin Skrodzki, Eric Zimmermann, and Konrad Polthier. 2020. Variational shape approximation of point set surfaces. *Computer Aided Geometric Design* 80 (2020), 101875.

The CGAL Project. 2021. CGAL user and reference manual. https://doc.cgal.org/5.2.1/Manual/packages.html

Jean-Marc Thiery, Émilie Guy, and Tamy Boubekeur. 2013. Sphere-Meshes: Shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics* 32, 6, Article 178 (nov 2013), 12 pages.

Philip Trettner and Leif Kobbelt. 2020. Fast and robust QEF minimization using probabilistic quadrics. *Computer Graphics Forum* 39, 2 (2020), 325–334.

Jun Wang, Z Yu, W Zhu, and J Cao. 2013. Feature-preserving surface reconstruction from unoriented, noisy point data. *Computer Graphics Forum* 32, 1 (2013), 164–176.

Jianhua Wu and Leif Kobbelt. 2005. Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum* 24, 3 (2005), 277–284.

Rui Xu, Zixiong Wang, Zhiyang Dou, Chen Zong, Shiqing Xin, Mingyan Jiang, Tao Ju, and Changhe Tu. 2022. RFEPS: Reconstructing feature-line equipped polygonal surface. *ACM Transactions on Graphics* 41, 6, Article 228 (nov 2022), 15 pages.

Sunil Kumar Yadav, Ulrich Reitebuch, and Konrad Polthier. 2017. Robust and high fidelity mesh denoising.

Mulin Yu and Florent Lafarge. 2022. Finding good configurations of planar primitives in unorganized point clouds. In *CVPR - IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, La Nouvelle-Orléans, United States, 11 pages.

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A dataset of 10,000 3D-printing models. *arXiv preprint arXiv:1605.04797* (2016).

Henrik Zimmer, Marcel Campen, Ralf Herkrath, and Leif Kobbelt. 2012. Variational tangent plane intersection for planar polygonal meshing. In *Advances in Architectural Geometry*. Springer, Paris, 14 pages.

7 GALLERY

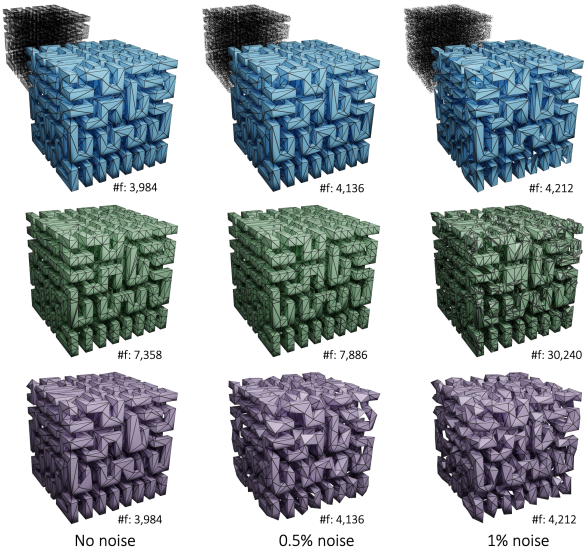


Figure 11: Robustness to noise on the Hilbert cube. Top: our method. Middle: output of kinetic shape reconstruction. Bottom: output of Poisson reconstruction followed by QEM-based mesh decimation. #f: number of faces.

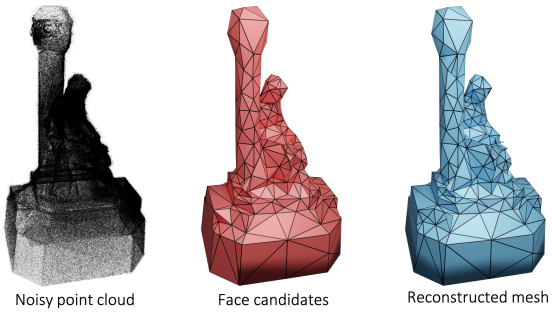


Figure 12: Left: point cloud generated by photogrammetry, with noise and outliers. We estimated point normals on the 100 nearest neighbors in order to smooth the normal field. Middle: facet candidates deduced from clustering. Right: reconstructed mesh.

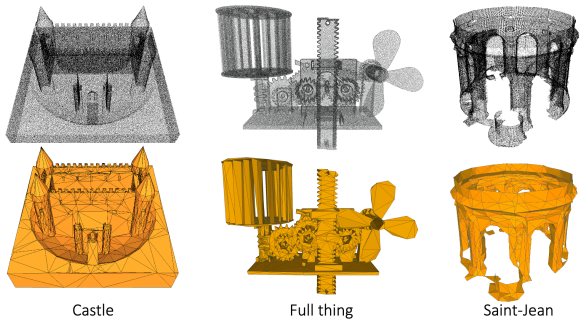


Figure 13: Non-manifold mesh reconstruction. By replacing the manifold constraint in the mesh solver by an objective function which rewards only low complexity of the output mesh, we can reconstruct meshes with boundaries and non-manifold edges. The output is a triangle soup.

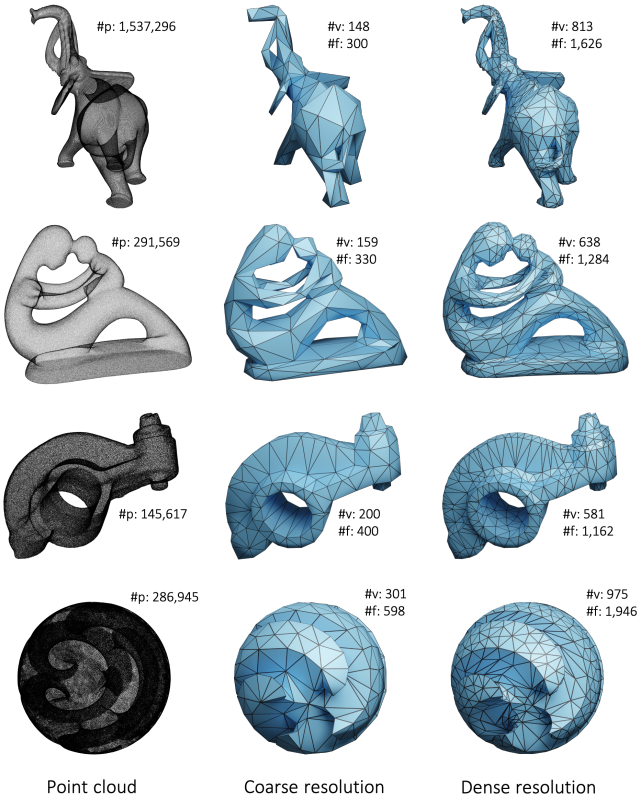


Figure 14: Meshes reconstructed with two resolutions. #p: number of points. #f: number of faces. #v: number of vertices. From top to bottom: *Elephant, Mother, Rocker arm, Sharp sphere.*

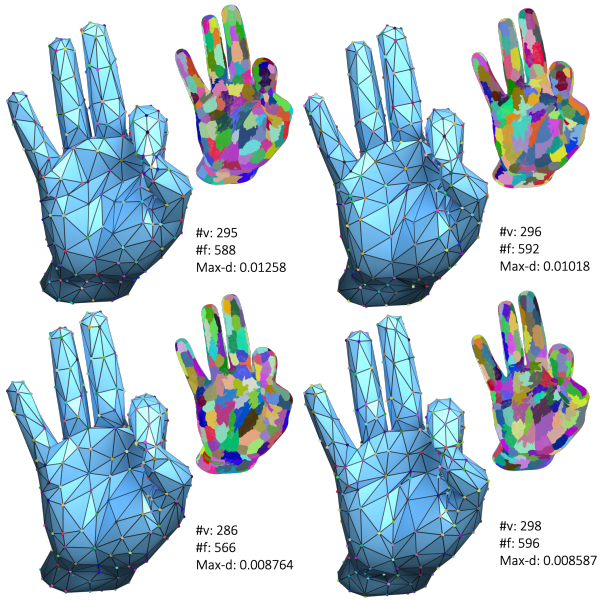


Figure 15: Meshes reconstructed from four different initializations.

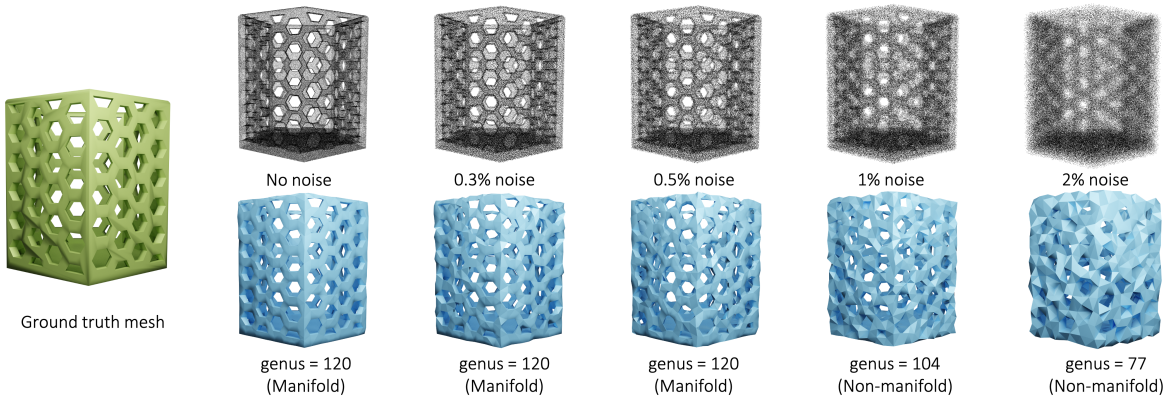


Figure 16: Robustness to noise on a genus 120 hex pen pot (taken from Thingi10K [Zhou and Jacobson 2016]).

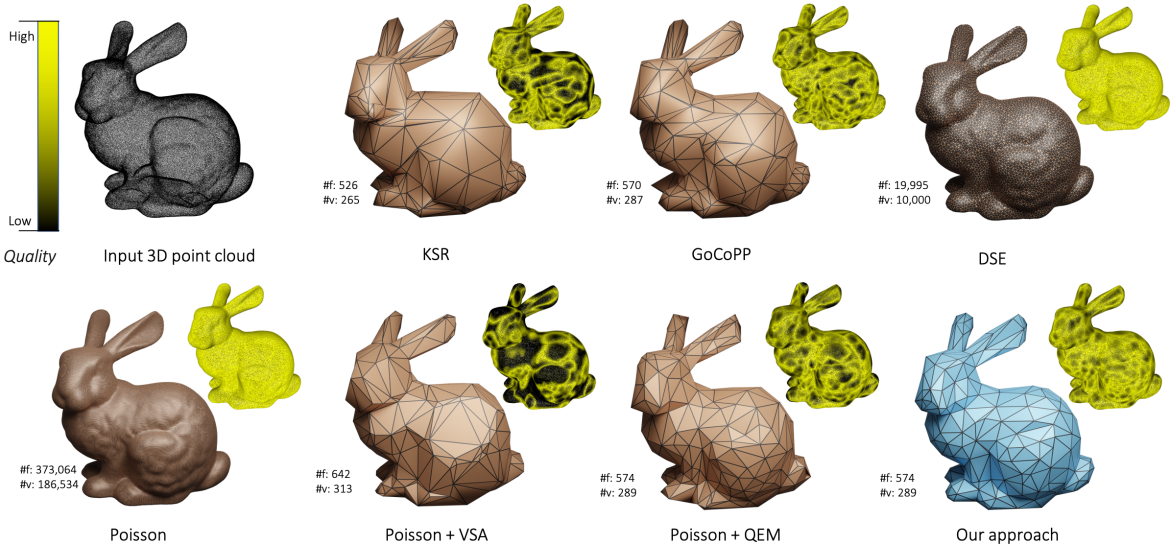


Figure 17: Comparisons on the bunny. KSR: Kinetic shape reconstruction [Bauchet and Lafarge 2020]. DSE: Delaunay surface elements [Rakotosaona et al. 2021]. GoCoPP: Good configurations of planar primitives [Yu and Lafarge 2022].

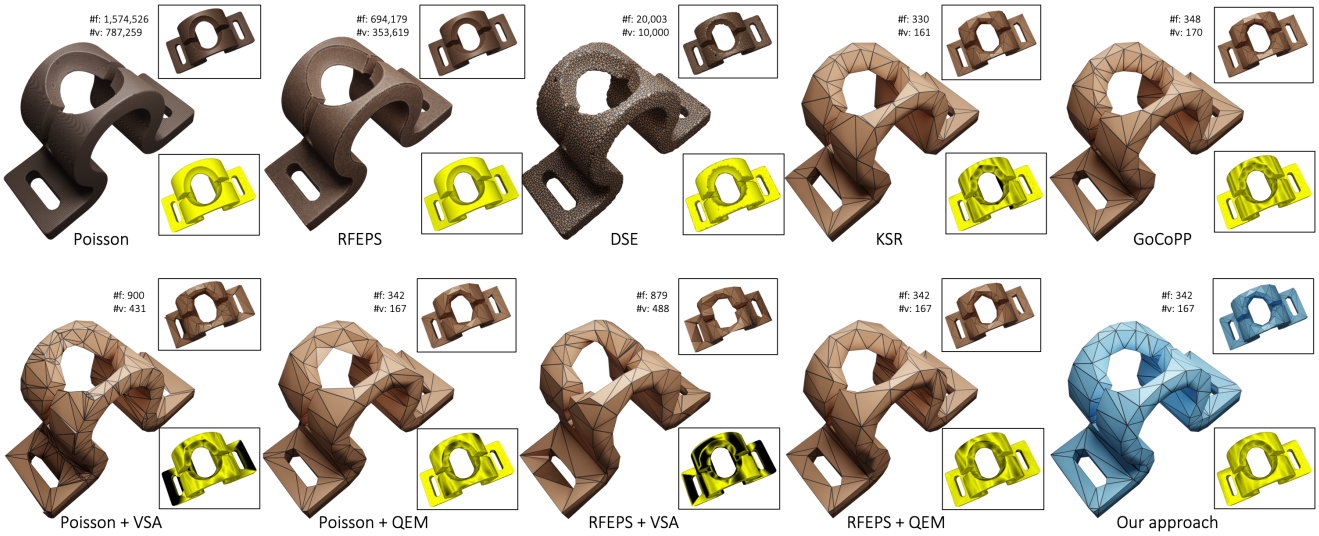


Figure 18: Comparisons on a mechanical part (point cloud taken from RFEPS [Xu et al. 2022]).