Delaunay Mesh Generation of Three Dimensional Domains *

Tamal K. Dey†

Abstract

Delaunay meshes are used in various applications such as finite element analysis, computer graphics rendering, geometric modeling, and shape analysis. As the applications vary, so do the domains to be meshed. Although meshing of geometric domains with Delaunay simplices have been around for a while, provable techniques to mesh various types of three dimensional domains have been developed only recently. We devote this article to presenting these techniques. We survey various related results and detail a few core algorithms that have provable guarantees and are amenable to practical implementation. Delaunay refinement, a paradigm originally developed for guaranteeing shape quality of mesh elements, is a common thread in these algorithms. We finish the article by listing a set of open questions.

*Research supported by NSF, USA (CCF-0430735 and CCF-0635008).

†Department of Computer Science and Engineering, Ohio State University, Ohio, USA. Email: `tamaldey@cse.ohio-state.edu`

1 Introduction

The need for meshing geometric domains in three dimensions is ubiquitous in scientific studies and engineering applications. Although a vast literature exists on mesh generation, there are not many results that provide theoretical guarantees about output. In unstructured meshing with triangles and tetrahedra, mainly two approaches are known that come with these guarantees; one is octree based [5, 45] and the other is Delaunay based [9, 32]. In this survey we focus on Delaunay based methods. The literature on Delaunay based meshing is huge and it is not our intention to survey all existing algorithms on the subject. We recommend other literature [6, 12, 46, 49] as a complement to ours. We focus on various provable techniques that enable meshing of different types of geometric domains in three dimensions. As a common theme we illustrate how an elegant algorithmic paradigm called *Delaunay refinement* can be adapted to the variety of domains.

Issues: There are two major issues in Delaunay mesh generation of geometric domains: (i) conformation and (ii) element quality.

If the input consists of linear elements such as segments, planar facets, and polyhedra, conformation means that each of these input elements appears in the output as a union of Delaunay simplices. When the input is curved, one cannot impose this condition since a curved element cannot be decomposed into finitely many linear elements. Therefore, in this case we require that the underlying space of the output mesh have the same topology as that of the input. This means, for example, a meshing of a sphere should be a triangulated sphere and not a triangulated torus. Since topology does not capture the geometry of a shape, it is also required that the geometry of the input be approximated well.

The shapes of the elements influence the approximation and numerical errors in finite element methods. The quality of the shape is determined by various measures. For a triangle, the minimum angle is a good measure of its quality. For a tetrahedron, the minimum or maximum over all face angles, dihedral angles, and solid angles is a good measure of its quality. Actually, the requirement on quality of simplices depends on the application. See Knupp [41] for various measures of quality and also the article by Shewchuk [53] for implications of different measures. In this article, we will use the well known radius-edge ratio measure that enjoys many nice properties [44] and suits Delaunay refinement techniques very well. For a triangle, a bound on radius-edge ratio imposes both an upper and a lower bound on its angles and thus forces it to be well shaped. For a tetrahedron an upper bound on radius-edge ratio does put bounds on face angles but fails to do the same for dihedral and solid angles. Indeed, an upper bound on radius-edge ratio eliminates all types of bad tetrahedra except one, the notorious slivers [15]. Slivers are tetrahedra that reside very close to a diametric plane of its circumscribing ball and have all four vertices more or less equi-spaced. See Figure 1.

Background: Delaunay refinement was pioneered by Chew [24] who applied it to meshing a point set in two dimensions with a quality guarantee for triangles. Chew showed that, by inserting Voronoi vertices which are also circumcenters of Delaunay triangles, one can guarantee a minimum angle of $30°$. In a novel extension Ruppert [50] showed how the Delaunay refinement strategy can be adapted to conform to line segments and points in the plane. His analysis showed that the Delaunay refinement produces a mesh that has a size within constant factor of the optimal.

In three dimensions Delaunay refinement was first applied to mesh convex polyhedra [29]. Shewchuk [51] showed how to apply it to polyhedral complexes. He made novel observations to

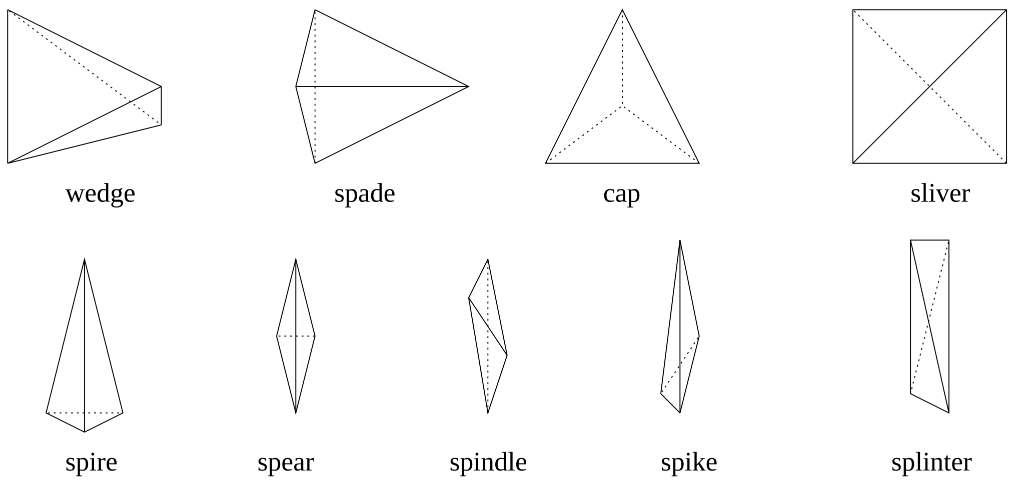


Figure 1: Tetrahedra with poor radius-edge ratio, taxonomy borrowed from [15]. All except the sliver have bad radius-edge ratio.

extend the analysis of Ruppert to this case. Although this is the first Delaunay refinement algorithm for a non-trivial domain in three dimensions, the algorithm could not handle polyhedral complexes with input angles smaller than $\frac{\pi}{2}$. Acute angles were known to be a menace for Delaunay refinement in two dimensions [50, 52]. Naturally they posed a harder challenge in three dimensions. Shewchuk [52] proposed a variant called the constrained Delaunay triangulation to deal with acute angles. However, refinements with constrained Delaunay triangulations do not produce Delaunay meshes. Shi and Gärtner [54] report that when constrained Delaunay triangulation is used as a preprocessor, Delaunay refinement can handle acute angles in practice. Cohen-Steiner, de Verdiére, and Yvinec [27] proposed a meshing algorithm with Delaunay simplices for polyhedral complexes allowing acute input angles. This algorithm only addressed the conformity issue but not the quality issue. Cheng and Poon [21] and Pav and Walkington [48] proposed algorithms that addressed both issues. Unfortunately, these algorithms are not very practical. Cheng, Dey, Ramos, and Ray [22] designed a Delaunay refinement algorithm for polyhedra with possibly acute angles which could be implemented in practice. They guarantee that all simplices except the ones near acute angles have bounded radius-edge ratio. A novelty of the algorithm is that it does not explicitly compute the intersection of a set of protecting balls with the polyhedron which was a bottleneck in Cheng and Poon [21]. We describe the algorithms of Shewchuk [51] and Cheng et al. [22] in section 2.

For polyhedral domains, topology preservation is not an issue since the output mesh can conform to the input exactly. For curved domains, this is not the case. Capturing the topology of the input is a foremost concern for them. Chew was the first to put forward a furthest point insertion strategy for meshing surfaces [25]. However, his algorithm lacked rigorous mathematical analysis and theoretical guarantees.

Delaunay refinement techniques for curved domains advanced considerably after Amenta and Bern [2] introduced local feature size based sampling theory for smooth surfaces in the context of surface reconstruction. Cheng, Dey, Edelsbrunner, and Sullivan [19] combined this sampling theory with the furthest point strategy of Chew to design a meshing algorithm for a specific type of smooth surfaces called *skin surfaces*. They used the fact that the Delaunay triangulation of a dense sample of a smooth surface contains a subcomplex called the *restricted*

Delaunay triangulation whose underlying space is homeomorphic to the sampled surface. Boissonnat and Oudot [11] showed how the furthest point insertion strategy can be applied to any C^2-smooth surface. This algorithm assumes that local feature sizes can be computed at any point required by the algorithm. Computing local feature sizes is hard in general. They suggested how to bypass the exact computation in practice. Oudot, Rineau, and Yvinec [47] extended this algorithm to volumes enclosed by a smooth surface. Cheng, Dey, Ramos, and Ray [23] came up with a different algorithm for surfaces that does not require any local feature size computations. Instead, it checks for violations of a topological property ensuring homeomorphism between input and output. We describe the algorithms of Boissonnat and Oudot [11] and the algorithm of Cheng et al. [23] in section 3.

Piecewise smooth surfaces were the next difficult class of domains on which Delaunay refinement was applied. Dey, Li, and Ray applied the strategy of [23] on piecewise linear surfaces which are assumed to approximate a smooth surface closely. Boissonnat and Oudot showed that their algorithm for smooth surface can work for a class of piecewise smooth surface called *Lipschitz surfaces* [11]. Unfortunately, the angles subtended by tangents of the surface patches meeting at the non-smooth regions are required to be close to π. The menace of small angles in Delaunay refinement for polyhedral case also appears in the piecewise smooth case. Cheng, Dey, and Ramos [18] presented an algorithm that can take piecewise smooth surfaces with arbitrary input angles. In fact, the algorithm can handle a very large class of domains called *piecewise smooth complex*, or PSC in short. This class includes polyhedral complexes, smooth surfaces, piecewise smooth surfaces, volumes enclosed by them, and non-manifolds. Unfortunately, because of some normal variation computations the algorithm is very hard to implement. Cheng, Dey, and Levine [16] modified the algorithm to make it amenable for implementation. We describe this algorithm in section 4.

1.1 Definitions

For convenience we borrow some definitions and notations from topology. All topological spaces in this paper are topological subspaces of some Euclidean space $\mathbb{R}^d$. We need maps between topological spaces that identify them as topologically equivalent. A continuous map $h\colon \mathbb{T}_1 \to \mathbb{T}_2$ is a *homeomorphism* between topological spaces $\mathbb{T}_1$ and $\mathbb{T}_2$ if h is bijective and has a continuous inverse. The second condition can be dropped if $\mathbb{T}_1$ and $\mathbb{T}_2$ are compact since any continuous bijective map between compact spaces has continuous inverse. Two spaces $\mathbb{T}_1$ and $\mathbb{T}_2$ are homeomorphic if there is a homeomorphism between them. A stronger condition than homeomorphism is captured by isotopy. Let $\mathbb{T}_1$ and $\mathbb{T}_2$ be embedded in $\mathbb{R}^d$. A continuous map $F\colon \mathbb{T}_1 \times [0,1] \to \mathbb{R}^d$ is an *isotopy* between $\mathbb{T}_1$ and $\mathbb{T}_2$ if $F(\cdot,0)$ is the identity, $F(\mathbb{T}_1, 1) = \mathbb{T}_2$, and $F(\cdot,t)$ is a homeomorphism onto its image for all $t \in [0,1]$. We say two spaces are *isotopic* if they have an isotopy between them. In words, isotopy means that one space can be continuously deformed into the other while maintaining homeomorphism all the time. Isotopic spaces are homeomorphic but the converse is not necessarily true. For example, a standard torus and a knotted torus in $\mathbb{R}^3$ are not isotopic though they are homeomorphic.

A special class of topological spaces is *manifolds*. A *k-manifold* is a topological space where each point has a neighborhood homeomorphic to $\mathbb{R}^k$ or halfspace $\mathbb{H}^k$. The points with a halfspace as neighborhood constitute the boundary of the manifold. The boundary of a topological space $\mathbb{T}$ is denoted $\mathrm{bd}\,\mathbb{T}$.

Let $d(X,Y)$ denote the Euclidean distance between two compact sets $X, Y \subseteq \mathbb{R}^d$. For $c \in \mathbb{R}^d$ and $r \in \mathbb{R}$, a *geometric* k-ball $B(c,r)$ in $\mathbb{R}^d$ is the set $\{x\}$ on a k-dimensional linear subspace of $\mathbb{R}^d$ where $d(x,c) \leqslant r$. A *topological* k-ball is a topological subspace of $\mathbb{R}^d$ which is homeomorphic to a geometric k-ball. A *k-simplex* in $\mathbb{R}^d$ is the convex hull of a set of $k+1$ linearly independent points. A *simplicial complex* is a collection of simplices where (i) a simplex is in the collection only if sub-simplices in its boundary are in the collection and (ii) any two simplices either do not intersect or intersect in a lower dimensional simplex which is also in the collection. A *cell complex* has the same property except that each element in the collection is a topological ball. Similarly, one may define a *manifold complex* with elements as manifolds. We say two elements in a complex are *incident* if they are the same or one is in the boundary of the other. They are *adjacent* if they have a non-empty intersection. For a complex $\mathcal{C}$ we denote its underlying space, that is, point-wise union of its elements as $|\mathcal{C}|$. The triangulation of a complex $\mathcal{C}$ is a simplicial complex $\mathcal{K}$ where $|\mathcal{K}|$ is homeomorphic to $|\mathcal{C}|$.

For any k-simplex t, let $c(t)$ and $r(t)$ denote the center and radius of the k-ball whose boundary contains the vertices of t. We often refer to $c(t)$ and $r(t)$ as the circumcenter and circumradius of t respectively. The radius-edge ratio $\rho(t)$ is $\frac{r(t)}{\ell(t)}$ where $\ell(t)$ is the length of the smallest edge in t. For triangles, an upper bound on ρ puts a lower bound on its minimum angle since $\rho(t) = \frac{1}{2\sin\theta}$ where θ is the minimum angle in t. We say a triangle or tetrahedron t is *ρ-skinny* if $\rho(t) \leqslant \rho$.

We specialize some of the definitions to Euclidean three space since it is the ambient space for domains considered in this article. Let $B(c,r)$ denote a geometric 3-ball in $\mathbb{R}^3$. A *circumball* of a simplex $t \subset \mathbb{R}^3$ is a 3-ball with all vertices of t on its boundary. Notice that a segment or a triangle has infintely many circumballs. A tetrahedron, on the other hand, has a single circumball. The smallest circumball of a simplex t is called its *diametric ball*. The diametric ball of a segment (edge) e is $B(c(e), r(e))$ where $c(e)$ is the midpoint of e. The diametric ball of a triangle t is $B(c(t), r(t))$ where $c(t)$ and $r(t)$ are the circumcenter and circumradius of t respectively. The diametric ball of a tetrahedron is its circumball.

We will use two very special data structures defined on a set of points $\mathcal{V} \subset \mathbb{R}^3$. Define a Voronoi cell

$$V_p = \{x \in \mathbb{R}^3 \mid d(x,p) \leqslant d(x,q)\ \forall q \in \mathcal{V}\}.$$

A Voronoi k-face is the intersection of $(4-k)$ Voronoi cells. We refer to a Voronoi 0-,1-,2-, and 3-face as Voronoi vertex, Voronoi edge, Voronoi facet, and Voronoi cell respectively. The Voronoi diagram $\operatorname{Vor}\mathcal{V}$ is the collection of all Voronoi faces. It is a cell complex. The dual complex of $\operatorname{Vor}\mathcal{V}$ is the Delaunay triangulation of $\mathcal{V}$ denoted $\operatorname{Del}\mathcal{V}$. The dual of a Voronoi k-face is a Delaunay $(3-k)$-simplex, that is, a Voronoi vertex, Voronoi edge, Voronoi facet, and Voronoi cell are dual to a Delaunay tetrahedron, Delaunay triangle, Delaunay edge, and Delaunay vertex respectively. An important property of Delaunay simplices is that they have a circumball which is *empty*, that is, their interior does not contain any of the vertices of $\mathcal{V}$. See Edelsbrunner [32] for more details.

1.2 Generic strategy

Given a domain $\mathcal{D} \subset \mathbb{R}^3$, the generic strategy of the Delaunay refinement can be stated as follows:

1. Initialize a vertex set $\mathcal{V}$ and compute $\operatorname{Del}\mathcal{V}$.
2. If a property is not satisfied, insert a point c from the domain $|\mathcal{D}|$ into $\mathcal{V}$, update $\operatorname{Vor}\mathcal{V}$ and/or $\operatorname{Del}\mathcal{V}$, and repeat step 2.
3. Return the computed mesh.

The initialization step varies from domain to domain. In polyhedral case, the vertices of the input domain are taken as the initial vertex set. For smooth domains, a set of carefully chosen points from the domain is taken as the initial vertex set. The main part of the algorithm is run by step 2. The property to be satisfied varies according to the desired output. It includes conformity to the domain, capturing topology and/or geometry of the domain, or guaranteeing shape quality of the elements. As long as the algorithm terminates, it is guaranteed that the desired properties are satisfied. Therefore, the main burdens become (i) to choose an appropriate property to be satisfied, (ii) to choose an appropriate point to be inserted, and (iii) to prove that the proposed algorithm terminate.

A dominant approach to prove the termination of the above refinement procedure is to prove that each point p is inserted in a compact subset of $\mathbb{R}^3$ where a positive lower bound is maintained on the distance of p to every other points inserted before and after p. The termination follows from the following fact.

Fact 1.1 *Let $\mathcal{C} \subset \mathbb{R}$ be compact. Let $\mathcal{V} \subset \mathcal{C}$ be a discrete point set where $d(u, v)$ for any two distinct points u, v is at least a constant $\lambda > 0$. Then, $\mathcal{V}$ is finite.*

Usually the lower bound λ on distances is proved using *local feature sizes* of the input. For polyhedral domains, it is defined as the distance between two non-adjacent input elements. For smooth cases, it is defined as the distance of a point to the medial axis of the input. We define these feature sizes precisely in the appropriate sections. The argument for termination establishes that each inserted point maintains an inter-point distance of at least some constant times the local feature size at that point. We do not provide the proofs of the results though we encourage the reader to consult the relevant papers for details.

2 Polyhedral domains

First we consider polyhedral domains. These domains are special in that they consist of planar elements, see Figure 2. A polyhedral k-cell is a k-manifold in $\mathbb{R}^3$ whose boundary is a piecewise linear $(k-1)$-manifold. We call a polyhedral 0-,1-,2-, and 3-cell as a *vertex*, *edge*, *facet*, and *polyhedron* respectively. A polyhedral complex, PC in short, is a collection of polyhedral cells, any two of which either do not intersect or intersect in a lower dimensional polyhedral cell in the complex. An input angle in a PC is any angle formed by two adjacent elements. Let σ be the lowest dimensional element containing a point x in a PC. The *local feature size* $f(x)$ is defined as the minimum distance of x to the elements non-adjacent to σ.

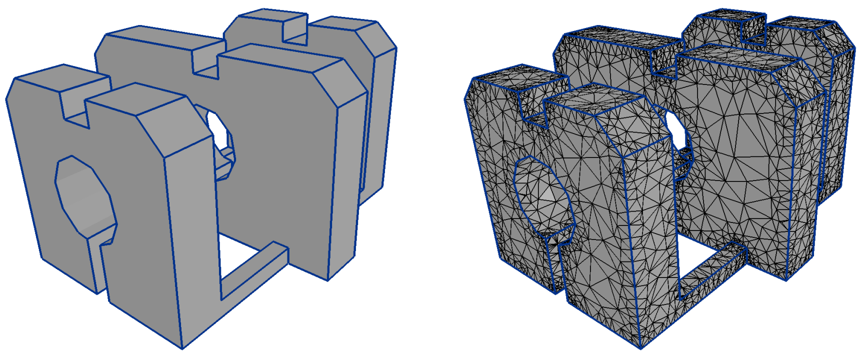


Figure 2: Example polyhedral surface and its Delaunay mesh. Input edges and facets are conformed by Delaunay simplices.

2.1 No acute angle

In this section we describe an algorithm by Shewchuk [51] for polyhedral complexes without any acute input angle. This algorithm is an extension of the algorithm of Ruppert [50] who applied Delaunay refinement for meshing planar straight line graphs in two dimensions. Let $\mathcal{P}$ denote the input PC. Just as in two dimensions, this algorithm requires that no input angle be acute.

The refinement algorithm proceeds hierarchically in the dimensions of the input elements. First, it inserts points to conform to the input edges, then to the input facets, and finally to improve the quality of the triangles and tetrahedra. The algorithm maintains a vertex set $\mathcal{V}$ and its Delaunay triangulation $\operatorname{Del}\mathcal{V}$ which is updated with each insertion. In the subroutines described below, $\mathcal{V}$ and $\operatorname{Del}\mathcal{V}$ are assumed to be globally accessible.

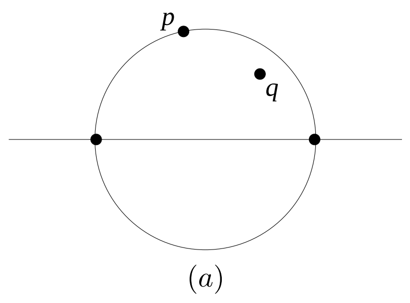
2.1.1 Conformity

The input edges are subdivided into *subsegments* by inserting points into them. We want each of the subsegments to appear in the Delaunay triangulation of the current vertex set $\mathcal{V}$. Actually, the algorithm imposes a stronger condition; the diametric ball of each subsegment should be empty of vertices in $\mathcal{V}$. Obviously, such a segment must appear as a Delaunay edge. If the diametric ball of any subsegment e contains a point, say p, we say e is *encroached* by p, or p *encroaches* e. The refinement algorithm goes on inserting the midpoint of encroached subsegments using SplitEdge.

```
SplitEdge($e$)
    Insert the midpoint of $e$ in $\mathcal{V}$ and update Del $\mathcal{V}$.
```

A set of points on an edge provides a unique subdivision of the edge into subsegments. However, the same is not true for facets. For facets one has to decide on a triangulation of the original facet by points inserted in them. Let F be any such input facet and V be the set of vertices in F. We consider the two dimensional Delaunay triangulation of V in the plane of F. Since the algorithm considers the facets only after each input edge is recovered, the boundary of F appears as a union of subsegments in this triangulation. Consequently, F is subdivided into Delaunay triangles each of which is called a *subfacet* of F. The subfacets of F may not appear in the three dimensional Delaunay triangulation of the current vertex set. Again, the algorithm enforces that the diametric ball of each subfacet be empty. Otherwise, a point is inserted. Obviously, if the process terminates, diametric balls of all subfacets become empty and hence they appear in the three dimensional Delaunay triangulation. If the *interior* of a diametric ball of any subfacet $h \subset F$ contains a point $p \in \mathcal{V}$, we say h is *encroached* by p. Notice that the definition of encroachment for subfacets is slightly different from that for subsegments, see Figure 3. The point to be inserted when a subfacet h is encroached is determined as follows. This point is the circumcenter c of h if c does not encroach any subsegment. If c encroaches a subsegment e, it may be too close to e, causing unnecessary splitting of e thereafter. To prevent this, instead of c, the midpoint of any subsegment encroached by c is inserted.

```
SplitFacet($h$)
    (i) Compute the circumcenter $c$ of $h$.
    (ii) If $c$ encroaches a subsegment $e$, call SplitEdge($e$). Otherwise,
         insert $c$ into $\mathcal{V}$ and update Del $\mathcal{V}$.
```



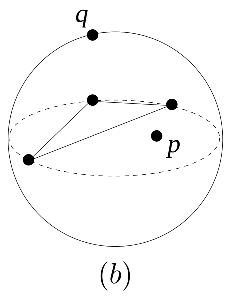


Figure 3: In Figure (a), p and q encroach the subsegment. In Figure (b), p encroaches the subfacet but q does not.

2.1.2 Quality enforcement

After conforming to the input edges and facets, the refinement algorithm focuses on improving the quality of the simplices. For an input parameter $\rho_1 > \sqrt{2}$, if there is a ρ_1-skinny triangle, it splits it using SplitFacet. Similarly, for an input parameter $\rho_2 > 2$, if there is a ρ_2-skinny tetrahedron in a polyhedron of $\mathcal{P}$, the tetrahedron is split as follows. The algorithm inserts the circumcenter c of this tetrahedron if this circumcenter does not encroach any subfacet. Otherwise, it calls SplitFacet on a subfacet encroached by c.

```
SplitTet($t$)
    1. Compute the circumcenter $c$ of $t$.
    2. If $c$ encroaches a subsegment $e$, call SplitEdge($e$) and return.
    3. If $c$ encroaches a subfacet $h$, call SplitFacet($h$) and return.
    4. Insert $c$ in $\mathcal{V}$ and update Del $\mathcal{V}$.
```

2.1.3 Algorithm

We enumerate the steps of the entire algorithm below.

```
DelPC($\mathcal{P}$,$\rho_1$,$\rho_2$)
    1. Initialize $\mathcal{V}$ to contain the vertices in $\mathcal{P}$ and compute Del $\mathcal{V}$.
    2. Find a subsegment $e$ encroached by a vertex in $\mathcal{V}$. If found, call SplitEdge($e$)
       and repeat step 2.
    3. Find a subfacet $h$ encroached by a vertex in $\mathcal{V}$. If found, call SplitFacet($h$)
       and go to step 2.
    4. Find a subfacet $h$ that has $\rho(h) > \rho_1$. If found, call SplitFacet($h$) and go to
       step 2.
    5. Find a tetrahedron $t$ inside a polyhedron in $\mathcal{P}$ so that $\rho(t) > \rho_2$. If found, call
       SplitTet($t$) and go to step 2.
    6. Return simplices in Del $\mathcal{V}$ whose union is $\mathcal{P}$.
```

One may use a bounding box enclosing $\mathcal{P}$ and then produce a Delaunay mesh of the bounding box which conforms to the input $\mathcal{P}$. Then, one does not need step 4 since it improves the quality of all tetrahedra inside the bounding box and hence also improves the quality of the triangles. Here we do not use such a bounding box and thus improve the triangle qualities explicitly. Nevertheless, the elegant analysis of Shewchuk extends easily to show that the above algorithm maintains a lower bound on inter-point distances. Specifically, it inserts a point p with a distance of at least $\lambda f(p)$ distance from all other existing points where $\lambda > 0$ is a constant [51]. This implies that the algorithm can insert only finitely many points if it inserts them in a bounded domain. We claim that all inserted points lie in the input domain $\mathcal{P}$ which is bounded. All points inserted by SPLITEDGE lie in some input edge. The following result prohibits circumcenters of triangles and tetrahedra to go outside $\mathcal{P}$.

Lemma 2.1 ([51]) *If no subsegment is encroached, the circumcenter of a Delaunay triangle in a facet F lies in F. If no subsegment and subfacet of a polyhedron $P \in \mathcal{P}$ is encroached, the circumcenter of a Delaunay tetrahedron in P lies in P.*

Thus, we have the argument that DELPC terminates. Obviously, at termination, the output mesh conforms to $\mathcal{P}$ and each triangle and tetrahedron has bounded radius-edge ratio.

Theorem 2.1 ([51]) *For a PC $\mathcal{P}$ with no acute angle,* DELPC *produces a Delaunay mesh of $\mathcal{P}$ where each simplex has a bounded radius-edge ratio.*

2.2 Allowing acute angles

It is crucial for DELPC that the input have no acute angle. Indeed, it may not terminate if input angles are allowed to be acute. Figure 4 shows an example why an acute input angle may cause incessant point insertion near it. In this section we present an algorithm DELPOLY due to Cheng et al. [22] that can cope with acute input angles when the input is a polyhedron. The vertices and edges subtending acute angles are called *sharp* vertices and edges respectively.

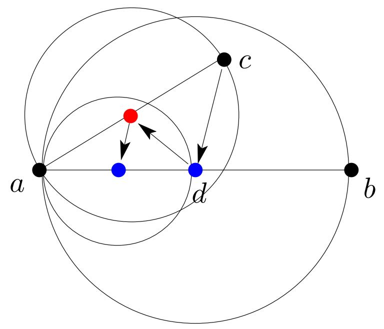


Figure 4: Indefinite splitting of encroached subsegments.

There are three main ideas that enable DELPOLY to cope with acute angles. First, in the conforming phase the vertices with an acute input angle are protected with balls called *vertex balls*. No points are allowed to be inserted inside the vertex balls. This prohibits incessant point insertions near sharp vertices. Second, we do not require the diametric balls of subfacets to be empty. Instead we relax the condition by only requiring that the subfacets appear in the three dimensional Delaunay triangulation. This prohibits inserted points ever approaching each other near sharp edges. Third, in the quality enforcing phase, we do not allow circumcenters of bad triangles or tetrahedra to be inserted near sharp vertices and sharp edges. For this purpose, we take the diametric balls of the subsegments on sharp edges after the conforming phase and double them. These balls called the *edge balls* protect sharp edges. Any triangle or tetrahedron whose radius-edge ratio violates the quality criterion is not allowed to insert its circumcenter inside any vertex or edge ball. The result of this constraint is that some of the poorly shaped elements remain in the final mesh. However, they lie near sharp vertices or edges. After all, one cannot avoid having poorly shaped elements near these regions.

Since we do not ensure that the diametric balls of the subfacets are empty, we cannot use Lemma 2.1 to guarantee that the circumcenters of tetrahedra lie inside $\mathcal{P}$. However, termination requires that the points be inserted in a bounded domain. To circumvent the problem we use a bounding box B around $\mathcal{P}$. The box B is chosen large enough so that it does not affect the local feature size of $\mathcal{P}$. The facets of the bounding box are meshed with subfacets that are not encroached. The input $\mathcal{P}$ includes the original polyhedron along with the vertices, edges, facets, and the volume of B. Once this modified $\mathcal{P}$ is meshed, one can extract the mesh triangulating original input polyhedron easily.

We describe the subroutines used by DelPoly in Sections 2.2.1–2.2.3 and then give the algorithm in Section 2.2.4.

2.2.1 Sharp vertex protection

First, we protect the sharp vertices with *vertex balls*. Points are not allowed to be inserted inside these vertex balls at later stages. This means that certain skinny tetrahedra are not removed since their removal requires insertions of vertices inside these vertex balls. Because of this constraint we compute the feature sizes at the sharp vertices explicitly and use it to compute the vertex balls. This allows one to argue that the skinny triangles and tetrahedra that we left out lie near sharp vertices and edges of the input.

For each sharp vertex u, we compute its distance from all elements of $\mathcal{P}$ which are not incident to u. This distance is the local feature size $f(u)$ at u. A brute-force computation of the feature size takes $O(n)$ time per vertex where n is the number of elements in $\mathcal{P}$. This brute-force computation is not prohibitive in practice since it is performed only once and that too before splitting any elements of the input $\mathcal{P}$.

We put a vertex ball $B_u = B(u, f(u)/4)$. The points where the boundary of B_u intersects edges of $\mathcal{P}$ are inserted into the vertex set $\mathcal{V}$. A subset of $B_u \cap \mathcal{P}$ is protected using a method similar to Cohen-Steiner, de Verdière, and Yvinec [27]. At any generic step of the algorithm, $\mathcal{V}$ contains vertices on the arc where a facet F incident to u intersects the boundary of B_u. The segments connecting consecutive points on such an arc form *shield subsegments*. Let ab be a shield subsegment. If the angle of the circular sector aub on F is at least π or ab is encroached, it is split by the following method called SOS (split on sphere) according to Cohen-Steiner et al. [27] (a more general version of this strategy is described in [8]). If the angle of the sector aub on F is at least π, we insert the midpoint x on the arc between a and b on the boundary of $B_u \cap F$. The subsegment ab is replaced with two shield subsegments ax and bx; see Figure 5(a). This type of splitting happens at most once for u, in which case a and b lie on the boundary edges of F incident to u. If the angle of the sector aub is less than π and ab is encroached, we insert the midpoint x on the shorter arc between a and b on the boundary of B_u. The subsegment ab is replaced with two shield subsegments ax and bx; see Figure 5(b). The protection procedure described above has to be modified slightly to take care of adjacent shield subsegments making acute angles between them.

When no shield subsegment corresponds to a sector at u with angle π or more, the shield subsegments around u create a set of *shield subfacets* incident to u. Figure 5(c) shows an

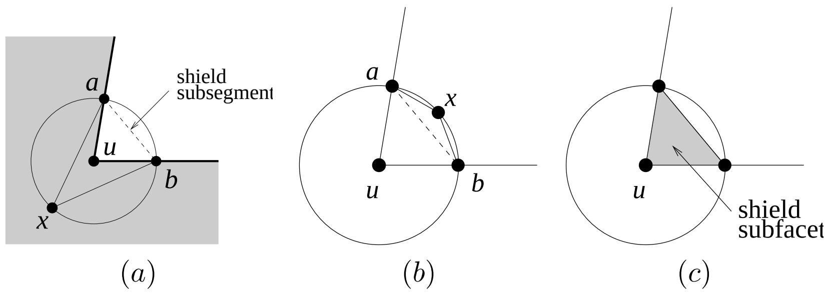


Figure 5: (a) and (b) show two shield subsegments and their splittings, (c) shows a shield subfacet.

example. The diametric ball of a shield subfacet lies in the union of the vertex ball B_u and the diametric ball of the corresponding shield subsegment. Since B_u is kept empty throughout the algorithm, it is sufficient to keep the diametric balls of shield subsegments empty to ensure that shield subfacets appear in Del $\mathcal{V}$.

In Initialize, we only insert the points where the incident edges of u intersects the boundary of B_u, and split shield subsegments that correspond to sectors with angles π or more. The encroachment of shield subsegments is handled in the conforming phase.

2.2.2 Conformity

Edges are split using a subroutine SplitEdge, which recovers the edges of $\mathcal{P}$ as union of Delaunay edges. SplitEdge is called to split any subsegment (sharp, non-sharp, or shield) that is encroached until no such segment exists.

```
SplitEdge($e$)
    If $e$ is a shield subsegment, split it with SOS else insert the midpoint
    of $e$ in $\mathcal{V}$ and update Del $\mathcal{V}$.
```

Notice that any point inserted by SplitEdge cannot lie in the vertex balls of sharp vertices. When SplitEdge terminates, each edge of $\mathcal{P}$ appears as a union of Delaunay edges.

Facets are split with subroutine SplitFacet. For the subfacets on the boundary of the bounding box, the main algorithm checks if their diametric balls are empty. For all other subfacets it checks only that the subfacets appear in Del $\mathcal{V}$. It can be shown that such a condition can be satisfied for a polyhedron after sufficient but finite amount of splitting.

```
SplitFacet($h$)
    1. Compute the circumcenter $c$ of $h$.
    2. Let $F$ be the facet containing $h$. If $c$ does not encroach any subseg-
       ment, insert $c$ into $\mathcal{V}$ and update Del $\mathcal{V}$. Otherwise, reject $c$ and
        (a) pick a subsegment $g$ encroached by $c$ with preference for those
            in bd $F$ or on $F$ (shield subsegment), and
        (b) call SplitEdge($g$).
```

In the conforming phase, after all the edges and facets are recovered, further splittings are done using the subroutine SplitBall to reduce the diametric balls of the sharp subsegments roughly to the order of local feature sizes. In order to avoid the computation of local feature sizes, this is achieved in a roundabout way. SplitBall splits any subfacet or subsegment h that is encroached by the midpoint of a sharp subsegment s provided that h and s are contained in disjoint elements of $\mathcal{P}$. The intuition is that some of the new vertices inserted to split these subfacets or subsegments will encroach s and cause s to split. We claim that, at the end of SplitBall, all diametric balls of sharp subsegments become small.

```
SplitBall($s$)
    If the midpoint of $s$ encroaches a subsegment or subfacet $h$, where
    $h$ and $s$ are contained in disjoint elements of $\mathcal{P}$, split $h$ accordingly
    using SplitEdge($h$) or SplitFacet($h$).
```

At the end of the conforming phase, for each sharp subsegment, we double the radius of its diametric ball with the center fixed and call them *edge balls*. The edge balls and the vertex balls constitute the entire set of *protecting balls* for the quality enforcing phase. Although sharp subsegments may be subdivided further in this phase, the protecting balls always refer to those computed at the end of the conforming phase.

Lemma 2.2 ([22]) *Any protecting ball $B(x,r)$ is either contained in a ball centering a sharp vertex v with radius $\frac{5}{4}f(v)$ or has $r \leqslant 2\sqrt{2}f(x)$.*

2.2.3 Quality enforcement

As in usual Delaunay refinement we attempt to insert the circumcenters of the skinny triangles and tetrahedra in this phase. For a skinny triangle if the center does not lie in any protecting balls, we call SplitFacet to split this triangle. For a skinny tetrahedron whose circumcenter does not lie in any protecting ball, we call a subroutine SplitTet which is exactly same as the one described in section 2.1.2.

Now we enumerate all steps of DelPoly which takes a piecewise linear 2- or 3-manifold $\mathcal{P}$ as input. The parameters $\rho_1 \geqslant 1$ and $\rho_2 > 2\sqrt{2}/(1-\tan(\pi/8))$ are constants chosen a priori which regulate the quality of the elements in the output mesh.

2.2.4 Algorithm

```
DelPoly($\mathcal{P}$,$\rho_1$,$\rho_2$)
    1. Initialize $\mathcal{V}$ to be the set of vertices of $\mathcal{P}$ and the bounding box B. Compute the
       vertex balls. Insert the intersections between their boundaries and the edges of
       $\mathcal{P}$ into $\mathcal{V}$. If any shield subsegment forms a sector with angle $\pi$ or more, split
       it with SOS. Compute Del $\mathcal{V}$.
    2. If there is an encroached subsegment $e$, call SplitEdge($e$) and repeat step 2.
    3. If there is a subfacet $h \subset$B that is encroached, or if $h \not\subset$B and $h$ does not
       appear in Del $\mathcal{V}$, call SplitFacet($h$) and go to step 2.
    4. If edge balls are already computed, go to step 5. For each sharp subsegment $s$
       call SplitBall($s$). Compute the edge balls by doubling the diametric balls of
       the sharp segments and go to step 2.
```

5. Find a triangle t with $\rho(t) > \rho_1$ and with its circumcenter c lying outside protecting balls. If found, call SPLITFACET(t) and go to step 2.
6. Find a tetrahedron t with $\rho(t) > \rho_2$ and with its circumcenter z lying outside protecting balls. If found, call SPLITTET(t) and go to step 2.
7. Return the subcomplex of Del $\mathcal{V}$ that cover the original input polyhedron.

DELPOLY can be modified to improve the threshold for ρ_2 to $2+\epsilon$ for any fixed $\epsilon > 0$. This worsens the output angles at the sharp vertices though. See [22] for more details.

We left out the details of the implementation of steps such as checking whether the circumcenters of ρ_1-skinny triangles and ρ_2-skinny tetrahedra lie inside a protecting ball. There is a lot of room for experimentation and variations. The main focus should be to make the computations local.

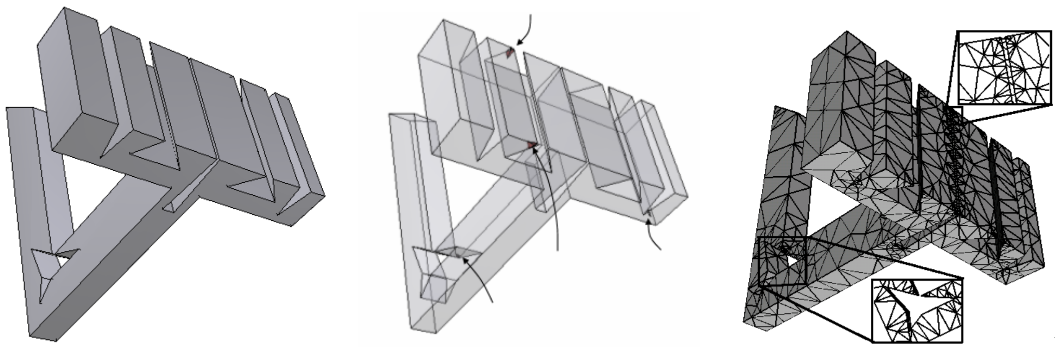


Figure 6: A polyhedral volume is meshed with QUALMESH software [58] that implements DELPOLY. Middle: some skinny tetrahedra could not be removed near sharp vertices. Right: the surface mesh.

To check whether the circumcenter c of a skinny triangle or tetrahedron lies inside a protecting ball, first find the tetrahedron t that contains c by a walk. If the protecting balls that intersect t are recorded at t, one can easily determine the protecting balls containing c. The initialization of such information can be done at the end of conformity phase by walking from sharp subsegments to all tetrahedra intersected by protecting balls. Afterwards, with each update of the Delaunay triangulation, the information can be updated with local computations only. An example of the output of DELPOLY with local computations is shown in Figure 6.

Theorem 2.2 ([22]) *Given a polyhedral surface or a polyhedraon $\mathcal{P} \subset \mathbb{R}^3$,* DELPOLY *produces a Delaunay mesh of $\mathcal{P}$ where each simplex has a bounded radius-edge ratio except the ones whose vertices lie within $5\sqrt{2}f(x)$ distance of a sharp point $x \in \mathcal{P}$.*

3 Smooth domains

In this section we consider domains that are smooth such as smooth surfaces and volumes enclosed by them. We require that the surfaces be at least C^2-smooth, compact, and have no boundary, see Figure 7. For such surfaces a sampling theory has been developed recently in the context of surface reconstruction [2, 3, 28]. This sampling theory is used to argue about the sampling and meshing of such surfaces with Delaunay refinement.

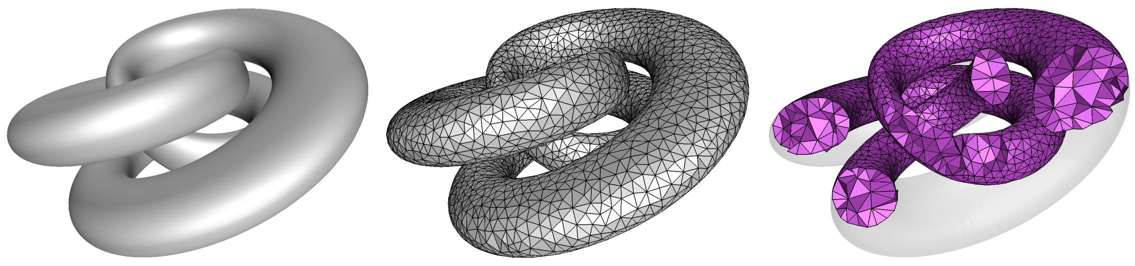


Figure 7: Delaunay mesh of a smooth knotted torus. The algorithms in section 3.1 and in section 3.2 can produce such meshes. They have been implemented in CGAL [56] and the SurfRemesh software [59] respectively.

Let $\Sigma \subset \mathbb{R}^3$ be a smooth, closed surface, that is, Σ is compact, C^2-smooth and has no boundary. As with polyhedral domains, we need a local feature size definition for smooth domains. The *medial axis* $M(\Sigma)$ of Σ is defined as the closure of the set of points $x \in \mathbb{R}^3$ so that $d(x, \Sigma)$ is realized by two or more points in Σ. The *local feature size* $f(x)$ at a point $x \in \Sigma$ is defined to be equal to $d(x, M)$. Amenta and Bern introduced this notion of local feature size in the context of surface reconstruction [2]. The local feature size captures the information of how complicated the surface is locally. Naturally, this local feature size can be used to measure how well a set of points samples Σ. A set of points $\mathcal{V} \subset \Sigma$ is called an ε-sample of Σ if $B(x, \varepsilon f(x)) \cap \mathcal{V}$ is non-empty for any $x \in \Sigma$.

It turns out that if $\mathcal{V}$ is an ε-sample of Σ for a sufficiently small value of ε, a subcomplex of the Delaunay triangulation of this sample captures the topology of Σ. We define this subcomplex in generality and then specialize it to Σ.

Let $\mathcal{V}$ be any point set in $\mathbb{R}^3$. Let V_ξ denote the dual Voronoi face of a Delaunay simplex ξ in $\operatorname{Del} \mathcal{V}$. The restricted Voronoi face of V_ξ with respect to $\mathbb{X} \subset \mathbb{R}^3$ is the intersection $V_\xi|_{\mathbb{X}} = V_\xi \cap \mathbb{X}$. The *restricted Voronoi diagram* and *restricted Delaunay triangulation* of $\mathcal{V}$ with respect to $\mathbb{X}$ are

$$\operatorname{Vor} \mathcal{V}|_{\mathbb{X}} = \{V_\xi|_{\mathbb{X}} \mid V_\xi|_{\mathbb{X}} \neq \emptyset\} \text{ and } \operatorname{Del} \mathcal{V}|_{\mathbb{X}} = \{\xi \mid V_\xi|_{\mathbb{X}} \neq \emptyset\} \text{ respectively .}$$

In words, $\operatorname{Del} \mathcal{V}|_{\mathbb{X}}$ consists of those Delaunay simplices in $\operatorname{Del} \mathcal{V}$ whose dual Voronoi face intersects $\mathbb{X}$. We call these simplices *restricted*.

Now consider a sample $\mathcal{V}$ on the surface Σ. The restricted Delaunay triangulation of $\mathcal{V}$ with respect to Σ is $\operatorname{Del} \mathcal{V}|_\Sigma$. It is known that, if $\mathcal{V}$ is an ε-sample of Σ for $\varepsilon \leqslant 0.2$, then $\operatorname{Del} \mathcal{V}|_\Sigma$ has its underlying space homeomorphic to Σ [2, 28]. To use this result one requires computing an ε-sample of Σ. A computation of local feature size or its approximation seems to be necessary to determine if a sample is an ε-sample for a pre-determined ε. Even if one is allowed to assume the availability of the local feature size at any given point, it is not immediately obvious how to place points on Σ so that they become ε-sample for a given $\varepsilon > 0$. Boissonnat and Oudot [11] showed that the furthest point strategy of Chew [25] can place the points appropriately if local feature sizes are available. We describe this algorithm in section 3.1.

When local feature sizes are not known, we cannot use the method of Boissonnat and Oudot [11]. Instead, we fall back upon a different strategy to drive the Delaunay refinement. A result of Edelsbrunner and Shah [36] says that if Voronoi faces intersect Σ in a closed topological ball of appropriate dimension, then the underlying space of the restricted Delaunay triangulation becomes homeomorphic to Σ. Therefore, a Delaunay refinement driven by the violation of the Edelsbrunner-Shah conditions provides a viable strategy for meshing with topological guarantees. This strategy is followed by Cheng, Dey, Ramos, and Ray [22]. We describe this algorithm in section 3.2.

3.1 Known feature size

Let $\mathcal{V}$ be a sample of a smooth, closed surface Σ. Consider the complex consisting of restricted triangles and their vertices and edges. It is a subcomplex of $\operatorname{Del}\mathcal{V}|_\Sigma$, namely,

$$\operatorname{Skl}^2 \mathcal{V}|_\Sigma = \{t \in \operatorname{Del}\mathcal{V}|_\Sigma \mid \ t \text{ is incident to a restricted triangle.}\}$$

Notice that the above definition uses the notion that a simplex is incident to itself. The difference between $\operatorname{Skl}^2 \mathcal{V}|_\Sigma$ and $\operatorname{Del}\mathcal{V}|_\Sigma$ is that the former one does not include restricted edges and vertices that are not incident to any restricted triangles. Since meshing of Σ should not include such hanging edges and isolated vertices, one can afford to not consider them.

The algorithm of Boissonnat and Oudot [11] is based on the following observations. For a restricted triangle $t \in \operatorname{Skl}^2 \mathcal{V}|_\Sigma$, the dual Voronoi edge V_t intersects Σ possibly at multiple points. Each ball centering such an intersection point and circumscribing vertices of t is called a *surface Delaunay ball* of t. The following lemma is proved in [11].

Lemma 3.1 *If all surface Delaunay balls $B(c, r)$ satisfy $r \leqslant \varepsilon f(c)$ for $\varepsilon \leqslant 0.09$, the underlying space of $\operatorname{Skl}^2 \mathcal{V}|_\Sigma$ is a triangulated 2-manifold without boundary.*

Let E be the triangulated 2-manifold guaranteed by Lemma 3.1. One can orient the triangles of E consistently. Let $\mathbf{n}_t$ denote the outward normal of such a triangle t. Similarly, let $\mathbf{n}_v$ denote the outward normal to Σ at a sample point v. A result of Amenta, Choi, Dey, and Leekha [3] can be called upon to establish a homeomorphism between E and Σ if one can establish that the triangles sharing vertices in E make small angles between their oriented normals. For convenience, let us say that a triangle is small if its circumradius is no more than $O(\varepsilon)$ times the local feature sizes at its vertices where $\varepsilon \leqslant 0.05$. A result of Amenta and Bern [2] shows that the acute angle between the lines normal to such triangles is $O(\varepsilon)$. However, one needs the angle between oriented normals. Therefore, a small acute angle between lines normal to triangles does not suffice. A result of [3] (also see [28]) comes to the rescue which says that oriented normals to restricted Delaunay triangles sharing an edge make less than $\frac{\pi}{2}$ angle if the triangles are small. The triangles in E are made small by the algorithm. Therefore, all triangles around a vertex v can be oriented consistently so that $\angle \mathbf{n}_t, \mathbf{n}_v = O(\varepsilon)$ for each triangle incident to v. Since adjacent vertices (joined by an edge in E) are not far apart, their oriented normals differ by $O(\varepsilon)$ angle. The implication of this observation is that $\angle \mathbf{n}_t, \mathbf{n}_v = O(\varepsilon)$ for any triangle t with a vertex v. Now we recall a result of Amenta et al. [3].

Theorem 3.1 ([3]) *Let $\nu\colon \mathbb{R}^3 \setminus M(\Sigma) \to \Sigma$ map each point of $\mathbb{R}^3$ to its nearest point in Σ. The restriction of ν to a simplicial 2-complex E whose vertices lie in Σ is a homeomorphism between E and Σ, provided that:*

1. *E is a manifold without boundary,*
2. *every triangle in E has circumradius of at most $0.113 \min\{f(v), v \text{ a vertex of } t\}$,*
3. *the normal $\mathbf{n}_t$ of every triangle t makes at most 0.375 radians with $\mathbf{n}_v$, where v is any vertex of t.*

4. E has vertices in all components of Σ.

The arguments in the paragraph following Lemma 3.1 show that conditions 1-3 of the above theorem are satisfied when $\varepsilon \leqslant 0.05$. Condition 4 has to be ensured by the algorithm. Therefore, we get an immediate algorithm as follows. Compute a point in each component of Σ. Maintain the Delaunay triangulation of the current point set $\mathcal{V}$ and check if there is any restricted triangle t with surface Delaunay ball $B(c, r)$, $r > 0.05f(c)$. If so, insert the point c into $\mathcal{V}$ and continue. There is one concern about this algorithm. How does one make sure that $\mathrm{Skl}^2\, \mathcal{V}|_\Sigma$ is non-empty at any stage of the algorithm? It may happen that no Voronoi edge of $\mathrm{Vor}\, \mathcal{V}$ intersects Σ. To avoid this, Boissonnat and Oudot describe a method to pick a triangle t for each component of Σ with a surface Delaunay ball $B(c, \frac{1}{3}f(c))$. They call such a triangle *persistent* as they persist throughout the algorithm as a restricted triangle. It is not clear how one can compute persistent triangles deterministically. The method described in [11] is a heuristic which works most of the time in practice.

We summarize the entire algorithm below. A list L of all surface Delaunay balls is maintained. Updates to $\mathrm{Vor}\, \mathcal{V}$, $\mathrm{Skl}^2\, \mathcal{V}|_\Sigma$, and L are done locally after each insertion.

```
DelSmSurf1($\Sigma$,$\varepsilon$)

    1. For each component of $\Sigma$ compute a persistent triangle. Initialize $\mathcal{V}$ to set of
       vertices of these persistent triangles. Initialize $L$ to the surface Delaunay balls
       of the persistent triangles.
    2. Find a ball $B(c, r) \in L$ so that $r > \varepsilon f(c)$. If found, insert $c$ into $\mathcal{V}$, update
       $\mathrm{Skl}^2\, \mathcal{V}|_\Sigma$ and $L$, and repeat step 2.
    3. Output $\mathrm{Skl}^2\, \mathcal{V}|_\Sigma$.
```

The parameter $\varepsilon \leqslant 0.05$ controls the refinement level of the mesh. Boissonnat and Oudot [11] show that the output mesh has a Hausdorff distance $O(\varepsilon^2)$ times the local feature sizes. Specifically, they prove the following theorem.

Theorem 3.2 ([11]) *Given a smooth, closed surface $\Sigma \subset \mathbb{R}^3$,* DelSmSurf1 *produces a mesh T with the following properties:*

1. There is an isotopy between Σ and $|T|$.

2. The isotopy moves a point $x \in \Sigma$ only by $O(\varepsilon^2)f(x)$ distance.

3. The number of vertices in T is within a constant factor of any ε-sample of Σ.

It is clear that the quality of the triangles can be easily controlled by adding a step that inserts points whenever a triangle has a radius-edge ratio greater than a threshold. See [11] for details.

3.2 Feature size unknown

The algorithm of Boissonnat and Oudot cannot be used if an oracle is not provided that can compute (or estimate in practice) the local feature size at a given point of the surface. Cheng, Dey, Ramos, and Ray [23] designed an algorithm that does not require any knowledge of local feature size. This algorithm is motivated by a result of Edelsbrunner and Shah [36] which says that if the restricted Voronoi diagram satisfies certain conditions, the restricted Delaunay triangulation becomes homeomorphic to the surface.

As in the previous section, let $\mathcal{V}$ be a point sample from a smooth closed surface $\Sigma \subset \mathbb{R}^3$. We say a k-face V_ξ of the Voronoi diagram $\mathrm{Vor}\,\mathcal{V}$ satisfies the *topological ball property*, TBP in short, if either V_ξ does not intersect Σ or intersects it transversally in a topological closed $(k-1)$-ball. The entire Voronoi diagram satisfies the TBP if each Voronoi face satisfies it. Edelsbrunner and Shah prove the following theorem [36].

Theorem 3.3 ([36]) *If* $\mathrm{Vor}\,\mathcal{V}$ *satisfies the TBP, the underlying space of the restricted Delaunay triangulation* $\mathrm{Del}\,\mathcal{V}|_\Sigma$ *is homeomorphic to* Σ.

Cheng et al. [23] drive the Delaunay refinement with violations of the TBP. For each Voronoi face violating any of the conditions of the TBP, the algorithm samples a point from the surface. The authors in [23] prove that these points maintain a positive lower bound on their distances, thereby guaranteeing the termination. Obviously, at the end, the TBP holds which by Theorem 3.3 ensures a homeomorphism between input and output. We go over the subroutines that process the TBP violations.

3.2.1 Subroutines

We have four subroutines VorEdge, TopoDisk, VorFacet, and Silhouette processing various types of TBP violations. In the case of a violation, they sample a new point from Σ. We use $\mathcal{V}$ to denote the set of sample points maintained by the algorithm. For simplicity we assume that no Voronoi vertex lies on Σ. If this assumption does not hold, the algorithm needs some modifications as detailed in [23].

The first subroutine VorEdge checks the TBP for a Voronoi edge. According to the TBP a Voronoi edge should intersect Σ transversally in a single point, a 0-ball, if it intersects at all. Recall that the Delaunay dual of e is a restricted triangle t_e. Let $cmax(t)$ and $rmax(t)$ denote the center and the radius of the largest surface Delaunay ball of a restricted triangle t. If e intersects Σ at multiple points, we insert $cmax(t_e)$.

```
VorEdge(e)
    1. If $e$ intersects $\Sigma$ tangentially, return the point of tangential contact.
    2. If $e$ intersects $\Sigma$ in two or more points, return $cmax(t_e)$.
```

In the next subroutine TopoDisk, we check whether the restricted triangles incident to a sample point p form a topological disk. Let T_p be the set of triangles in $\mathrm{Del}\,\mathcal{V}|_\Sigma$ incident to p. First, we check if every triangle edge in T_p is incident to exactly two triangles in T_p. Second, we check if T_p forms exactly one cycle of triangles around p. If both tests are passed, T_p forms a topological disk; otherwise, it does not.

TopoDisk prevents two types of TBP violations. First, if a Voronoi facet intersects Σ in two or more topological intervals, its dual Delaunay edge has more than two restricted triangles incident to it given that no Voronoi edge intersects Σ in more than one point. This condition is disallowed by TopoDisk. Second, if the boundary of a Voronoi cell V_p intersects Σ in two or more cycles, there are two or more topological disks in T_p which is again prevented by TopoDisk.

```
TopoDisk(p)
  1. If $T_p$ forms a topological disk, return null.
  2. Find the restricted triangle $t \in T_p$ for which $rmax(t)$ is the maximum among
     all triangles in $T_p$. Return $cmax(t)$.
```

A Voronoi facet should intersect Σ transversally in a single topological interval if it intersects at all. It can violate this property by (i) intersecting Σ tangentially, (ii) intersecting Σ in two or more topological intervals, and/or (iii) intersecting Σ in a closed curve. The possibility of (ii) is prevented by TopoDisk. The next subroutine VorFacet guards against the possibilities of (i) and (iii). This subroutine uses some critical point computations. Let C be a smooth closed curve on a plane. Given a direction d, the critical points of C in direction d are the points where the tangent to C is orthogonal to d.

```
VorFacet(F)
  1. If $F$ intersects $\Sigma$ tangentially, return the tangential contact point.
  2. Let $\Pi$ be the plane of $F$. Choose a direction $d$ parallel to $\Pi$. Compute $X$, the
     set of critical points of the curves in $\Pi \cap \Sigma$ in the direction $d$.
  3. If no point in $X$ lies on $F$, return null.
  4. Since $F$ intersects $\Sigma$ transversally, $F \cap \Sigma$ is a collection of disjoint simple curves
     (open or closed) and $X \cap F$ is the set of critical points of these curves in direction
     $d$. Let $V_p$ be a Voronoi cell incident to $F$. For each $x \in X \cap F$,
     (a) Compute the line $\ell_x$ in $\Pi$ through $x$ parallel to $d$. Notice that $\ell_x$ is normal
         to $F \cap \Sigma$ at $x$.
     (b) Compute $X' := \ell_x \cap \Sigma$. If $|X' \cap F| \geqslant 2$, return the point in $X' \cap F$ furthest
         from $p$.
  5. Return null.
```

The previous three subroutines make sure that Voronoi edges and Voronoi facets satisfy the TBP. Now we consider the case for Voronoi cell which, according to the TBP, should intersect Σ in a 2-ball, a topological disk. We argue that, at the end of VorEdge, TopoDisk, and VorFacet a Voronoi cell V_p cannot intersect Σ in multiple components. First, no connected component of Σ can be completely inside V_p since we initialize $\mathcal{V}$ with a point from each component of Σ. Therefore, each connected component of $V_p \cap \Sigma$ has a non-empty boundary. Second, if $V_p \cap \Sigma$ had multiple components, $(\mathrm{bd}\, V_p) \cap \Sigma$ would have multiple components. However, this is prevented by TopoDisk and VorFacet. Therefore, $V_p \cap \Sigma$ is a connected 2-manifold with a connected boundary. This surface is orientable as Σ is. Such a surface cannot be a disk only if it contains a handle. We detect the presence of handles by *silhouettes*.

For a smooth surface σ and a given direction d, the silhouette J_d is a set of closed curves formed by points $x \in \sigma$ where $\mathbf{n}_x \cdot d = 0$. This means that the normal to σ at each point of the silhouette is orthogonal to the direction d. The following result motivates the use of silhouettes.

Lemma 3.2 ([23]) *Let σ be a compact smooth surface with non-empty boundary in $\mathbb{R}^3$. If J_d is empty for any direction d, σ is a topological disk.*

The subroutine Silhouette utilizes Lemma 3.2. It checks if a Voronoi cell V_p intersects the silhouette J_d, where $d = \mathbf{n}_p$. If $V_p \cap J_d \neq \emptyset$, either J_d intersects some facets of V_p

or V_p completely contains a component of J_d. The first possibility is checked by the subroutine SilhouettePlane. The second possibility can be detected by checking if V_p contains any critical point of J_d in a direction orthogonal to d. This is done by the subroutine SilhouetteCritical. The details of SilhouettePlane and SilhouetteCritical can be found in [23].

```
Silhouette(p)

  1. Choose a direction $d'$ orthogonal to $\mathbf{n}_p$.
  2. Compute $X := $ SilhouetteCritical$(\Sigma, \mathbf{n}_p, d')$.
  3. If $X$ contains a point inside $V_p$, return it.
  4. Otherwise, for each facet $F$ of $V_p$,
      (a) Compute $X' := $ SilhouettePlane$(\Sigma, \Pi, \mathbf{n}_p)$, where $\Pi$ is the plane of $F$.
      (b) If $X'$ contains a point in $F$, return it.
  5. Return null.
```

3.2.2 Algorithm

The algorithm SampleTopology samples a set of points $\mathcal{V}$ from Σ so that $\mathrm{Del}\,\mathcal{V}|_\Sigma$ is homeomorphic to Σ. It initializes $\mathcal{V}$ with with a seed set of critical points of Σ in a chosen direction. Then it calls a procedure Topology that repeatedly invokes the subroutines in case of any violation of TBP. Upon the return of Topology, $\mathrm{Del}\,\mathcal{V}|_\Sigma$ is homeomorphic to Σ. However, it is possible that some seeds are too close together, which means that the surface triangulation may be denser than necessary around the seeds. One may fix this problem by deleting the seeds incrementally. Observe that we may start the algorithm with a single initial point and then let Silhouette generate more points on each component of Σ instead of computing the seed set. However, to avoid the more expensive computation of critical points of the silhouette in a given direction, we recommend starting with the seed set.

```
SampleTopology($\Sigma$)

  1. Initialize $\mathcal{V}$ with critical points of $\Sigma$.
  2. Compute $\mathcal{V} := $ Topology$(\mathcal{V})$.
  3. While there is a seed $p \in \mathcal{V}$, delete $p$ from $\mathcal{V}$ and compute $\mathcal{V} := $ Topology$(\mathcal{V})$.
  4. Return $\mathcal{V}$.
```

```
Topology($\mathcal{V}$)

  1. Perform steps (a)–(d) in order. Terminate the current step as soon as the
     returned $x$ is non-null; skip the following steps; and go to step 2.
      (a) For every edge $e$ of $\mathrm{Vor}\,\mathcal{V}$, compute $x := $ VorEdge$(e)$.
      (b) For every $p \in \mathcal{V}$, compute $x := $ TopoDisk$(p)$.
      (c) For every facet $F$ of $\mathrm{Vor}\,\mathcal{V}$, compute $x := $ VorFacet$(F)$.
      (d) For every $p \in \mathcal{V}$, compute $x := $ Silhouette$(p)$.
  2. If $x$ is non-null, insert $x$ into $\mathcal{V}$, update $\mathrm{Vor}\,\mathcal{V}$, and go to step 1. Otherwise,
     return $\mathcal{V}$.
```

Although the above algorithm captures the topology of the input surface Σ, the output is often a crude approximation. In order to capture the geometry better, Cheng et al. suggest some geometric refinement [23]. The triangles can be refined for their shape quality. A parameter λ controls the shape quality. If there is a triangle t in $\operatorname{Del}\mathcal{V}|_{\Sigma}$ with $\rho(t) > 1 + \lambda$, the procedure QUALITY inserts the furthest intersection point between Σ and the dual Voronoi edge of t.

```
QUALITY($\mathcal{V}$,$\lambda$)
    1. If there is a restricted triangle $t$ with $\rho(t) > 1 + \lambda$, insert $cmax(t)$ into $\mathcal{V}$ and
       update Vor $\mathcal{V}$.
    2. Return $\mathcal{V}$.
```

To capture geometry better, one may also improve the the internal dihedral angles subtended by the edges of the output mesh. Since we are meshing a smooth surface, these dihedral angles should be close to π. We measure the smoothness of $\operatorname{Del}\mathcal{V}|_{\Sigma}$ using the dihedral angles at the edges. Specifically, for each edge e in $\operatorname{Del}\mathcal{V}|_{\Sigma}$, we define the *roughness* of e, denoted by $g(e)$, to be π minus the dihedral angle at e. The procedure SMOOTH samples a point from Σ if the roughness of some edge exceeds λ.

```
SMOOTH($\mathcal{V}$,$\lambda$)
    1. If there is an edge $pq$ in $\operatorname{Del}\mathcal{V}|_{\Sigma}$ such that $g(pq) > \lambda$ insert $cmax(t)$ where $t$ is
       an incident restricted triangle of $e$ and update Vor $\mathcal{V}$.
    2. Return $\mathcal{V}$.
```

QUALITY enforces that the angles of every triangle are no less than $\arcsin\left(\frac{1}{2+2\lambda}\right)$. SMOOTH enforces the dihedral angles are no less than $\pi - O(\lambda)$. Thus, we can improve the triangle shape and smoothness by decreasing λ. However, as explained in [22] the mesh size increases linearly in $\frac{1}{\lambda^2}$.

The algorithm DELSMSURF2 combines all subroutines. It maintains the sample set $\mathcal{V}$ and $\operatorname{Del}\mathcal{V}|_{\Sigma}$ throughout its execution. The final triangulation $\operatorname{Del}\mathcal{V}|_{\Sigma}$ is the output surface mesh desired.

```
DELSMSURF2($\Sigma$,$\lambda$)
    1. Compute $\mathcal{V}$ := SAMPLETOPOLOGY($\Sigma$).
    2. Compute $\mathcal{V}$ := QUALITY($\mathcal{V}, \lambda$). If QUALITY inserted some point(s) into $\mathcal{V}$,
       compute $\mathcal{V}$ := TOPOLOGY($\mathcal{V}$) and repeat step 2.
    3. Compute $\mathcal{V}$ := SMOOTH($\mathcal{V}, \lambda$). If SMOOTH inserted a point into $\mathcal{V}$, compute
       $\mathcal{V}$ := TOPOLOGY($\mathcal{V}$) and go to step 2.
    4. Output $\operatorname{Del}\mathcal{V}|_{\Sigma}$.
```

Notice that, after QUALITY or SMOOTH, we call TOPOLOGY again because the new sample point(s) may disturb the topology of $\operatorname{Del}\mathcal{V}|_{\Sigma}$. It is worthwhile to note that one does not need to search the entire Vor $\mathcal{V}$ for a possible topology violation. Instead, a local search suffices since the insertion of a new point changes Vor $\mathcal{V}$ locally.

Theorem 3.4 ([23]) *Given a smooth, closed surface Σ, DELSMSURF2 computes a restricted Delaunay mesh whose underlying space is homeomorphic to Σ where each triangle has a bounded aspect ratio and each edge has a dihedral angle close to π.*

Notice that the above theorem does not have any guarantee about Hausdorff distance being small compared to the local feature sizes. This cannot be assured unless feature sizes are computed, and the design of DelSmSurf2 avoids these computations.

3.3 Smooth volume

In this section we consider meshing of volumes enclosed by smooth surfaces. Let $\mathcal{O}$ denote the volume enclosed by a smooth surface Σ. Let $\mathcal{V}$ be the vertex set produced either by DelSmSurf1 or DelSmSurf2. Let $T = \mathrm{Skl}^2\,\mathcal{V}|_\Sigma$ which is the mesh output by both the algorithms.

The volume enclosed by T is already triangulated with Delaunay tetrahedra. We can further refine them for quality using the radius-edge ratio condition. The circumcenters of skinny tetrahedra can be added as long as they do not disturb the surface triangulation. One easy approach is to skip adding those circumcenters who encroach the surface Delaunay balls meaning that they lie inside these balls. This ensures that all surface triangles remain intact. The trade off of this easy fix is that the tetrahedra near the boundary may not have bounded radius-edge ratios. To ensure the quality for all tetrahedra, additional effort is required to maintain the surface. We describe an algorithm by Oudot, Rineau, and Yvinec [47] which uses DelSmSurf1 for surface triangulation.

The algorithm first runs DelSmSurf1 to obtain a surface triangulation with a vertex set $\mathcal{V}$ on the surface. It splits the surface triangles with the following subroutine.

```
SplitF($t$)

    Let $B(c,r)$ be the surface Delaunay ball circumscribing $t$ where $r/f(c)$ is max-
    imum among all surface Delaunay balls of $t$. Insert $c$ into $\mathcal{V}$ and update Vor $\mathcal{V}$.
```

To mesh volumes we need tetrahedron and hence can forget about the hanging lower dimensional restricted simplices that may be present. Just as in the case of surfaces, we define the following subcomplex for the volume.

$$\mathrm{Skl}^3\,\mathcal{V}|_\mathcal{O} = \{t \in \mathrm{Del}\,\mathcal{V}|_\mathcal{O} \mid t \text{ is incident to a restricted tetrahedron.}\}$$

```
DelSmVol($\mathcal{V}, \rho, \varepsilon$)

  1. Call DelSmSurf1($\Sigma$,$\varepsilon$).
  2. Find a triangle $t \in \mathrm{Skl}^2\,\mathcal{V}|_\Sigma$ where all three vertices of $t$ are not in $\Sigma$. If found,
     call SplitF($t$,) and repeat step 2.
  3. Find a triangle $t \in \mathrm{Skl}^2\,\mathcal{V}|_\Sigma$ with a surface Delaunay ball $B(c,r)$ where $r/f(c) >$
     $\varepsilon$. If found, call SplitF($t$) and repeat step 2.
  4. Find a tetrahedron $t \in \mathrm{Skl}^3\,\mathcal{V}|_\mathcal{O}$ with $\rho(t) > \rho$. If found, do the following:
         If the circumcenter $c$ of $t$ does not encroach any surface Delaunay ball,
         insert $c$ into $\mathcal{V}$, update Vor $\mathcal{V}$, and go to step 4. Otherwise, let $c$ encroach
         a surface Delaunay ball $B(c',r)$. Insert $c'$ into $\mathcal{V}$, update Vor $\mathcal{V}$, and go to
         step 2.
  5. Return $\mathrm{Skl}^3\,\mathcal{V}|_\mathcal{O}$.
```

Step 2 ensures that all restricted triangles have vertices from Σ. Step 3 refines surface triangles as in DelSmSurf1. Step 4 refines the tetrahedra. Refinement of surface triangles is given priority over the tetrahedra. Oudot et al. [47] prove that the above algorithm terminates and has the following geometric and topological guarantees.

Theorem 3.5 ([47]) *Given a volume $\mathcal{O}$ bounded by a smooth surface Σ, for $\varepsilon \leqslant 0.05$ and $\rho > 1$,* DelSmVol *produces $T = \mathrm{Skl}^3\, \mathcal{V}|_{\mathcal{O}}$ whose underlying space is homeomorphic (isotopic) to $\mathcal{O}$ and $\mathrm{bd}\, T = \mathrm{Skl}^2\, \mathcal{V}|_{\Sigma}$. Furthermore, the isotopy moves a point $x \in \Sigma$ by at most $O(\varepsilon^2) f(x)$ distance.*

4 Piecewise smooth complex

In this section we consider a fairly large class of input domains called piecewise smooth complexes (PSCs). This class includes smooth surfaces with or without boundaries, piecewise smooth surfaces including polyhedral surfaces, non-manifold surfaces, and volumes enclosed by them. Figure 8 shows some example inputs and their meshes.

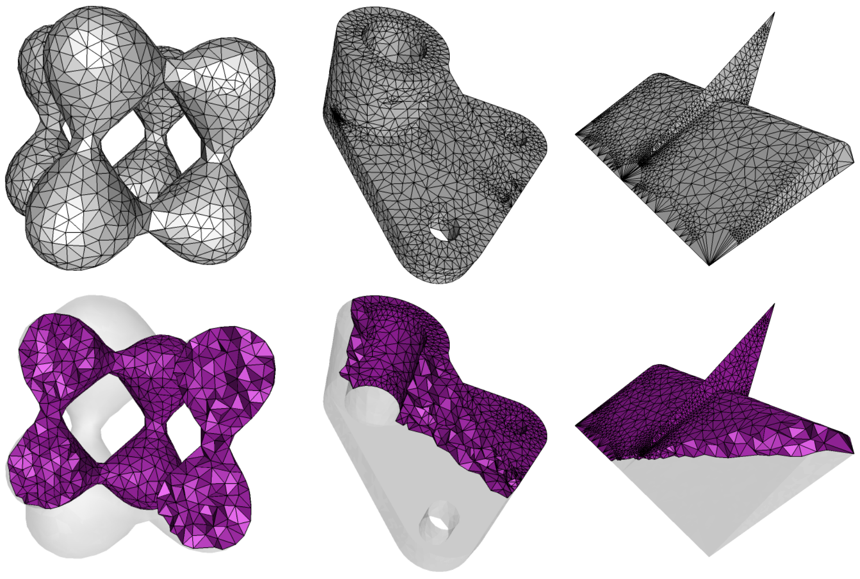


Figure 8: Meshed PSCs, Metaball (Smooth), Part (Manifold PSC), and Wedge (Non-manifold, PSC with small angles). Top row: surface mesh, bottom row: volume mesh. The meshes are all produced by DelPSC software [57].

4.1 Domain

A piecewise smooth complex (PSC) is a manifold complex where each element is a compact smooth (C^2) k-manifold, $0 \leqslant k \leqslant 3$. An element may have boundary though the element is assumed to be contained in a smooth k-manifold without boundary. We use $\mathcal{D}_k$ to denote the kth stratum, i.e., the subset of all k-dimensional elements. $\mathcal{D}_0$ is a set of *vertices*; $\mathcal{D}_1$ is a set of curves called *1-faces*; $\mathcal{D}_2$ is a set of surface patches called *2-faces*; and $\mathcal{D}_3$ is a set of volumes called *3-faces*. For $1 \leqslant k \leqslant 2$, we use $\mathcal{D}_{\leqslant k}$ to denote $\mathcal{D}_0 \cup \ldots \cup \mathcal{D}_k$. The domain is $\mathcal{D} = \mathcal{D}_0 \cup ... \cup \mathcal{D}_3$.

The domain $\mathcal{D}$, being a manifold complex, satisfies the requirements of a complex: (i) interiors of the elements are pairwise disjoint and for any $\sigma \in \mathcal{D}$, $\operatorname{bd}\sigma \subset \mathcal{D}$; (ii) for any $\sigma, \sigma' \in \mathcal{D}$, either $\sigma \cap \sigma' = \emptyset$ or $\sigma \cap \sigma'$ is a union of elements in $\mathcal{D}$.

The meshing algorithm that we are going to describe generates sample points some of which are weighted. A weighted point p with weight w_p is represented with a ball $\hat{p} = B(p, w_p)$. The squared weighted distance of any point $x \in \mathbb{R}^3$ from $\hat{p}$ is given by $\|x - p\|^2 - w_p^2$. One can define a Voronoi diagram and its dual Delaunay triangulation for a weighted point set just like their Euclidean counterparts by replacing Euclidean distances with weighted distances [4]. For a weighted point set $\mathcal{V} \subset \mathbb{R}^3$, we overload the notations $\operatorname{Vor}\mathcal{V}$ and $\operatorname{Del}\mathcal{V}$ to denote the weighted Voronoi and Delaunay diagrams of $\mathcal{V}$ respectively. For a simplex in the weighted Delaunay triangulation, it is no longer true that there exists a circumscribing ball of the simplex which is empty. Instead, a weighted version of this statement holds. For a simplex with weighted vertices $\{B(p_i, w_i)\}$, a ball $B(c, r)$ is an *orthoball* if $d(c, p_i)^2 = r^2 + w_i^2$ for all i. A simplex is in the weighted Delaunay triangulation if it has an orthoball whose weighted distance to every point in $\mathcal{V}$ is non-negative. The surface Delaunay balls of a restricted simplex are its orthoballs with centers in Σ and with non-negative weighted distance to every point in $\mathcal{V}$. The orthoball of a tetrahedron is unique whose center is called its *orthocenter*.

We have already seen in the case of smooth surface meshing that it is useful to consider a subcomplex of the restricted Delaunay triangulation consisting of restricted triangles and their subsimplices. We consider similar subcomplexes for meshing of PSCs. For example, for a 2-manifold in $\mathcal{D}$, we consider only restricted triangles and their edges and vertices. Similarly, for 1-manifolds, we consider only restricted edges and their vertices. In general, for $\sigma \in \mathcal{D}_i$, let $\operatorname{Skl}^i \mathcal{V}|_\sigma$ denote the i-dimensional complex

$$\operatorname{Skl}^i \mathcal{V}|_\sigma = \{t \in \operatorname{Del}\mathcal{V}|_\sigma \mid t \text{ is incident to a restricted } i\text{-simplex}\}.$$

Intuitively, $\operatorname{Skl}^i \mathcal{V}|_\sigma$ is an i-dimensional complex without any hanging lower dimensional simplices. For example, in Figure 9, the dark edge connecting between upper and lower part of σ is eliminated in $\operatorname{Skl}^2 \mathcal{V}|_\sigma$. We extend the definition to strata and the domain:

$$\operatorname{Skl}^i \mathcal{V}|_{\mathcal{D}_i} = \bigcup_{\sigma \in \mathcal{D}_i} \operatorname{Skl}^i \mathcal{V}|_\sigma, \ \operatorname{Skl}\mathcal{V}|_{\mathcal{D}} = \bigcup_i \operatorname{Skl}^i \mathcal{V}|_{\mathcal{D}_i}.$$

Notice that computation of $\operatorname{Skl}^i \mathcal{V}|_{\mathcal{D}_i}$ is easier than $\operatorname{Del}\mathcal{V}|_{\mathcal{D}_i}$ since the former one involves computations of intersections only between $(3-i)$-dimensional Voronoi faces with i-faces in $\mathcal{D}$. In fact, because of our special protections of $\mathcal{D}_1$, the only computation we need to determine $\operatorname{Skl}^1 \mathcal{V}|_{\mathcal{D}_1}$ and $\operatorname{Skl}^2 \mathcal{V}|_{\mathcal{D}_2}$ is Voronoi edge-surface intersections.

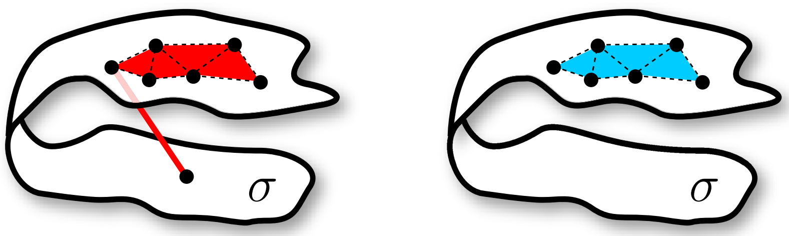


Figure 9: Left: $\operatorname{Del}\mathcal{V}|_\sigma$. Right: $\operatorname{Skl}^2 \mathcal{V}|_\sigma$.

4.2 Algorithm

The neighborhoods of the curves and vertices in $\mathcal{D}_{\leqslant 1}$ are regions of potential problems to Delaunay refinements of PSCs. First, if the elements incident to these curves and vertices make small angles at the points of incidences, usual Delaunay refinement may not terminate. Second, these curves and vertices represent "features" in the input which should be preserved in the output for many applications. Usual Delaunay refinement may destroy these features [10, 30]. As in the polyhedral case, some kind of protection mechanism is used to handle these non-smooth regions.

4.2.1 Protection

The elements in $\mathcal{D}_{\leqslant 1}$ are protected with balls before the refinement stage starts. These balls are turned into weighted points during refinement. The protecting balls satisfy the following properties:

Protection properties: Let $\omega \leqslant 0.076$ be a positive constant and B_p denote the protecting ball of a point p.

1. Any two adjacent balls on a 1-face overlap significantly without containing each other's centers.

2. No three balls have a common intersection.

3. Let $p \in \sigma$ be the center of a protecting ball. Further, let $B = B(p, R)$ be a ball where $R \leqslant c \cdot \text{radius}(B_p)$ for some $c \leqslant 8$. For a point $x \in \tau$, $n_\tau(x)$ denotes the normal to τ at x if τ is a 2-face, and it denotes the tangent to τ at x if τ is a 1-face.

   (a) For $\tau = \sigma$ or any 2-face incident to σ, $\angle n_\tau(p), n_\tau(z) \leqslant 2\omega$ for any $z \in B \cap \tau$. The same result holds for the surfaces of the 2-faces incident to σ.

   (b) B intersects σ in a single open curve and any 2-face incident to σ in a topological disk. The same result holds for the surfaces of the 2-faces incident to σ.

Each protecting ball $B_p = B(p, w_p)$ is turned into a weighted point (p, w_p). Just as in the case of smooth surfaces, it turns out that it is necessary to keep each 2-face intersected by some Voronoi edge in $\operatorname{Vor} \mathcal{V}$ throughout the algorithm. The weighted vertices ensure it for 2-faces that have boundaries. For 2-faces without boundary, Cheng et al. [16] suggest placing three weighted points satisfying the protection properties. The triangle connecting these weighted points remain restricted throughout the algorithm. The properties of the protecting balls make sure that the curves in $\mathcal{D}_1$ remain meshed properly throughout the algorithm. In particular, *adjacent* points along any curve in $\mathcal{D}_1$ remain connected with restricted Delaunay edges, see [18] for details.

4.2.2 Refinement

After protection, the algorithm inserts points iteratively outside the protected regions to mesh 2-faces. This insertion is triggered by a disk condition which essentially imposes that the triangles around a point on a 2-face form a topological disk. After 2-faces are meshed, 3-faces (volumes) are meshed with the usual circumcenter insertion procedure for refining tetrahedra. Each inserted point maintains a lower bound on its distances to all existing points. Therefore, the termination of the refinement follows from standard packing argument. At termination the

restricted complex $\mathrm{Skl}\,\mathcal{V}|_{\mathcal{D}}$ is output which has following properties:
Preserved features: All curves in $\mathcal{D}_1$ are meshed homeomorphically with restricted Delaunay edges whose vertices lie on the curves. This preserves non-smooth features or user defined features in the output, see Figure 10.
Faithful topology: All surface patches and volumes in $\mathcal{D}_{\leqslant 3}$ are meshed with a piecewise linear manifold. Furthermore, the algorithm accepts a resolution parameter λ so that it refines the Delaunay triangulations until the restricted triangles have 'size' less than λ. When λ is sufficiently small, the output restricted complex becomes homeomorphic to input $|\mathcal{D}|$.

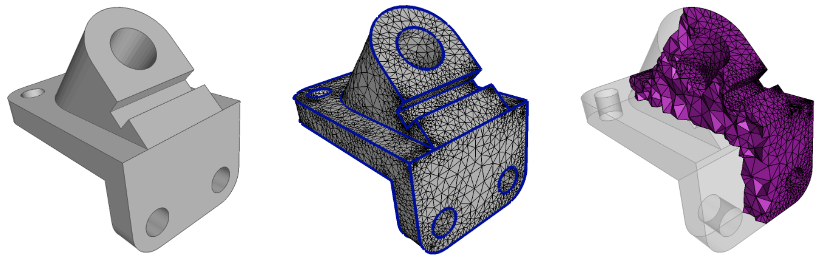


Figure 10: Features on ANCHOR are preserved in both surface (middle) and volume (right) meshing.

In a mesh of a 2-manifold, the triangles incident to a vertex should form a topological disk. Therefore, one can turn this into a condition for sampling 2-manifolds in the input PSC. The refinement condition applied to only a single 2-manifold is as simple as this. However, since a PSC may have several 2-manifolds, potentially forming even non-manifolds, one needs to incorporate some more conditions into the disk condition. Let p be a point on a 2-face σ. Let $T_p(\mathcal{D})$ and $T_p(\sigma)$ be the set of triangles in $\mathrm{Skl}^2\,\mathcal{V}|_{\mathcal{D}_2}$ and $\mathrm{Skl}^2\,\mathcal{V}|_{\sigma}$ respectively which are incident to p. The following disk condition is used for refinement. Once the restricted Delaunay triangles are collected, this check is only combinatorial.

Disk_Conditions(p) : (i) $T_p(\mathcal{D}) = \bigcup_{\sigma \ni p} T_p(\sigma)$, (ii) for each $\sigma \in \mathcal{D}_2$ containing p, underlying space of $T_p(\sigma)$ is a 2-disk which has all vertices in σ. Point p is in the interior of this 2-disk if and only if $p \in \operatorname{int}\sigma$. Also, if p is in $\operatorname{bd}\sigma$, it is not connected to any other point on $\mathcal{D}_1$ which is not adjacent to it. Figure 11 explains these conditions.

When we mesh volumes, we use the standard technique of inserting circumcenters (orthocenter) of tetrahedra that have radius-edge ratio greater than a threshold, $\rho \geqslant 1$. If the orthocenter encroaches a surface triangle, that is, the center lies in all surface Delaunay balls of a triangle in $\mathrm{Skl}^2\,\mathcal{V}|_{\mathcal{D}_2}$, the orthocenter is not inserted. Essentially, this strategy allows refining most of the tetrahedra except the ones near boundary. The following pseudocode summarizes the algorithm.

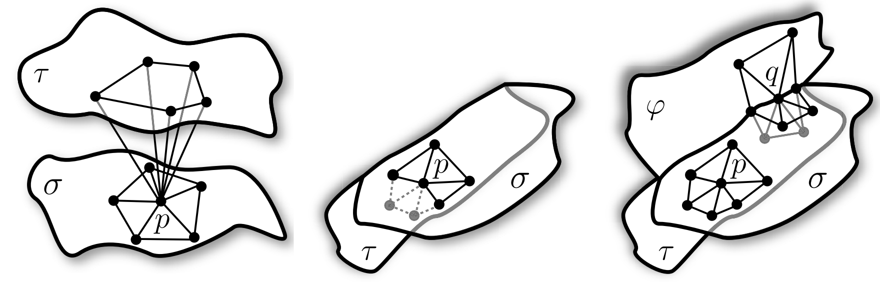


Figure 11: Left: point $p \in \sigma$ has a disk in σ and another disk in $\tau \neq \sigma$ violating condition (i). Middle: point $p \in \sigma$ has a topological disk but some of its vertices (lightly shaded) belong to τ violating condition (ii). Right: Points p and q satisfy disk condition. Point p, an interior point in σ, lies in the interior of its disk in σ. Point q, a boundary point, has three disks for each of the three 2-faces.

DelPSC $(\mathcal{D}, \lambda, \rho)$

1. Protection. Protect elements in $\mathcal{D}_{\leqslant 1}$ with weighted points. Insert three weighted points in each element of $\mathcal{D}_2$ that has no boundary. Let $\mathcal{V}$ be the current (weighted) point set.
2. Mesh2Complex.
   (a) Let (p, σ) be any tuple where $p \in \mathrm{Skl}^2\, \mathcal{V}|_\sigma$. If **Disk_Conditions**$(p)$ is violated, find the triangle $t \in T_p(\mathcal{D})$ which has the largest surface Delaunay ball $B(c, r)$. Insert c into $\mathcal{V}$, update $\mathrm{Vor}\, \mathcal{V}$, and repeat step 2(a).
   (b) If there is restricted triangle with a surface Delaunay ball $B(c, r)$ where $r > \lambda$, insert c into $\mathcal{V}$, update $\mathrm{Vor}\, \mathcal{V}$, and go to step 2(a). Otherwise, go to step 3.
3. Mesh3Complex. Find a tetrahedron $t \in \mathrm{Skl}^3\, \mathcal{V}|_\sigma$ where $\rho(t) > \rho$ and the orthocenter c of t does not encroach any triangle in $\mathrm{Skl}^2\, \mathcal{V}|_\mathcal{D}$ and any ball $B(p, 2r)$ where $B(p, r)$ is a protecting ball. If found, insert c into $\mathcal{V}$, update $\mathrm{Vor}\, \mathcal{V}$, and repeat step 3.
4. Return $\mathrm{Skl}\, \mathcal{V}|_\mathcal{D}$.

4.2.3 Guarantees

Cheng, Dey, and Levine [17] show the following.

Theorem 4.1 ([17]) *(i)* DelPSC *terminates, (ii) the output of* DelPSC *satisfies properties* T1-T3*:*

(T1) *For each $\sigma \in \mathcal{D}_1$, $\mathrm{Skl}^1\, \mathcal{V}|_\sigma$ is homeomorphic to σ and two vertices are joined by an edge in $\mathrm{Skl}^1\, \mathcal{V}|_\sigma$ if and only if these two vertices are adjacent on σ.*

(T2) *For $0 \leqslant i \leqslant 2$ and $\sigma \in \mathcal{D}_i$, $\mathrm{Skl}^i\, \mathcal{V}|_\sigma$ is a i-manifold with vertices only in σ. Further, $\mathrm{bd}\, \mathrm{Skl}^i\, \mathcal{V}|_\sigma = \mathrm{Skl}^{i-1}\, \mathcal{V}|_{\mathrm{bd}\, \sigma}$. For $i = 3$, the statement is true only if the set $\mathrm{Skl}^i\, \mathcal{V}|_\sigma$ is not empty at the end of* Mesh2Complex.

(T3) *There exists a $\lambda > 0$ so that the output mesh of* DelPSC*$(\mathcal{D},\lambda,\rho)$ has an underlying space homeomorphic to $|\mathcal{D}|$. Further, this homeomorphism respects stratification with* *vertex restrictions, that is, for $0 \leqslant i \leqslant 3$, $\mathrm{Skl}^i \mathcal{V}|_\sigma$ is homeomorphic to $\sigma \in \mathcal{D}_i$ where $\mathrm{bd}\,\mathrm{Skl}^i \mathcal{V}|_\sigma = \mathrm{Skl}^{i-1} \mathcal{V}|_{\mathrm{bd}\,\sigma}$ and vertices of $\mathrm{Skl}^i \mathcal{V}|_\sigma$ lie in σ.*

The implication of T1 is that the non-smooth features are preserved in the output, see Figure 10. The implication of T1 and T2 is that each k-manifold is meshed with a simplicial k-manifold. Also, the mesh of the boundary of a k-manifold appears as the boundary of the mesh of the k-manifold. The meshes may not be homeomorphic to the original manifold, but with increasing levels of refinement (controlled by parameter λ) homeomorphism is achieved. In practice this level is reached quite early in the refinement.

One can add the following step at the end of Mesh2Complex to improve the quality of the triangles.

> If there is a triangle $t \in \mathrm{Skl}^2 \mathcal{V}|_{\mathcal{D}_2}$ with $\rho(t) > \rho_1$, insert the center c of its surface Delaunay ball into $\mathcal{V}$ if c does not lie in $B(p, 2r)$ where $B(p, r)$ is a protecting ball. Update $\mathrm{Vor}\,\mathcal{V}$, and go back to step 2(a).

In Mesh3Complex we avoided inserting the circumcenter c of a tetrahedron if it encroaches a surface Delaunay ball of a triangle. Instead one can take an approach as in the polyhedral case by rejecting c and splitting the triangle encroached by c.

> If there is a tetrahedron $t \in \mathrm{Skl}^3 \mathcal{V}|_{\mathcal{D}_3}$ where $\rho(t) > \rho_2$ and the orthocenter c of t does not lie in $B(p, 2r)$ where $B(p, r)$ is a protecting ball do the following: insert c if it does not encroach any triangle in $t' \in \mathrm{Skl}^2 \mathcal{V}|_{\mathcal{D}_2}$, otherwise reject c and insert the center of a surface Delaunay ball of t' instead. In both cases update $\mathrm{Vor}\,\mathcal{V}$ and go back to step 2(a).

The analysis in [16] can easily be modified to claim that Theorem 4.1 still holds with the added steps for quality enforcement.

5 Open issues

In this section we briefly describe different open issues in Delaunay meshing of three dimensional domains.

The algorithms described in previous sections attempt to improve the quality of tetrahedra by improving their radius-edge ratios. As mentioned earlier, an upper bound on radius-edge ratios eliminate all types of bad tetrahedra except one, slivers. A sliver tetrahedron t has an upper bound on radius-edge ratio $\rho(t)$ but has poor volume-edge ratio given by $\sigma(t) = \frac{v(t)}{\ell^3(t)}$ where $v(t)$ and $\ell(t)$ are the volume and smallest edge length of t. Considerable research effort over the past decade has been devoted to exude slivers from Delaunay meshes. The first successful result was given by Cheng et al. in their sliver exudation paper [15]. The authors show that, for a periodic point set $\mathcal{V} \subset \mathbb{R}^3$, if $\mathrm{Del}\,\mathcal{V}$ has an upper bound on radius-edge ratios for tetrahedra, the vertices can be assigned weights so that all slivers with volume-edge ratio smaller than a positive constant σ_0 are eliminated. Although it is the first result of its kind, the algorithm cannot handle bounded domain like polyhedra. Cheng and Dey [14] adapted the sliver exudation method to polyhedral complexes. For other attempts in sliver removal, see [26, 33, 43]. The sliver exudation method of Cheng and Dey [14] can be used with the algorithms for smooth and piecewise smooth volume meshing [16, 47] to get rid of slivers. Theoretically, the constant σ_0 for sliver removal is extremely small. In practice, it has been observed that slivers with dihedral angles in the range $1^\circ - 3^\circ$ are not eliminated [20, 35]. It is still an open question how to eliminate slivers in Delaunay meshing of bounded domains where the angle bounds are large, say near 10°. Labelle [42] showed a technique to achieve such a bound for periodic point set but it is not clear how to extend it to bounded domains.

The Delaunay refinement strategy inserts points that are locally furthest. They are generally points where a dual Voronoi face of a Delaunay element to be eliminated intersects the domain. A natural question is if there are other insertion strategies that may improve the refinement algorithms. Edelsbrunner and Guoy [34] showed that insertion of "sinks" that are not necessarily circumcenters of bad triangles or tetrahedra can indeed improve the mesh size in practice. Üngör proposed insertions of "off-centers" which remarkably improve the mesh size in two dimensions [55]. It remains open how to apply these or other insertion strategies for bounded three dimensional domains.

In the past few years some optimization based techniques have been proposed for producing meshes with well shaped simplices [13, 31]. These approaches seem promising; in particular, for addressing the question of slivers in tetrahedral meshing [1]. However, it is still open to apply these techniques to bounded domains with provable guarantees. Some applications require that the mesh simplices contain their dual Voronoi vertex. For triangles, this means that they do not have any obtuse angle. Bern, Mitchell, and Ruppert [7] gave an algorithm for producing linear size non-obtuse triangulations for polygons in two dimensions (also see [37]). The question remains open for three dimensional bounded domains.

The question of space and time complexities of Delaunay refinement algorithms for three dimensional domains remains mostly open. If m is the output size of the mesh, it is straightforward to derive an $O(m^2)$ time complexity bound. Each insertion of a point cannot take more than $O(m)$ time and hence $O(m^2)$ bound is trivial. However, the challenge remains to achieve a non-trivial bound, in particular an $O(n \log n + m)$ bound where n is the space complexity of the input domain. Har-Peled and Üngör [39] presented an off-center based algorithm for two dimensional point sets that runs in optimal time and space. For a restricted class of PCs, Hudson, Miller, and Phillips [40] show that a Delaunay refinement scheme can be run in $O(n \log(L/s) + m)$ time where L/s is the ratio of the diameter to minimum feature size. The output mesh size m certainly regulates the time complexity of the Delaunay refinement and also of the post-processing algorithms that use these meshes. Ideally, m should be close to optimal. For an input domain, let m^* denote the size of a Delaunay complex that has minimum number of simplices over all possible meshes conforming to the input domain. If one adds the mesh quality condition, m^* may change. It is very difficult and perhaps impossible to find algorithms that has optimal output size and runs in polynomial time in terms of output complexity. However, it might be possible to design algorithms that produce output size within a constant factor of the optimal. In 2D, Ruppert [50] achieves this. In 3D, one has to address the issue of slivers to obtain constant-factor optimal algorithms. For bounded domains, the only algorithm that achieves it for space complexity is of Cheng and Dey [14] for PCs with no acute angles. For other three dimensional bounded domains, achieving optimal algorithms in terms of space and time complexity remains an important open question.

Acknowledgements

We acknowledge the use of CGAL libraries for developing various meshing software [57, 58, 59] that have been used to generate some of the pictures used in this article. Also, we acknowledge the support of NSF grants CCF-0430735 and CCF-0635008.

References

[1] P. Alliez, D. Cohen-Steiner, M. Desbrun, and M. Yvinec. Variational tetrahedral meshing. *ACM Siggraph 2005* (2005), 617–625.

[2] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discr. Comput. Geom.* **22** (1999), 481–504.

[3] N. Amenta, S. Choi, T. K. Dey and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. *Internat. J. Comput. Geom. Applications* **12** (2002), 125–141.

[4] F. Aurenhammer. Power diagrams : properties, algorithms, and applications. *SIAM J. Computing* **16** (1987),78–96.

[5] M. Bern, D. Eppstein and J. Gilbert. Provably good mesh generation. *J. Comput. Syst. Sci.*, 48 (1994), 384–409.

[6] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In D.-Z. Du and F. K. Hwang editors, *Computing in Euclidean Geometry, 2nd Edition*, World Scientific, Singapore, (1995), 47–123.

[7] M. Bern, S. A. Mitchell, and J. Ruppert. Linear-size nonobtuse triangulation of polygons. *Proc. 10th Annu. ACM Sympos. Comput. Geom.* (1994), 221–230.

[8] C. Boivin and C. Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *Intl. J. Numer. Methods in Engineer.*, 55 (2002), 1185–1213.

[9] J.-D. Boissonnat, D. Cohen-Steiner, B. Mourrain, G. Rote, and G. Vegter. Meshing of surfaces. Chapter 5 in *Effective Computational Geometry for Curves and Surfaces*, eds. J.-D. Boissonnat, M. Teillaud, Springer Verlag, 2006.

[10] J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of Lipschitz surfaces. *Proc. 22nd Ann. Sympos. Comput. Geom.* (2006), 337–346.

[11] J.-D. Boissonnat and S. Oudot. Provably good surface sampling and meshing of surfaces. *Graphical Models* **67** (2005), 405–451.

[12] J.-D. Boissonnat, D. Cohen-Steiner, and G. Vegter. Isotopic implicit surface meshing. *Proc. 36th Annu. ACM Sympos. Theory Comput.* (2004), 301–309.

[13] L. Chen and J. Xu. Optimal Delaunay triangulations. *J. Comput. Mathematics* **22** (2004), 299–308.

[14] S.-W. Cheng and T. K. Dey. Quality meshing with weighted Delaunay refinement. *SIAM J. Comput.*, 33 (2003), 69–93.

[15] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello and S.-H. Teng. Sliver exudation. *J. ACM*, 47 (2000), 883–904.

[16] S.-W. Cheng, T. K. Dey, and J. Levine. A practical Delaunay meshing algorithm for a large class of domains. *Proc. 16th. Internat. Meshing Roundtable*, 2007.

[17] S.-W. Cheng, T. K. Dey, and J. Levine. Theory of a practical Delaunay meshing algorithm for a large class of domains. Manuscript, 2007.

[18] S.-W. Cheng, T. K. Dey, and E. A. Ramos. Delaunay refinement for piecewise smooth complexes. *Proc. 18th Annu. ACM-SIAM Sympos. Discrete Algorithms* (2007), 1096–1105.

[19] H.-L. Cheng, T. K. Dey, H. Edelsbrunner, and J. Sullivan. Dynamic skin triangulation. *Discrete Comput. Geom.* **25** (2001), 525–568.

[20] S.-W. Cheng, T. K. Dey, and T. Ray. Weighted Delaunay refinement for polyhedra with small angle. *proc. 14th Internat. Meshing Roundtable* (2005), 325–342.

[21] S.-W. Cheng and S.-H. Poon. Three-dimensional Delaunay mesh generation. *Discrete Comput. Geom.* **36** (2006), 419–456.

[22] S.-W. Cheng, T. K. Dey, E. A. Ramos and T. Ray. Quality meshing for polyhedra with small angles. *Internat. J. Comput. Geom. Appl.* **15** (2005), 421–461.

[23] S.-W. Cheng, T. K. Dey, E. A. Ramos and T. Ray. Sampling and meshing a surface with guaranteed topology and geometry. *Proc. 20th Ann. Sympos. Comput. Geom.* (2004), 280–289.

[24] L. P. Chew. Guaranteed-quality triangular meshes. Report TR-98-983, Comput. Sci. Dept., Cornell Univ., Ithaca, New York, 1989.

[25] L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. *Proc. 9th Ann. Sympos. Comput. Geom.* (1993), 274–280.

[26] L. P. Chew. Guaranteed-quality Delaunay meshing in 3D. *Proc. 13th Ann. Sympos. Comput. Geom.* (1997), 391–393.

[27] D. Cohen-Steiner, E. C. de Verdière and M. Yvinec. Conforming Delaunay triangulations in 3D. *Proc. Ann. Sympos. Comput. Geom.*, 2002, 199–208.

[28] T. K. Dey. Curve and surface reconstruction : algorithms with mathematical analysis. Cambridge U. Press, New York, 2006.

[29] T. K. Dey, C. Bajaj and K. Sugihara. On good triangulations in three dimensions. *Internat. J. Comput. Geom.* **2** (1992), 75–95.

[30] T. K. Dey, G. Li, and T. Ray. Polygonal surface remeshing with Delaunay refinement. *Proc. 14th Internat. Meshing Roundtable* (2005), 343–361.

[31] Q. Du and D. Wang. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. *Internat. J. Numeric. Method. Engineering* **56** (2003), 1355-1373.

[32] H. Edelsbrunner. *Geometry and Topology for Mesh Generation.* Cambridge Univ. Press, England, 2001.

[33] H. Edelsbrunner, X.-Y. Li, G. L. Miller, A. Stathopoulos, D. Talmor, S.-H. Teng, A. Üngör and N. Walkington. Smoothing cleans up slivers. *Proc. 32nd Ann. ACM Sympos. Theory Comput.*, (2000), 273–277.

[34] H. Edelsbrunner and D. Guoy. Sink-insertion for mesh improvement. *Internat. J. Found. Comput. Sci.* **13** (2002), 223–242.

[35] H. Edelsbrunner and D. Guoy. An experimental study of sliver exudation. *Proc. 10th Intl. Meshing Roundtable*, (2001), 307–316.

[36] H. Edelsbrunner and N. Shah. Triangulating topological spaces. *Internat. J. Comput. Geom. Appl.* **7** (1997), 365–378.

[37] D. Eppstein. Faster circle packing with application to nonobtuse triangulations. *Internat. J. Comput. Geom. Appl.* **7** (1997), 485–491.

[38] L. A. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *Intl. J. Numer. Methods Engg.*, **40**, (1997), 3979–4002.

[39] S. Har-Peled and A. Üngör. A time optimal Delaunay refinement algorithm in two dimensions. *Proc. 22nd Ann. Sympos. Comput. Geom.* (2005), 228–236.

[40] B. Hudson, G. Miller, and T. Phillips. Sparse Voronoi refinement. *Proc. 15th Internat. Meshing Roundtable* (2006).

[41] P. Knupp. Algebraic mesh quality metrics. *SIAM J. Sci. Comput.* **23** (2001), 193-218.

[42] F. Labelle. Sliver removal by lattice refinement. *Proc. Ann. Sympos. Comput. Geom.* (2006), 347–356.

[43] X.-Y. Li and S.-H. Teng. Generating well-shaped Delaunay meshes in 3D. *Proc. 12th. Ann. ACM-SIAM Sympos. Discrete Algorithm* (2001), 28–37.

[44] G. L. Miller, D. Talmor, S.-H. Teng and N. Walkington. A Delaunay based numerical method for three dimensions: generation, formulation, and partition. *Proc. 27th Ann. ACM Sympos. Theory Comput.* (1995), 683–692.

[45] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in higher dimensions. *SIAM J. Comput.*, 29 (2000), 1334–1370.

[46] S. Owen. A survey of unstructured mesh generation technology. http://www.andrew.cmu.edu/sowen/survey.

[47] S. Oudot, L. Rineau, and M. Yvinec. Meshing volumes bounded by smooth surfaces. *Proc. 14th Internat. Meshing Roundtable* (2005), 203–219.

[48] S. Pav and N. Walkington. Robust three dimensional Delaunay refinement. *13th Internat. Meshing Roundtable* 2004.

[49] S. Plantinga and G. Vegter. Isotopic meshing of implicit surfaces. *The Visual Computer* **23** (2007), 45–58.

[50] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, **18** (1995), 548–585.

[51] J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. *Proc. 14th Ann. Sympos. Comput. Geom.* (1998), 86–95.

[52] J. R. Shewchuk. Mesh generation for domains with small angles. *Proc. 16th Ann. Sympos. Comput. Geom.* (2000), 1–10.

[53] J. R. Shewchuk. What is a good linear element? interpolation, conditioning and quality measures. *Proc. 11th Internat. Meshing Roundtable* (2002), 115–126.

[54] H. Shi and K. Gärtner. Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. *Proc. 14th Internat. Meshing Roundtable* (2005), 147–164.

[55] A. Üngör. Off-centers : A new type of Steiner points for computing size-optimal guaranteed-quality Delaunay triangulations. *Proc. Latin American Theoretical Informatics* (2004), 152–161.

[56] http://www.cgal.org

[57] http://www.cse.ohio-state.edu/∼tamaldey/delpsc

[58] http://www.cse.ohio-state.edu/∼tamaldey/qualmesh

[59] http://www.cse.ohio-state.edu/∼tamaldey/surfremesh