Weierstraß-Institut
für Angewandte Analysis und Stochastik

im Forschungsverbund Berlin e.V.

Preprint ISSN 0946 – 8633

Constrained Delaunay tetrahedral mesh generation and refinement

Hang Si[1]

submitted: 1st December 2008

[1] Weierstrass Institute for Applied Analysis and Stochastics
si@wias-berlin.de

No. 1372
Berlin 2008

2000 *Mathematics Subject Classification.* 52B55, 65D18.

Key words and phrases. Constrained Delaunay tetrahedralization, mesh generation, boundary recovery, mesh refinement.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Mohrenstraße 39
10117 Berlin
Germany

Fax: + 49 30 2044975
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Abstract

A *constrained Delaunay tetrahedralization* of a domain in $\mathbb{R}^3$ is a tetrahedralization such that it respects the boundaries of this domain, and it has properties similar to those of a Delaunay tetrahedralization. Such objects have various applications such as finite element analysis, computer graphics rendering, geometric modeling, and shape analysis.

This article is devoted to presenting recent developments on constrained Delaunay tetrahedralizations of piecewise linear domains. The focus is on the application of numerically solving partial differential equations using finite element or finite volume methods. We survey various related results and detail two core algorithms that have provable guarantees and are amenable to practical implementation. We end this article by listing a set of open questions.

1 Introduction

Meshing of geometric domains has various applications such as finite element analysis, computer graphics rendering, geometric modeling, and shape analysis. Although a vast literature exists on mesh generation, many fundamental three-dimensional meshing problems are still challenging in both theory and practice. This article deals with two closely related meshing problems, namely *boundary conformity* and *mesh refinement*, in their applications to adaptive finite element analysis. As a common theme, we illustrate how a special object, called *constrained Delaunay tetrahedralizations*, can be used to solve these problems.

1.1 Boundary Conformity

Let $\mathcal{F}$ be a surface mesh which is a discretization of the boundary of a three-dimensional domain Ω, see Fig. 1 for an example. The problem of *boundary conformity* asks to generate a tetrahedral mesh $\mathcal{T}$ conforming to $\mathcal{F}$, i.e., $\mathcal{F}$ is represented by a union of elements of $\mathcal{T}$. Additional points (so-called *Steiner points*) are allowed in $\mathcal{T}$, but the number of Steiner points should be limited as small as possible. This problem is fundamental to many applications. For example, many engineering mesh generation methods [4, 59, 58, 28, 53] make use of such tetrahedral meshes as the intermediate objects to obtain good quality meshes suitable for numerical simulations. Some applications, e.g., local re-meshing and anisotropic meshing, use a pre-discrized surface mesh as input, and require that the tetrahedral mesh must

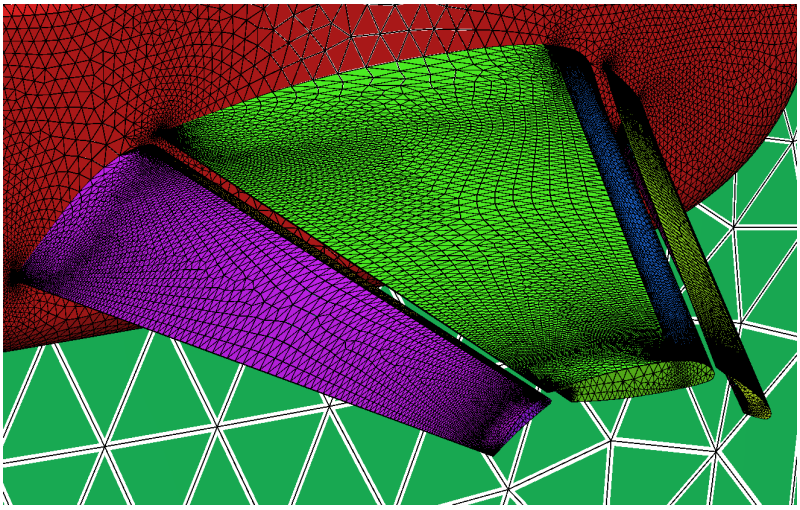


Figure 1: A surface mesh of an airfoil.

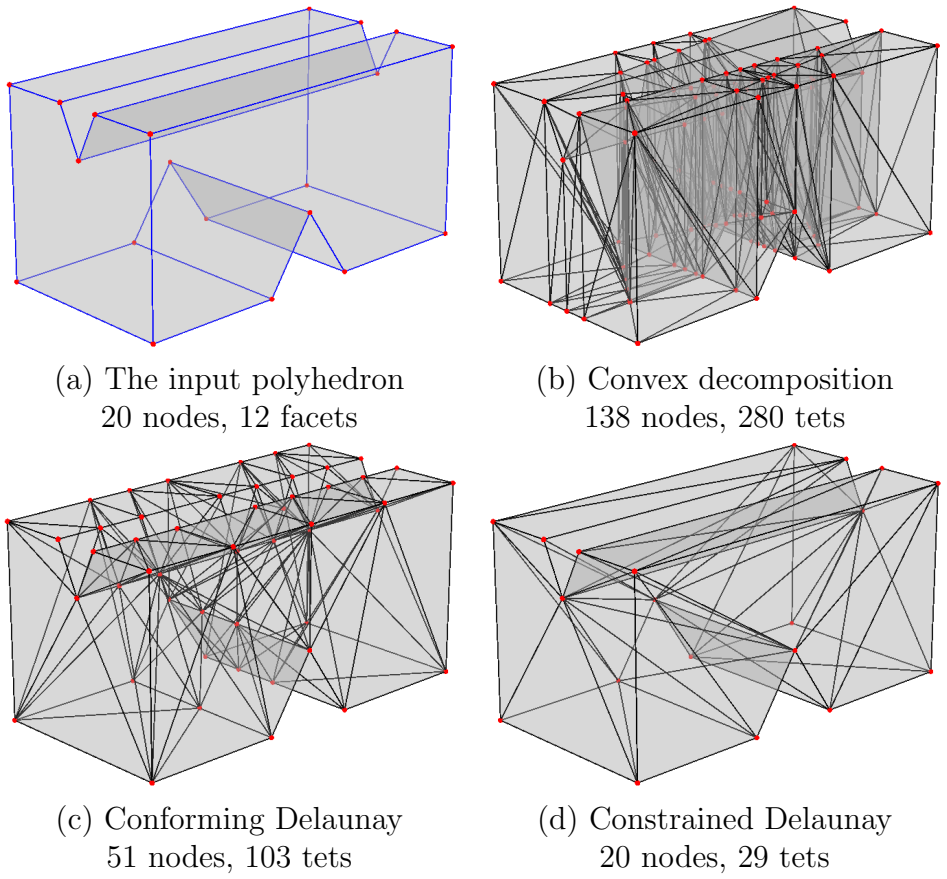
match the surface exactly. This imposes an additional requirement for this problem, i.e., no Steiner points should be added on $\mathcal{F}$.

In three dimensions, this problem faces many difficulties. There are simple polyhedra which may not be tetrahedralized without Steiner points [44]. The problem of determining whether or not a non-convex polyhedron can be tetrahedralized without Steiner points is NP-complete [43]. Chazelle [8] showed that a large number of Steiner points may be needed to tetrahedralize a simple polyhedron.

A number of engineering methods have been proposed which provided a constructive proof of the existence of a solution to the problem, see e.g., [30, 31, 58, 6, 33, 29, 20, 21]. A common feature of these methods is to insert the Steiner points directly in the places where the boundaries (edges and faces) and an initial tetrahedralization intersect. These methods showed great success in solving some realistic cases arising from engineering applications. However, they are not designed for arbitrary inputs. Particularly, the number of Steiner points may be large for some pathological cases.

Chazelle and Palios [9] proved an upper bound for the number of Steiner points by giving an algorithm to decompose a simple polyhedron P using $O(n + r^2)$ Steiner points, where r is the number of reflex edges (a quantitative measure of nonconvexity) of P. However, their algorithm will usually introduce an unnecessarily large number of Steiner points even for a simply-shaped polyhedron, see Fig. 2 (b). Hence it is only of theoretical interest. Practical approaches using conforming Delaunay triangulations [41, 15] and constrained Delaunay triangulations [48, 55] are proposed, see Fig. 2 (c) and (d). Constrained Delaunay triangulations need less Steiner points than conforming Delaunay triangulations.

Let P be a three-dimensional polyhedron. The set of vertices of P is denoted as $V(P)$. Let S be a finite set of points in $\mathbb{R}^3$ and $V(P) \subseteq S$. Two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$



(a) The input polyhedron
20 nodes, 12 facets

(b) Convex decomposition
138 nodes, 280 tets

(c) Conforming Delaunay
51 nodes, 103 tets

(d) Constrained Delaunay
20 nodes, 29 tets

Figure 2: A comparison of meshing polyhedra by different approaches

are *invisible* to each other if the line segments $[\mathbf{x}, \mathbf{y}]$ intersects a face F of P in a single point other than $\mathbf{x}$ and $\mathbf{y}$. A simplex σ whose vertices are in S is *constrained Delaunay* if either it is Delaunay in S, or it has a circumscribed sphere that does not contain any other vertex of S which is visible from the interior of σ.

A *constrained Delaunay tetrahedralization* (abbreviated as CDT) of P is defined as a partition $\mathcal{T}$ of P such that $\mathcal{T}$ is a simplicial complex and every simplex of $\mathcal{T}$ is constrained Delaunay. By this definition, a CDT of P may contain Steiner points, i.e., those points in $S \setminus V(P)$. In general, there are infinitely many CDTs of P (by different choices of Steiner points). Our goal is to find a CDT of P which contains a small number of Steiner points.

Problem 1.1 *Given a three-dimensional polyhedron P, generate a constrained Delaunay tetrahedralization $\mathcal{T}$ of P such that the number of Steiner points in $\mathcal{T}$ is as small as possible.*

A key question to the above problem is to determine under which condition Steiner points are not needed. Call an edge σ of P *strongly Delaunay* if there is a circumscribed sphere Σ of σ such that all other vertices lie strictly outside and not on Σ. Shewchuk [45] showed that if all edges of P are strongly Delaunay, then P has a

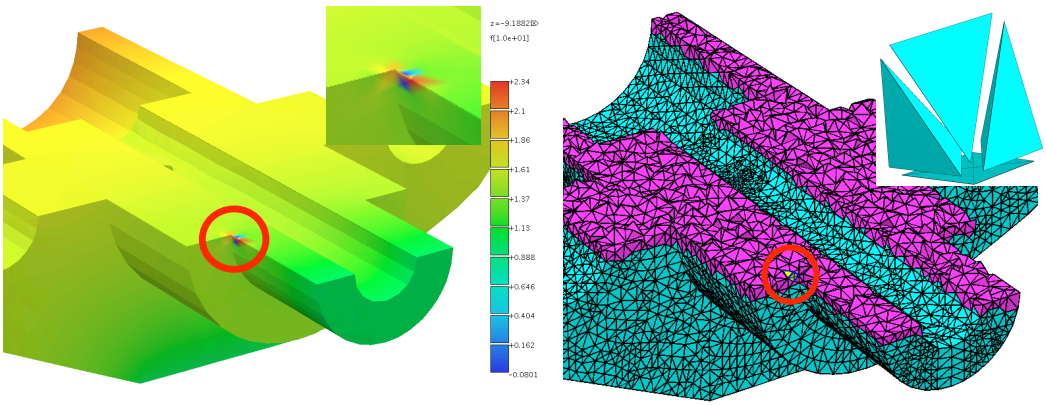


Figure 3: Locally distorted numerical solution due to slivers (very flat tetrahedra) contained in the mesh. Left: A numerical solution of the Laplace equation with non-negative Dirichlet boundary conditions. The annotated place has negative value. Right: The used tetrahedral mesh. On the annotated place exists a sliver (enlarged at top-right).

CDT with no Steiner points. This condition is useful in practice. It suggests that Steiner points are only needed on some of the input edges. In [48, 55], practical algorithms for recovering Delaunay edges are proposed. Steiner points are inserted in such a way that no unnecessarily short edges will be introduced.

Another key question: Assume it is known that P has a CDT without Steiner points. How to efficiently construct such a CDT? So far, Shewchuk proposed several algorithms for this purpose [47, 48, 50]. Among them, the flip-based facet insertion algorithm [50] has good performance. Si and Gärtner [55] proposed another incremental facet recovery algorithm which is practically efficient and easy to implement.

A complete algorithm for Problem 1.1 is discussed in Section 4.

1.2 Mesh Refinement

The purpose of finite element mesh generation is to generate a "suitable" mesh for obtaining numerical solutions with high accuracy at a low computational cost. Here the main issues are *mesh quality* and *mesh size* which will affect the accuracy and convergence of numerical methods.

Numerical analysis shows that elements having small or large angles are bad [1, 49]. A notable problem is that tetrahedral meshes may contain one type of badly-shaped tetrahedra, so-called *slivers* [7], which are tetrahedra whose volumes are close to zero. Slivers can have both very large (near 180°) and very small (close to 0°) dihedral angles which may cause big numerical errors, see Fig. 3 for an example. Many algorithms for generating quality Delaunay meshes may not avoid slivers [46, 42, 11].

In general the nature of the exact solution of a given problem is not known before-

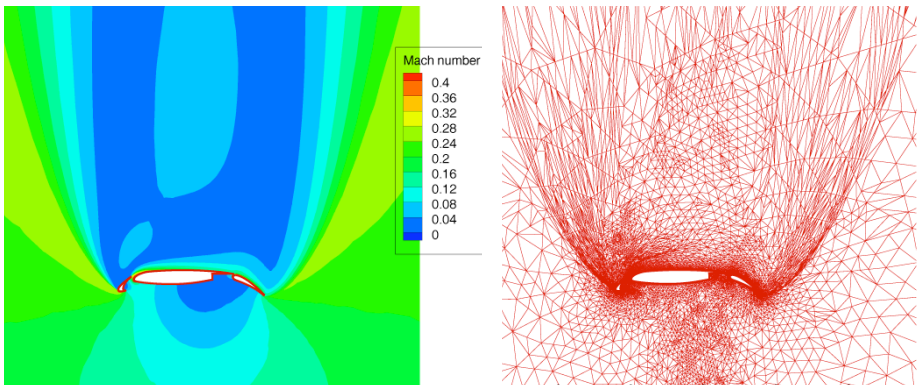


Figure 4: Adaptive numerical simulation of the lift versus angle-of-attack for a multi-element airfoil (`GGNS` [56]). Left: Mach distribution at 90° angle-of-attack. Right: An intermediate adapted mesh.

hand. Then it is not clear how to generate a mesh with a small number of degrees of freedom and resolve the detailed features of the solution, such as the edge or corner singularities and shock fronts. Adaptive numerical methods seek the best approximate solution at a low computational cost through a sequence of computed solutions on successively changed meshes, see e.g., [3, 32, 57, 2, 5]. Fig. 4 shows such an example.

Let $\mathbf{u}$ be the exact solution, and δ be a given tolerance. Each adaptive loop i mainly includes five subsequent phases: (1) the computation of an approximated solution $\mathbf{u}_i$ by FEM (or FVM), (2) the estimation of the error, e.g., $\mathbf{e}_i = \mathbf{u} - \mathbf{u}_i$, by an a posteriori error estimator, (3) the generation of a *mesh sizing function* H_i from $\mathbf{e}_i$, (4) the generation of an adapted mesh $\mathcal{T}_{i+1}$ conforming to H_i, and (5) the interpolation of the solution $\mathbf{u}_i$ on $\mathcal{T}_{i+1}$. The whole adaptive process can then be viewed as a non-linear optimization problem [32]. The goal is to find a mesh $\mathcal{T}_k$ with as few degrees of freedom as possible, and satisfies the termination criterion, e.g., $\|\mathbf{u} - \mathbf{u}_k\|_{L^2} \leq \delta$. The convergence of adaptive finite element methods for elliptic problems has been theoretically proved [18].

Mesh adaptation, i.e., the phase (4) in the above process, is one of the key steps in adaptive numerical methods. Let Ω be a three-dimensional simulation domain, and let the appropriate mesh sizing function H be defined over Ω and indicate the desired size of mesh elements, note that H may be anisotropic. One of the approaches to obtain a good quality tetrahedral mesh of Ω with the mesh size conforming to H, takes mainly three steps:

(1) construct an initial tetrahedral mesh $\mathcal{T}$ of Ω,
(2) add new points into $\mathcal{T}$ to conform H, and
(3) optimize $\mathcal{T}$ to improve the mesh quality.

Each of these steps can be extended into a more complex process and is a topic of interest in mesh generation. For a comprehensive survey of these technologies, we refer to [27] and [38]. In this work, we focus on how to efficiently perform the step *(2)* in the above process.

Problem 1.2 *Given an initial tetrahedral mesh $\mathcal{T}$ of a three-dimensional domain Ω and an isotropic mesh sizing function H defined on Ω, insert new points into $\mathcal{T}$ to form a good quality tetrahedral mesh $\mathcal{T}'$ of Ω such that the mesh size of $\mathcal{T}'$ conforms to H.*

A central question in the above problem is how to place the new points such that the requirements are simultaneously satisfied. Various approaches have been developed for this purpose, such as advancing-front methods [36, 37], Octree-based methods [60, 40], Delaunay-based methods [39, 14], and the combinations of them [58, 28]. Among these methods, the Delaunay-based methods which utilize the Delaunay criterion and the Delaunay triangulation [16] are the most robust and efficient. In Section 5 we present a point insertion algorithm based on the Delaunay refinement technique.

1.3 Outline

The rest of this article is organized as follows. Section 2 introduces a object called piecewise linear system which is used as an approximation of a mesh domain. The definition of constrained Delaunay tetrahedralization (CDT) is formalized in Section 3, some basic properties of CDTs are outlined. In Section 4, a CDT algorithm is presented and discussed in detail. In Section 5, an algorithm for refining a CDT into a good quality tetrahedral mesh is presented. We end this article by a list of open questions in Section 6.

2 Piecewise Linear Systems

A *physical domain* Ω in $\mathbb{R}^3$ used for numerical simulation is the volume enclosed by the *boundary* $\partial\Omega$ of Ω. Usually, $\partial\Omega$ may consist of arbitrarily shaped (e.g., curved) edges and surfaces. It is necessary that $\partial\Omega$ includes *internal boundaries* which may separate Ω into sub-domains so that the discontinuity between different materials can be modeled. Hence $\partial\Omega$ is in general not a topological manifold.

A *mesh domain* is an object such that it preserves the topology of Ω and it approximates Ω geometrically. Miller *et al.* [39] introduced a geometric object which uses convex polytopes as the main components. In the following, we define a generalization of this object to represent mesh domains with piecewise linear boundaries.

Convex polyhedra are well-defined as either the convex hull of a set of vertices or the intersection of a set of halfspaces [61]. We define a general and therefore not necessarily convex *polyhedron* P as the union of a finite set $\mathcal{P}$ of convex polyhedra, i.e., $P = \bigcup_{U\in\mathcal{P}} U$, and the space of P is connected. The *dimension* $\dim(P)$ is the largest dimension of a convex polyhedron in $\mathcal{P}$. Note that P may contain holes in

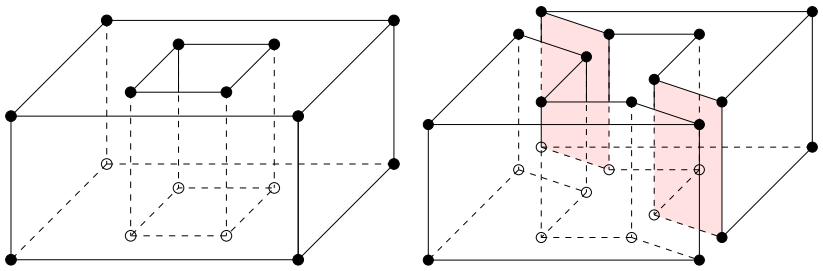


Figure 5: Polyhedra and faces. Left: A three-dimensional polyhedron (a torus) formed by the union of four convex polytopes. It consists of 16 vertices (zero-faces), 24 edges (1-faces), 10 two-faces (the faces at top and bottom are not simply connected), and 1 three-face (which is the object). Right: Two three-dimensional polyhedra. Each one has 12 vertices, 18 edges, 8 two-faces, and 1 three-face. The shaded area highlights two 2-faces whose points have the same face figures.

its interior. Whatever, we require that the space of P must be connected. See Fig. 5 for examples.

We follow the definition of faces of a polyhedron by Edelsbrunner [22] with a minor modification in the connectedness of the faces.

For a point $\mathbf{x}$ in a polyhedron P we consider a sufficiently small neighborhood $N_\epsilon(\mathbf{x}) = (\mathbf{x} + \mathbb{B}(\mathbf{0}, \epsilon)) \cap P$. The *face figure* of $\mathbf{x}$ is the enlarged version of this neighborhood within P, i.e., $\mathbf{x} + \bigcup_{\lambda>0} \lambda(N_\epsilon(\mathbf{x}) - \mathbf{x})$. A *face* F of a polyhedron P is the closure of a maximal connected set of points with identical face figures. See Fig. 5 for examples.

A face F of P is again a polyhedron. Particularly, $\emptyset$ is a face of P. If all convex polyhedra in $\mathcal{P}$ have the same dimension, then P itself is a face of P. All other faces of P are *proper* faces of P. The faces of dimension 0, 1, $\dim(P) - 2$, and $\dim(P) - 1$ are called *vertices*, *edges*, *ridges*, and *facets*, respectively. The set of all vertices of P, the *vertex set*, will be denoted by $\mathrm{vert}(P)$. The union of all proper faces of P is called the *boundary* of P, denoted as $\mathrm{bd}(P)$. The *interior* $\mathrm{int}(P)$ is $P - \mathrm{bd}(P)$.

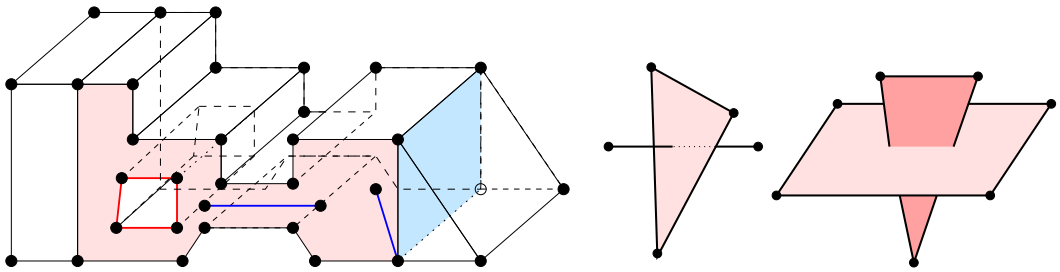
We define a *piecewise linear system* (abbreviated as PLS) to be a finite collection $\mathcal{X}$ of polyhedra with the following properties

(i) $P \in \mathcal{X} \Longrightarrow$ all faces of P are in $\mathcal{X}$,

(ii) $P, Q \in \mathcal{X} \Longrightarrow \exists K \in \mathcal{X}$, s.t. $P \cap Q = \cup_{K \in \mathcal{X}} K$, and

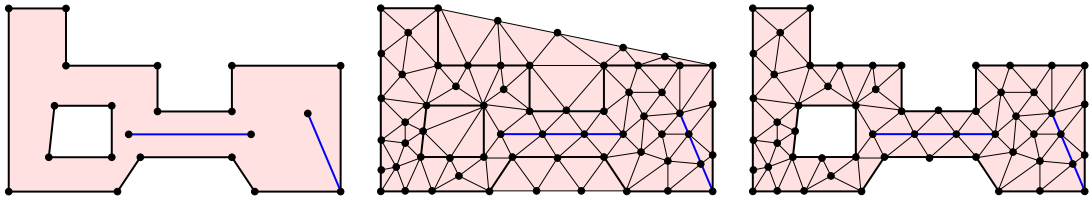
(iii) $\dim(P \cap Q) = \dim(P), P \neq Q \Longrightarrow P \subseteq Q$ and $\dim(P) < \dim(Q)$.

This definition generalizes the one introduced by Miller *et al.* [39] by allowing non-convex polyhedra. PLSs are flexible for representing non-manifold objects. The properties *(i)* and *(ii)* are essential, they ensures that a PLS is closed by both taking boundaries and taking intersections. The property *(ii)* is relaxed from that



(a) A 3D PLS (b) Non-PLSs

Figure 6: (a) A three-dimensional PLS $\mathcal{X}$. The pink area highlights a facet in $\mathcal{X}$, which is a non-convex polygon with a hole in its interior. Moreover, this face contains two floated segments in $\mathcal{X}$ (shown in blue) in its interior. The light blue area shows an internal facet in $\mathcal{X}$. (b) Two non-PLC objects. They are not closed under intersections.



(a) A 2D PLS $\mathcal{X}$ (b) A triangulation of $\mathcal{X}$ (c) A mesh of $\mathcal{X}$

Figure 7: Triangulation and mesh.

of a complex. Since two non-convex polyhedra P and Q may intersect at more than one faces of them, $P \cap Q$ is a subset of $\mathcal{X}$. *(iii)* is an extra property for a PLS which makes it more flexible. For example, it allows that a cube encloses an edge in its interior with no need to further decompose it. Furthermore, it excludes the case that two polyhedra having the same dimension overlap each other. See Fig. 6 for examples.

The dimension of a PLS $\mathcal{X}$, denoted as $\dim(\mathcal{X})$, is the largest dimension of its polyhedra. A *subsystem* of $\mathcal{X}$ is a subset of $\mathcal{X}$ which is again a PLS. A particular subsystem is the *i-skeleton*, $\mathcal{X}^{(i)}$, of $\mathcal{X}$ which consists of all polyhedra of $\mathcal{X}$ whose dimensions $\leq i$. For example, $\mathcal{X}^{(0)}$ is the *vertex set*, denoted as $\mathrm{vert}(\mathcal{X})$, of $\mathcal{X}$. The *boundary system*, denoted as $\partial\mathcal{X}$, of $\mathcal{X}$ is the $(\dim(\mathcal{X}) - 1)$-skeleton of $\mathcal{X}$. The *underlying space* of $\mathcal{X}$ is $|\mathcal{X}| = \bigcup_{P \in \mathcal{X}} P$. Note that $|\mathcal{X}| \subseteq \mathbb{R}^d$ is a topological subspace of $\mathbb{R}^d$. The collection $\mathcal{X}$ gives a special topology on $|\mathcal{X}|$, refer to [54].

Given a physical domain Ω, we use a PLS $\mathcal{X}$ to represent it such that Ω and $|\mathcal{X}|$ are homeomorphic (i.e., they are topologically equivalent) and the shape of Ω is "approximated by $|\mathcal{X}|$ geometrically".

A *triangulation* of a PLS $\mathcal{X}$ is a simplicial complex $\mathcal{T}$ such that the underlying space of $\mathcal{T}$ equals to the convex hull of the vertices of $\mathcal{X}$ and every polyhedron of $\mathcal{X}$ is represented by a subcomplex of $\mathcal{T}$. More formally,

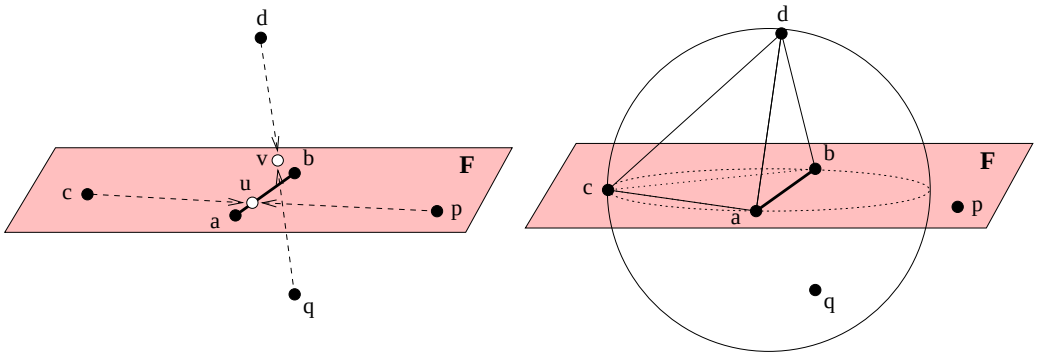


Figure 8: Visibility and constrained Delaunay criterion. The shaded region is a facet F of a PLS $\mathcal{X}$ in $\mathbb{R}^3$, $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{p} \in F$. $\mathbf{ab}$ is a segment of $\mathcal{X}$. Left: $\mathbf{d}$ and $\mathbf{q}$ are invisible to each other since $\mathbf{dq} \cap F = \mathbf{v}$. $\mathbf{c}$ and $\mathbf{p}$ are invisible to each other since $\mathbf{cp} \cap \mathbf{ab} = \mathbf{u}$. $\mathbf{u}$ sees both $\mathbf{c}$ and $\mathbf{p}$. Right: A circumball of the tetrahedron $\mathbf{abcd}$ contains $\mathbf{q}$. $\mathbf{abcd}$ is constrained Delaunay since $\mathbf{q}$ is not visible from its interior. The triangle $\mathbf{abc} \subset F$ is constrained Delaunay since $\mathbf{p}$ is outside its diametric ball.

(i) $|\mathcal{T}| = \text{conv}(\text{vert}(\mathcal{X}))$, and

(ii) $\forall P \in \mathcal{X} \Longrightarrow \exists \mathcal{K} \subseteq \mathcal{T}$ such that $|\mathcal{K}| = P$.

Note that $\mathcal{T}$ may contain Steiner points. We define a *mesh* of $\mathcal{X}$ to be a subcomplex $\mathcal{K}$ of $\mathcal{T}$ such that $|\mathcal{K}| = |\mathcal{X}|$. According to our definitions, a triangulation of a set S of vertices triangulates the convex hull of S, while a mesh of S is just a partition of S itself. See Fig. 7 for examples. Our output object is either a triangulation or a mesh of the input PLS.

3 Constrained Delaunay Tetrahedralizations

Constrained Delaunay triangulations are first studied by Lee and Lin [34] and Chew [13] for generating two-dimensional Delaunay-like triangulations from planar straight line graphs (a 1-dimensional PLS). The same concept can be generalized into three and higher dimensions. However it is necessary to take Steiner points into account.

A crucial concept is the *visibility* of points in $\mathbb{R}^3$. The basic idea is: every polyhedron $P \in \mathcal{X}$ may block the visibility of points which are not in P, while P does not block the visibility for its own points. Two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ are *invisible* to each other if the interior of the line segment $\mathbf{xy}$ intersects a polyhedron $P \in \mathcal{X}$ at a single point. Otherwise $\mathbf{x}$ and $\mathbf{y}$ are *visible* to each other. See Fig. 8 left for examples.

The next definition, referred to as the *constrained Delaunay criterion*, relaxes the Delaunay criterion. Let S be a finite set of points and $\mathcal{X}$ be a PLS in $\mathbb{R}^3$ with $\text{vert}(\mathcal{X}) \subseteq S$. A simplex σ whose vertices are in S is *constrained Delaunay* if it is in one of the two cases:

(i) There is a circumball B_σ of σ contains no vertices of S in its interior.

(ii) There exists $F \in \mathcal{X}$, such that $\text{int}(\sigma) \subseteq \text{int}(F)$. Let $K = S \cap \text{aff}(F)$, then no vertex of K contained in the interior of B_σ is visible from any point in $\text{int}(\sigma)$.

Case *(i)* means that every Delaunay simplex is also constrained Delaunay. In *(ii)*, F is the lowest-dimensional polyhedron of $\mathcal{X}$ that contains σ, K is the subset of S in the affine hull generated by F. The fact that a simplex $\sigma \subset F$ is constrained Delaunay or not only depends on the vertices of K. See Fig. 8 right for examples.

A *constrained Delaunay tetrahedralization* (abbreviated as CDT) of $\mathcal{X}$ is defined as a tetrahedralization $\mathcal{T}$ of $\mathcal{X}$ such that every simplex of $\mathcal{T}$ is constrained Delaunay.

By this definition, a CDT of $\mathcal{X}$ may contain Steiner points, i.e., points in $S \backslash \text{vert}(\mathcal{X})$. It is called a *pure* CDT if it does not contain Steiner points. A 2-dimensional pure CDT is the same as the one defined by Lee and Lin [34] and Chew [12]. Shewchuk's definition of a CDT [51] is also a pure CDT. It is well known that a pure CDT of a 3-dimensional PLS may not exist while there are infinitely many CDTs of $\mathcal{X}$ with Steiner points.

In the following, we introduce some basic properties of the CDTs we have just defined. These properties show that a CDT of a PLS $\mathcal{X}$ is very close to a conforming Delaunay triangulation of $\mathcal{X}$. The proofs are omitted, they are found in [54].

Delaunay triangulations can be checked locally. This is true for CDTs as well. Let $\mathcal{T}$ be any tetrahedralization of a three-dimensional PLS $\mathcal{X}$. A 2-simplex σ of $\mathcal{T}$ is called *locally Delaunay* if either *(i)* σ is on the boundary of the convex hull, or *(ii)* $\sigma \subset |\partial\mathcal{X}|$, or *(iii)* the opposite vertex of τ is not in $\text{int}(B_\nu)$ of ν, where $\tau, \nu \in \mathcal{T}$ are the unique simplices such that $\sigma = \tau \cap \nu$. Note that *(ii)* implies that one can ignore the 2-simplices contained in the boundary of $|\mathcal{X}|$.

Theorem 3.1 (Constrained Delaunay Lemma [54]) *If every $(d-1)$-simplex of $\mathcal{T}$ is locally Delaunay, then $\mathcal{T}$ is a CDT of $\mathcal{X}$.*

If a point set S in $\mathbb{R}^3$ is in general position, i.e., no 5 points of S share a common 2-sphere, then the Delaunay tetrahedralization of S is unique. This property holds in CDT as well.

Corollary 3.2 *Let $\mathcal{T}$ be a CDT of $\mathcal{X}$. If vert($\mathcal{T}$) is in general position, then $\mathcal{T}$ is the unique CDT of $\mathcal{X}$ with the set of vertices of $\mathcal{T}$.*

The i-skeleton $\mathcal{X}^{(i)}$ of $\mathcal{X}$ is an i-dimensional PLS, where $0 \leq i \leq 2$. It is useful to know the properties of a CDT of $\mathcal{X}^{(i)}$. We call a triangulation of $\mathcal{X}$ *conforming Delaunay triangulation* if every simplex of $\mathcal{T}$ is Delaunay.

Theorem 3.3 ([54]) *Let $\mathcal{X}$ be a d-dimensional PLS.*
(i) *A CDT of $\mathcal{X}^{(2)}$ is a CDT of $\mathcal{X}$.*
(ii) *A CDT of $\mathcal{X}^{(i)}$ is a conforming Delaunay triangulation of $\mathcal{X}^{(i)}$, where $i = 0, 1$.*

4 The CDT Algorithm

Let $\mathcal{X}$ be a three-dimensional PLS, i.e., $\mathcal{X}$ is a set of polyhedra of dimensions from 0 to 3. We call 1- and 2-polyhedra of $\mathcal{X}$ *segments* and *facets*. The algorithm to construct a constrained Delaunay tetrahedralization $\mathcal{T}$ of $\mathcal{X}$ works in the following steps:

1. Initialize a CDT $\mathcal{D}_0$ of $\mathcal{X}^{(0)}$.
2. Let $\mathcal{D}_1 = \mathcal{D}_0$. Recover segments of $\mathcal{X}$ in $\mathcal{D}_1$ such that $\mathcal{D}_1$ is a CDT of $\mathcal{X}^{(1)}$, such that every segment is a union of edges in $\mathcal{D}_1$.
3. Let $\mathcal{D}_2 = \mathcal{D}_1$. Recover facets of $\mathcal{X}$ in $\mathcal{D}_2$ such that $\mathcal{D}_2$ is a CDT of $\mathcal{X}^{(2)}$, such that every facet is a union of faces in $\mathcal{D}_2$.

This algorithm proceeds in the increasing order of the dimensions of the skeletons. It initializes a CDT of $\mathcal{X}^{(0)}$ (which is a Delaunay tetrahedralization of $\text{vert}(\mathcal{X})$). The next two steps incrementally construct a CDT $\mathcal{D}_i$ of $\mathcal{X}^{(i)}$ from a CDT $\mathcal{D}_{i-1}$, where $i = 1, 2$. By Theorem 3.3, $\mathcal{D}_2$ is a CDT of $\mathcal{X}$. A constrained Delaunay mesh of $\mathcal{X}$ can be obtained by removing the simplices of $\mathcal{D}_2$ not contained to $|\mathcal{X}|$.

4.1 Segment Recovery

The Delaunay tetrahedralization $\mathcal{D}_0$ of $\text{vert}(\mathcal{X})$ may not contain all segments of $\mathcal{X}$. This section presents a segment recovery algorithm for recovering missing segments of $\mathcal{X}$. The inputs are $\mathcal{X}^{(1)}$ and $\mathcal{D}_0$. The output of this algorithm is a CDT $\mathcal{D}_1$ of $\mathcal{X}^{(1)}$. Hence every segment of $\mathcal{X}$ is a union of edges of $\mathcal{D}_1$. For the mesh quality requirement, it is desired that no unnecessarily short edge is introduced in $\mathcal{D}_1$.

We need some definitions. A vertex of $\mathcal{X}$ is *acute* if at least two segments of $\mathcal{X}$ incident at it form an angle less than 60°. We distinguish two types of segments in $\mathcal{X}$: a segment is *type-0* if its both endpoints are not acute, it is *type-1* if exactly one of its endpoints is acute. If both endpoints of a segment are acute, it is treated as a type-0 segment at the beginning and it is transformed into two type-1 segments immediately after a Steiner point is inserted to it.

Let $\mathbf{e}_i\mathbf{e}_j$ be a segment of $\mathcal{X}$ with endpoints $\mathbf{e}_i$ and $\mathbf{e}_j$. $\mathbf{e}_i\mathbf{e}_j$ is split by adding a Steiner point in the interior of it. The two resulting edges are called *subsegments* of $\mathbf{e}_i\mathbf{e}_j$. Subsegments inherit types from the original segments. For example, if $\mathbf{e}_i\mathbf{e}_j$ is a subsegment of $\mathbf{e}_1\mathbf{e}_2$ which is a type-1 segment, $\mathbf{e}_i\mathbf{e}_j$ is also type-1 exen if none of its endpoints is acute. For any vertex $\mathbf{v}$ inserted on a type-1 segment (or subsegment), let $R(\mathbf{v})$ denote its original acute vertex. If $\mathbf{v}$ is an input vertex, then $R(\mathbf{v}) = \mathbf{v}$. A tacit rule is used throughout this section, if $\mathbf{e}_i\mathbf{e}_j$ is a type-1 segment, it implies that either $\mathbf{e}_i$ or $R(\mathbf{e}_i)$ is acute. In the following, unless it is explicitly mentioned, the term "segment" means either a segment or a subsegment.

The *diametric circumball* of a segment is by definition the smallest circumscribed ball of it. A vertex is said to *encroach upon* a segment if it lies inside the diametric circumball of that segment. We have the following fact.

Fact 4.1 *If a segment of $\mathcal{X}$ is missing in $\mathcal{D}_0$ and vert($\mathcal{D}_0$) is in general position, then it must be encroached by at least one vertex of $\mathcal{D}_0$.*

Let $\mathbf{e}_i\mathbf{e}_j$ be a missing segment. Let P be the set of all encroaching points of $\mathbf{e}_i\mathbf{e}_j$. A *reference point* $\mathbf{p}$ of $\mathbf{e}_i\mathbf{e}_j$, which is used for a splitting point in $\mathbf{e}_i\mathbf{e}_j$, is defined as follows

(i) $\mathbf{p} \in P$, and

(ii) the angle between $\mathbf{pe}_i$ and $\mathbf{pe}_j$ is maximized for all $\mathbf{p} \in P$.

Notice that $\mathbf{p}$ may not be unique (because several points can share the same sphere). In this case randomly choose one to be $\mathbf{p}$.

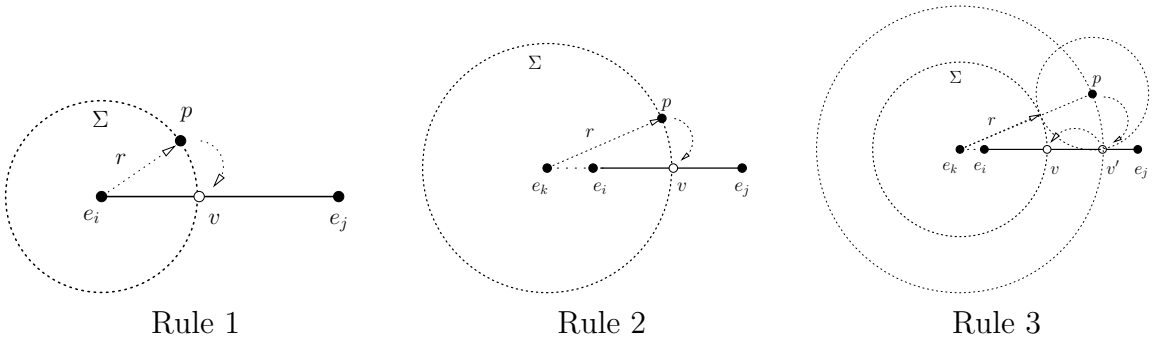
Let $\mathbf{p}$ be the reference point of a missing segment $\mathbf{e}_i\mathbf{e}_j$. The choice of a splitting point $\mathbf{v}$ is governed by three rules given below. Let $\Sigma(\mathbf{c}, r)$ be a sphere with center $\mathbf{c}$ and radius r, and let $\|\cdot\|$ be the Euclidean distance function.

1. $\mathbf{e}_i\mathbf{e}_j$ is type-0 (Fig. 9 left), then $\mathbf{v} = \mathbf{e}_i\mathbf{e}_j \cap \Sigma(\mathbf{c}, r)$, where
   if $\|\mathbf{e}_i - \mathbf{p}\| < \frac{1}{2}\|\mathbf{e}_i - \mathbf{e}_j\|$ **then**
   $\mathbf{c} = \mathbf{e}_i, r = \|\mathbf{e}_i - \mathbf{p}\|$;
   else if $\|\mathbf{e}_j - \mathbf{p}\| < \frac{1}{2}\|\mathbf{e}_i - \mathbf{e}_j\|$ **then**
   $\mathbf{c} = \mathbf{e}_j, r = \|\mathbf{e}_j - \mathbf{p}\|$;
   else
   $\mathbf{c} = \mathbf{e}_i, r = \frac{1}{2}\|\mathbf{e}_i - \mathbf{e}_j\|$;
   end.

2. $\mathbf{e}_i\mathbf{e}_j$ is type-1 (Fig. 9 middle), let $\mathbf{e}_k = R(\mathbf{e}_i)$, then $\mathbf{v} = \mathbf{e}_k\mathbf{e}_j \cap \Sigma(\mathbf{c}, r)$, where $\mathbf{c} = \mathbf{e}_k$ and $r = \|\mathbf{e}_k - \mathbf{p}\|$. However, **if** $\|\mathbf{v} - \mathbf{e}_j\| < \|\mathbf{v} - \mathbf{p}\|$, **then** reject $\mathbf{v}$ and use Rule 3; **end**.

3. (Continued from Rule 2) Let $\mathbf{v}'$ be the vertex rejected by Rule 2 (Fig. 9 right), then $\mathbf{v} = \mathbf{e}_k\mathbf{e}_j \cap \Sigma(\mathbf{c}, r)$, where $\mathbf{c} = \mathbf{e}_k$, and
   if $\|\mathbf{p} - \mathbf{v}'\| < \frac{1}{2}\|\mathbf{e}_i - \mathbf{v}'\|$ **then**
   $r = \|\mathbf{e}_k - \mathbf{e}_i\| + \|\mathbf{e}_i - \mathbf{v}'\| - \|\mathbf{p} - \mathbf{v}'\|$;
   else
   $r = \|\mathbf{e}_k - \mathbf{e}_i\| + \frac{1}{2}\|\mathbf{e}_i - \mathbf{v}'\|$;
   end.

The idea of these segment splitting rules is to avoid short edges, and the choice of the locations should not cause an endless loop. All the three rules guarantee that the newly inserted vertex is not too close to the existing vertices. Note that Rule 1



Rule 1 Rule 2 Rule 3

Figure 9: Segment splitting rules.

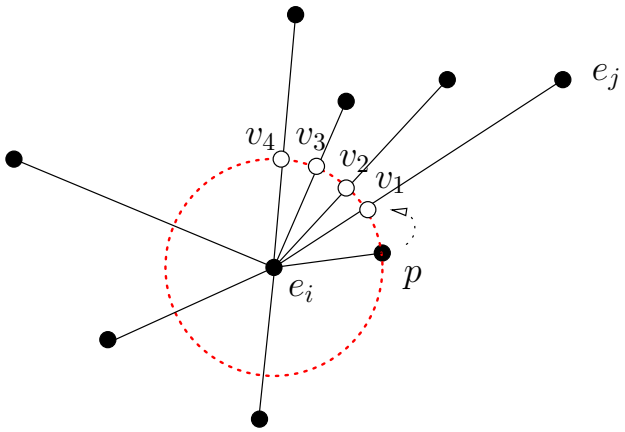


Figure 10: The protecting ball of an acute vertex $\mathbf{e}_i$. $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, and $\mathbf{v}_4$ are points inserted on segments (by rule 2) sharing $\mathbf{e}_i$. They automatically create a protecting ball of $\mathbf{e}_i$.

and 2 never create an edge shorter than the distance $\|R(\mathbf{e}_i) - v\|$. Rule 3 may create an edge which has a length of at most one third of the length of $\|R(\mathbf{e}_i) - \mathbf{e}_j\|$. Our analysis showed that the total number of application of Rule 3 is bounded.

For several segments sharing an acute vertex, by repeatedly using Rule 2 or 3, a protecting ball is automatically created which ensures: no other vertex can be inserted inside the ball. The effect is shown in Figure 10. Notice, the protecting ball is not necessarily completely created, only the missing segments will be split and protected. Existing segments remain untouched. This reduces the number of Steiner points.

The SegmentRecovery algorithm is described in Fig. 11. The inputs are a three-dimensional PLS $\mathcal{X}$ and a Delaunay tetrahedralization $\mathcal{D}_0$ of vert($\mathcal{X}$). Some segments of $\mathcal{X}$ may be missing in $\mathcal{D}_0$. The algorithm first initializes a queue Q of all segments of $\mathcal{X}$. Then the algorithm runs into a loop until Q is empty.

For each segment (or subsegment) $\mathbf{e}_i\mathbf{e}_j \in Q$. If it is missing in $\mathcal{D}_1$, a Steiner point $\mathbf{v}$ inside $\mathbf{e}_i\mathbf{e}_j$ is generated by one of the three rules described in previous section

SegmentRecovery ($\mathcal{X}$, $\mathcal{D}_0$)
// $\mathcal{X}$ is a three-dimensional PLS; $\mathcal{D}_0$ is the DT of vert($\mathcal{X}$).
1. $\mathcal{D}_1 = \mathcal{D}_0$;
2. Initialize a queue Q of all segments of $\mathcal{X}$;
3. **while** $Q \neq \emptyset$ **do**
4. get a segment $\mathbf{e}_i\mathbf{e}_j \in Q$; $Q = Q \setminus \{\mathbf{e}_i\mathbf{e}_j\}$;
5. **if** $\mathbf{e}_i\mathbf{e}_j$ is missing in $\mathcal{D}_1$, **then**
6. find a Steiner point $\mathbf{v} \in \text{int}(\mathbf{e}_i\mathbf{e}_j)$ by Rule i, $i \in \{1, 2, 3\}$;
7. $Q = Q \cup \{\mathbf{e}_i\mathbf{v}, \mathbf{e}_j\mathbf{v}\}$;
8. $Q = Q \cup \{\sigma \in \mathcal{X}^{(1)}, \sigma \in \mathcal{D}_1 \,|\, \mathbf{v} \in \text{int}(B_\sigma)\}$;
9. update $\mathcal{D}_1$ to be the DT of vert($\mathcal{D}_1$) $\cup \{\mathbf{v}\}$;
10. **endif**
11. **endwhile**
12. **return** $\mathcal{D}_1$;

Figure 11: The segment recovery algorithm. The B_σ (in line 8) means the diametric circumball of the segment σ.

(Fig. 11, line 6). The new point $\mathbf{v}$ will split $\mathbf{e}_i\mathbf{e}_j$ into two subsegments $\mathbf{e}_i\mathbf{v}$ and $\mathbf{e}_j\mathbf{v}$, they are queued in Q (line 7). Moreover, the insertion of $\mathbf{v}$ may cause other existing segments (subsegments) of $\mathcal{X}$ missing in $\mathcal{D}_1$, they are queued in Q and will be recovered later (line 8). Then $\mathcal{D}_1$ is updated to a Delaunay tetrahedralization of the vertex set including $\mathbf{v}$ (line 9).

The termination of this algorithm can be proved by showing that the length of every subsegment is bounded by some positive value depending only on the input. Please refer to [54] for the details.

4.2 Facet Recovery

Each facet $F \in \mathcal{X}$ together with the Steiner points inserted on F is first triangulated into a two-dimensional CDT $\mathcal{T}_F$. Hence $\partial\mathcal{X}$ is triangulated into a triangulation $\mathcal{F}$. We call triangles of $\mathcal{F}$ *subfaces* to distinguish other faces of $\mathcal{D}_2$. Some subfaces may be missing in $\mathcal{D}_2$. The facet recovery algorithm incrementally recovers missing subfaces of $\mathcal{F}$.

At initialization, let $\mathcal{D}_2 = \mathcal{D}_1$; add all missing subfaces of $\mathcal{F}$ into a set $\mathcal{S}$. The algorithm iteratively recovers the subfaces in $\mathcal{S}$ and updates $\mathcal{D}_2$, it stops when $\mathcal{S}$ is empty.

At each iteration i, several missing subfaces are recovered together. We define a *missing region* Ω to be a set of subfaces of $\mathcal{F}$ such that

(i) all subfaces in Ω are coplanar,
(ii) the edges on $\partial\Omega$ are edges of $\mathcal{D}_2$, and
(iii) the edges in int(Ω) are missing in $\mathcal{D}_2$.

Hence Ω is a connected set of missing subfaces. It may not be simply connected,

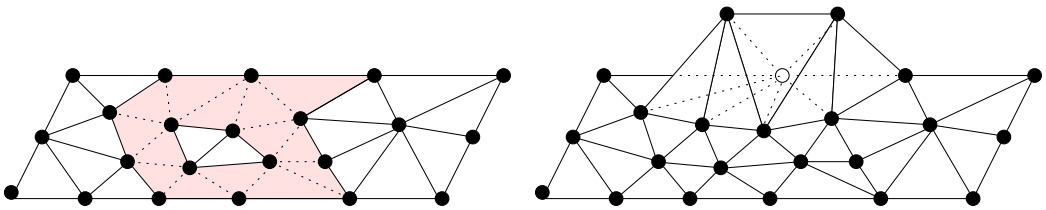


Figure 12: Left: The shaded area highlights a missing region Ω. Right: One of the cavities resulting from a missing region is illustrated.

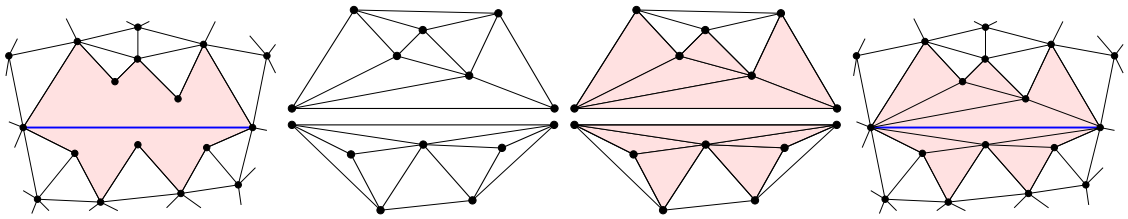


Figure 13: The FacetRecovery algorithm (illustrated in 2D). From left to right, the two initial cavities C_1 and C_2 separated by a segment, the initial Delaunay triangulations $\mathcal{D}_{C_1}$ and $\mathcal{D}_{C_2}$, triangles of $\mathcal{D}_{C_1}$ and $\mathcal{D}_{C_1}$ are classified as "inside" or "outside", and the new partitions of C_1 and C_2.

i.e., Ω can contain a hole inside. Each missing subface belongs to a missing region. A facet can have more than one missing regions.

When a missing region Ω is found, one can derive a cavity in $|\mathcal{D}_2|$ by removing all its tetrahedra whose interiors intersect with Ω. This cavity can be further subdivided into two cavities by inserting the subfaces of Ω in it, see Fig. 12 right. Each cavity is a three-dimensional polyhedron C whose facets are triangles, some of them are subfaces of $\mathcal{F}$.

The next step is to tetrahedralize each cavity C without using Steiner points. The TetrahedralizeCavity subroutine first constructs the Delaunay tetrahedralization $\mathcal{D}_C$ of vert($\mathcal{C}$) (line 1). Next it removes those tetrahedra of $\mathcal{D}_C$ which are not in the interior of C from $\mathcal{D}_C$ (lines $2-6$). On finish, the remaining tetrahedra in $\mathcal{D}_C$ form a partition of C.

Subroutine TetrahedralizeCavity (C)
// C is a cavity (a polyhedron with triangular facets).
1. form the Delaunay tetrahedralization $\mathcal{D}_C$ of vert(C);
2. **for** each tetrahedron $\tau \in \mathcal{D}_C$, **do**
3. **if** $\tau \not\subseteq$ int(C), **then**
4. $\mathcal{D}_C = \mathcal{D}_C \setminus \{\tau\}$;
5. **endif**
6. **endfor**
7. **return** $\mathcal{D}_C$;

The FacetRecovery algorithm first initializes a set $\mathcal{S}$ of all subfaces of $\mathcal{X}$. Then it

Algorithm FacetRecovery ($\mathcal{X}$, $\mathcal{D}_1$)
// $\mathcal{X}$ is a three-dimensional PLS; $\mathcal{D}_1$ is the CDT of $\mathcal{X}^{(1)}$.
1. $\mathcal{D}_2 = \mathcal{D}_1$;
2. Initialize a set $\mathcal{S}$ of all subfaces of $\mathcal{X}$;
3. **while** $\mathcal{S} \neq \emptyset$ **do**
4. get a subface $\sigma \in \mathcal{S}$; $\mathcal{S} = \mathcal{S} \setminus \{\sigma\}$;
5. **if** σ is missing in $\mathcal{D}_2$, **then**
6. form a missing region Ω containing σ;
7. $\mathcal{D}'_2 = \mathcal{D}_2 \setminus \{\tau \in \mathcal{D}_2 \,|\, \text{int}(\tau) \cap \Omega \neq \emptyset\}$;
8. form two cavities C_1, C_2 (in $|\mathcal{D}_2|$), where $C_1 \cap C_2 = \Omega$;
9. $\mathcal{D}_{C_1}$ = TetrahedralizeCavity(C_1);
10. $\mathcal{D}_{C_2}$ = TetrahedralizeCavity(C_2);
11. $\mathcal{D}_2 = \mathcal{D}'_2 \cup \mathcal{D}_{C_1} \cup \mathcal{D}_{C_2}$;
12. **endif**
13. **endwhile**
14. **return** $\mathcal{D}_2$;

runs into a loop until $\mathcal{S}$ is empty. Once a subface σ is found missing in $\mathcal{D}_2$, a missing region Ω containing σ is formed (line 6). All tetrahedra crossing Ω are removed from $\mathcal{D}_2$ (line 7) resulting in a temporary object $\mathcal{D}'_2$. Two cavities C_1 and C_2 separated by Ω are formed in the interior of $|\mathcal{D}_2|$ (line 8). Then C_1 and C_2 are partitioned into two sets ($\mathcal{D}_{C_1}$ and $\mathcal{D}_{C_2}$) of tetrahedra by the subroutine TetrahedralizeCavity (lines 9 and 10), respectively. $\mathcal{D}_2$ is then updated to conform Ω with the new partitions of C_1 and C_2 (lines 11). Fig. 13 illustrates the idea of this algorithm in two dimensions.

4.3 Correctness

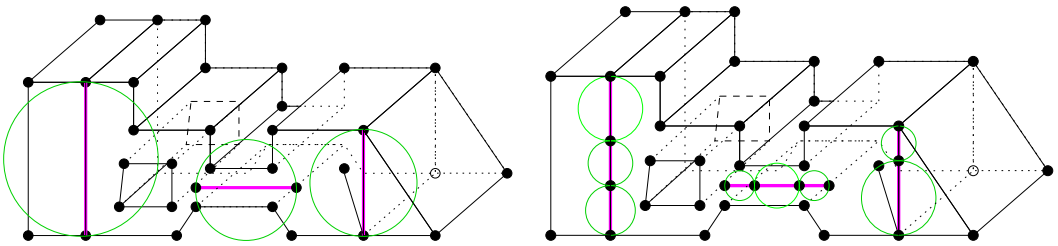
In this algorithm, Steiner points are only introduced in the step *2* (segment recovery). In order to show the correctness of this algorithm, we will first need the following assumption.

Assumption 4.2 *Assume the vertex set vert($\mathcal{D}_1$) is in general position, i.e., no five vertices of vert($\mathcal{D}_1$) share a common sphere.*

Although this assumption is very strong, it is easy to be satisfied by applying the techniques of symbolic perturbations [25, 48, 17] in both steps *1* and *2*. Hence, (theoretically) there is no need to actually perturb the vertices.

The following theorem proved by Shewchuk [45] ensures the correctness of the algorithm. Let S be a finite set of vertices. A simplex σ whose vertices are in S is called *strongly Delaunay* if there exists a circumscribed ball B_σ of σ, such that $B_\sigma \cap S = \emptyset$.

Theorem 4.3 ([45]) *If every segment of a PLS $\mathcal{Y}$ is strongly Delaunay in vert($\mathcal{Y}$), then $\mathcal{Y}$ has a CDT with no Steiner points.*



(a) A PLS $\mathcal{X}$ (b) A PLS $\mathcal{Y}$

Figure 14: Shewchuk's CDT Theorem [45]. (a): The PLS $\mathcal{X}$. Several segments of $\mathcal{X}$ are not Delaunay. (b): A refined PLS $\mathcal{Y}$ which is topologically and geometrically equivalent to $\mathcal{X}$. All segments of $\mathcal{Y}$ are Delaunay.

After step *2* of the algorithm, each segment of $\mathcal{X}$ is a union of edges in $\mathcal{D}_1$. Moreover, $\mathcal{D}_1$ is the Delaunay tetrahedralization of vert($\mathcal{D}_1$). Hence, each subsegment of $\mathcal{X}$ is strongly Delaunay (by the Assumption 4.2) in $\mathcal{D}_1$. Let $\mathcal{Y}$ be a PLS which is the *refinement* of $\mathcal{X}$, that is, each segment of $\mathcal{X}$ is a union of segments in $\mathcal{Y}$, and the segments of $\mathcal{Y}$ are all edges in $\mathcal{D}_1$. Then $\mathcal{Y}$ has a CDT with no Steiner points. See Fig. 14 for an example.

This condition is also useful in practice since it suggests that the Steiner points can be inserted on segments only.

Once the existence of a CDT with no Steiner points is known, we still need to show that the step *3*, i.e., facet recovery, can be done without using Steiner points. In [54], we have proved the following guarantees for the facet recovery algorithm.

Lemma 4.4 ([54]) *Assume the old tetrahedralization $\mathcal{D}_2$ is a CDT and it satisfies the assumption 4.2. Then the two calls of* TetrahedralizeCavity *subroutines (in lines* 9 *and* 10*) succeed.*

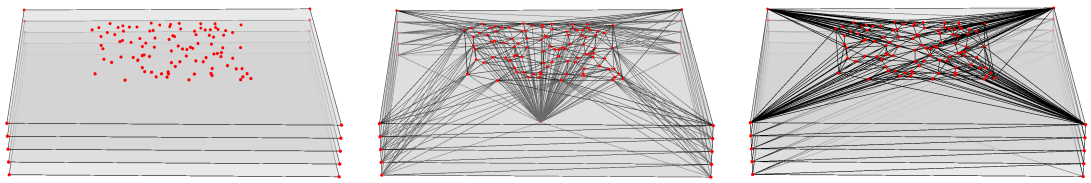
Lemma 4.5 ([54]) *The new tetrahedralization $\mathcal{D}_2$ (in line* 11*) is a CDT.*

Now we can prove the termination of the FacetRecovery algorithm by combining Lemma 4.4 and Lemma 4.5.

Theorem 4.6 ([54]) *The* FacetRecovery *algorithm terminates and results in a CDT of $\mathcal{X}$.*

4.4 Complexity

In this section we show the worst case behavior of the FacetRecovery algorithm with respect to the number of vertices and facets of the input PLS.



(a) $\mathcal{L}_{100,5}$ (5 layers) (b) The DT (c) The CDT

Figure 15: Examples (Layers). The PLS $\mathcal{L}_{100,5}$ shown in (a) has 5 parallel facets, 100 vertices lie above the top facet, one vertex below the center of the bottom facet. (b) and (c) respectively show the Delaunay tetrahedralization and the CDT of $\mathcal{L}_{100,5}$.

Theorem 4.7 ([54]) *Let $\mathcal{X}$ be a three-dimensional PLS which has v vertices and f facets. The* FacetRecovery *algorithm runs in time $O(fv^2 \log v)$.*

The prove that the complexity estimate is sharp we have to construct is completed by construct a PLS which needs such running time, then showing that it is indeed the worst case. Such an example is shown in Fig. 15. In this example $\mathcal{L}_{100,5}$, all facets will be missing from the Delaunay tetrahedralization of its points. We get the upper bound of the running time by recovering missing facets from bottom to top. The cavity formed from each facet has size $O(v)$. The Delaunay tetrahedralization has the worst case running time $O(v^2)$.

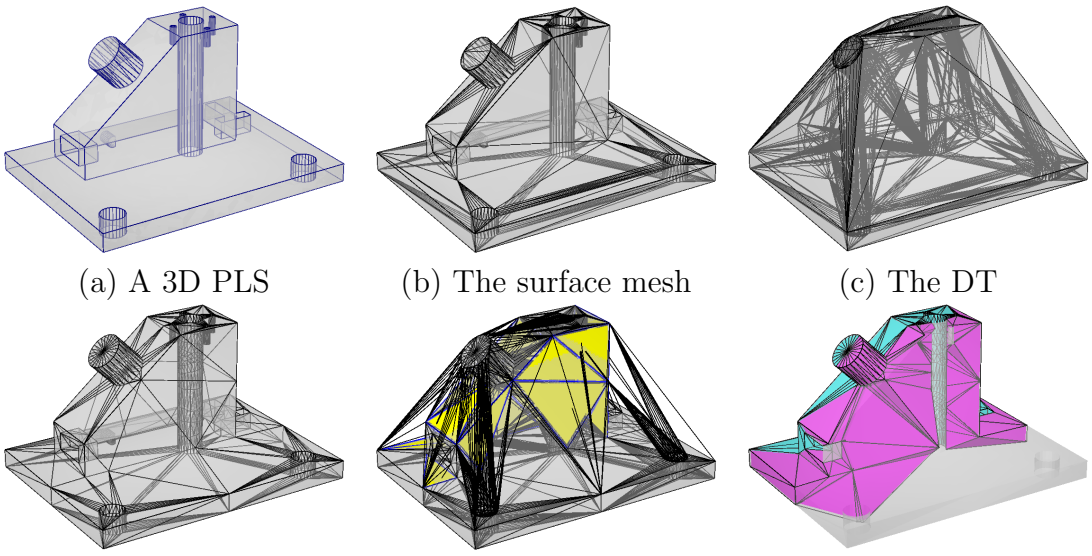
It is still an open problem to show a polynomial upper bound for the total number of Steiner points of the segment recovery algorithm.

4.5 Examples

The CDT algorithm has been implemented in the program `TetGen` [52]. In this section, we provide two examples to illustrate the practical behavior and the effectiveness of the CDT algorithm.

Fig. 16 illustrates an example of one run of the CDT algorithm on a mechanical part (`Cami1a` with the intermediate status of the different steps. The input PLS shown in *(a)* has 460 vertices, 706 segments, and 328 facets. The surface mesh shown in *(b)*, which is the input of the local degeneracy removal algorithm, contains 954 subfaces. *(c)* is the initial Delaunay tetrahedralization of the vertex set. The status after the SegmentRecovery algorithms is shown in *(d)*. The number of break points and protect points are 213 and 269, respectively. *(e)* shows the initial status of the FacetRecovery algorithm, there are in total 6446 subfaces in which 22 are missing (highlighted in yellow). The resulting CDT is shown in *(f)*. A vertical cut is made for visualizing the interior constrained Delaunay tetrahedra.

The geometry of the next example shown in Fig. 17 is the wing of an airplane placed inside a large bounding box, see *(a)*. The surface of the wing and the bounding box were triangulated by 22905 nodes and 45806 triangles. A detailed section of the



(a) A 3D PLS (b) The surface mesh (c) The DT

(d) Segment recovery (e) Facet recovery (f) A CDT

Figure 16: Example: `Cami1a`. The input PLS and the constructed CDT are shown in (a) and (f), respectively. Pictures from (b) to (e) show the intermediate states of the CDT algorithm.

surface triangulation of the wing is shown in *(b)*. To generate the CDT from the surface mesh, `TetGen` added in total 19,455 Steiner points in which 2,542 are break points and 16,913 are protect points. A view of the inside of the CDT near the wing is shown in *(c)*. In *(d)*, the modified surface mesh of the CDT is shown.

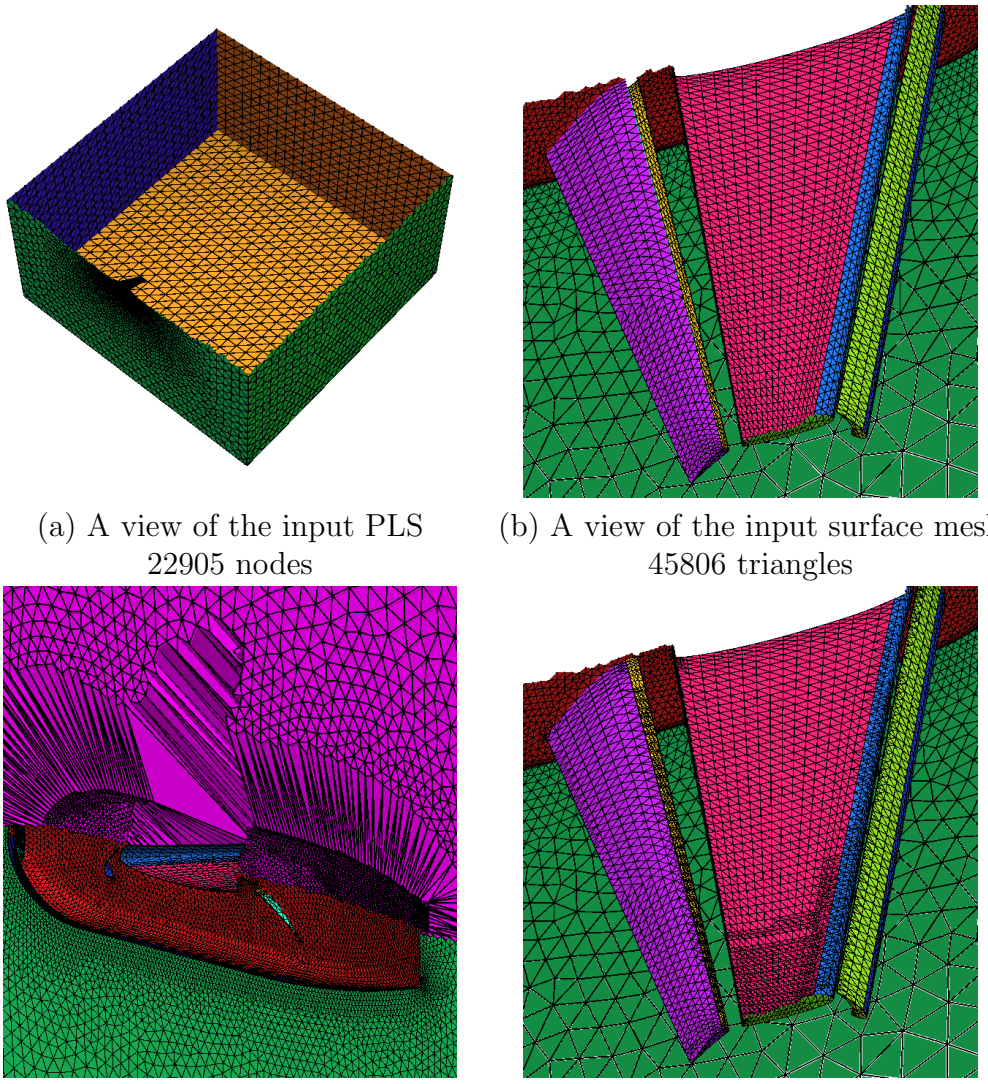
5 Constrained Delaunay Refinement

In this section, the algorithm for refining a tetrahedral mesh is presented. It behaves like the Delaunay refinement algorithm of Shewchuk [46], i.e. it finds the badly-shaped tetrahedra and eliminates them by inserting their circumcenters. However, the insertion of circumcenters is restricted by the local mesh sizing information specified on input. We refer to this algorithm as *constrained Delaunay refinement*.

5.1 Element Shape and Mesh Size

A *tetrahedron shape measure* is a continuous function which evaluates the quality of a tetrahedron by a real number. The most general shape measure for a simplex is the *aspect ratio*. The aspect ratio $\eta(\tau)$ of a tetrahedron τ is the ratio between the longest edge length and the shortest height. Aspect ratio measures the "roundness" of a tetrahedron in terms of a value between $\sqrt{2}/\sqrt{3}$ and $+\infty$. Low aspect ratio implies better shape.

Delaunay refinement algorithms use a weaker tetrahedron shape measure. The



(a) A view of the input PLS
22905 nodes

(b) A view of the input surface mesh
45806 triangles

(c) A view of the output CDT
43044 nodes
134700 tetrahedra

(d) A view of the output surface mesh
2542 break points
17597 protect points

Figure 17: Example: `Wing-Iso`. Global and local views of the input PLS are shown in (a) and (b), respectively. Two detailed views of the output CDT are shown in (c) and (d).

radius-edge ratio $\rho(\tau)$ of a tetrahedron τ is the ratio between the radius r of its circumscribed ball and the length l of its shortest edge, i.e., $\rho(\tau) = r/l$. $\rho(\tau)$ is at least $\sqrt{6}/4 \approx 0.612$, achieved by the regular tetrahedron. Most of the badly shaped tetrahedra will have a big radius-edge ratio except the sliver, which can have a minimal value $\sqrt{2}/2 \approx 0.707$.

Each of the six edges of a tetrahedron τ is surrounded by two faces. At a given edge, a *dihedral angle* between two faces is the angle between the intersection of these faces and a plane perpendicular to the edge. A dihedral angle in τ is between $0°$ and $180°$. The minimum dihedral angle $\phi_{min}(\tau)$ of τ is a tetrahedron shape measure.

Let $\mathcal{X}$ be a three-dimensional PLS. We define a *mesh sizing function* $H : |\mathcal{X}| \to \mathbb{R}$, such that for each point $\mathbf{p} \in |\mathcal{X}|$, $H(\mathbf{p})$ specifies the desired length of edges to the vertex inserted at the location $\mathbf{p}$. For example, the *local feature size* $\text{lfs}(\mathbf{p})$ at a point $\mathbf{p} \in |\mathcal{X}|$ is defined as the radius of the smallest ball centered at $\mathbf{p}$ that intersects two non-incident components of $\mathcal{X}$. lfs() defines a default distance function on $|\mathcal{X}|$ based on its boundary information. It is 1-Lipschitz, i.e., $\text{lfs}(\mathbf{p}) \leq \text{lfs}(\mathbf{q}) + |\mathbf{p} + \mathbf{q}|$, for any $\mathbf{p}, \mathbf{q} \in |\mathcal{X}|$.

The function H is *isotropic* if the edge length does not vary with respect to the directions at $\mathbf{p}$, otherwise, it is *anisotropic*. Generally, H can be represented by a 3×3 metric tensor M which defines a field of symmetric positive definite matrices. In isotropic case, for any $\mathbf{p} \in |\mathcal{X}|$, $M(\mathbf{p}) = \frac{1}{H^2(\mathbf{p})}\mathbf{I}_3$, where $\mathbf{I}_3$ is the identity matrix in $\mathbb{R}^{3\times3}$. In the scope of this work, we assume that H is isotropic. An ideal sizing function is C^∞, $\forall \mathbf{p} \in |\mathcal{X}|$. However, in most cases, H is approximated by a discrete function specified at some points in $|\mathcal{X}|$. The size on other points is obtained by means of interpolation. A background mesh can be used for this purpose.

One of our goals is to create a tetrahedral mesh $\mathcal{T}$ of $\mathcal{X}$ such that the mesh size of $\mathcal{T}$ conforms to H. We use the following criterion to measure the mesh size conformity. Let $\mathbf{p}$ be a vertex in $\mathcal{T}$. Let $S(\mathbf{p})$ and $L(\mathbf{p})$ denote the shortest and longest edge lengths at $\mathbf{p}$, respectively. We say that the size of $\mathcal{T}$ *conforms* to H if there exist two constants C_S and C_L, where $0 < C_S \leq C_L < \infty$, such that for every vertex $\mathbf{p} \in \mathcal{T}$, the following relation holds

$$C_S \leq \frac{S(\mathbf{p})}{H(\mathbf{p})} \leq \frac{L(\mathbf{p})}{H(\mathbf{p})} \leq C_L.$$

The best conformity would be the case $C_S = C_L = 1$. It is generally not possible to obtain the best conformity. One goal is to bound the ratio $\frac{C_L}{C_S}$.

5.2 The Algorithm

Starting with an initial Delaunay tetrahedralization, Shewchuk's Delaunay refinement scheme [46] uses three rules to add new points. One adds a point $\mathbf{v}$ at the circumcenter of a badly-shaped tetrahedron τ. τ will be removed after reconnecting the local mesh edges to $\mathbf{v}$ using the Delaunay criterion. The other two rules

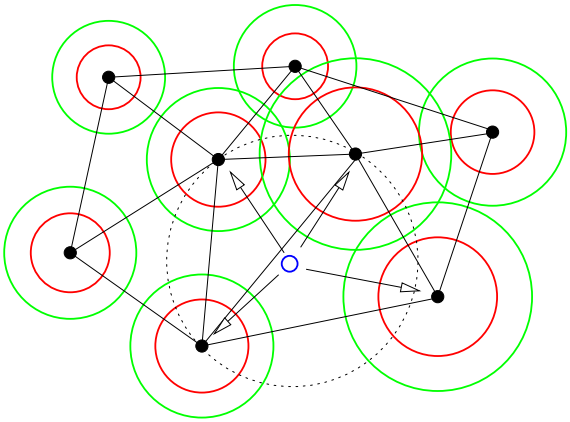


Figure 18: For each point $\mathbf{p} \in \mathcal{T}$, assume there are two virtual balls, one protect ball (shown in red), and one sparse ball (shown in green).

are used for boundary protection, i.e., split a segment or a subface by adding its circumcenter. Boundary protection has higher priority than removing badly-shaped tetrahedra.

The proposed algorithm makes use of these rules to find new points. But two variants are made: *(1)* Attempt to insert a new point at the location where the local mesh is either badly shaped or sparse, the sparseness is indicated by the sizing function at mesh vertices, and *(2)* the new point can be inserted only if it is not "close" to any existing vertex. Intuitively, one can assume that each vertex $\mathbf{p}$ of the mesh is surrounded by two virtual balls, one *sparse ball* with radius $\alpha_1 H(\mathbf{p})$, and one *protecting ball* with radius $\alpha_2 H(\mathbf{p})$ (see Fig. 18). The space outside the sparse ball of $\mathbf{p}$ is *sparse* from the viewpoint of $\mathbf{p}$, while any point inside the protecting ball of $\mathbf{p}$ is *close* to $\mathbf{p}$.

The two parameters α_1 and α_2 are used to scale the size of the balls. In particular, we get the basic Delaunay refinement algorithm by setting $\alpha_1 = \infty$ (no sparse ball), and $\alpha_2 = 0$ (no protect ball).

Point Generating Rules A candidate $\mathbf{v}$ for insertion is found by the following three *point-generating rules*. We say a segment (or a subface) is *encroached* if its diametric ball contains at least one vertex in its interior. Any tetrahedron τ be considered to be of bad quality if $\rho(\tau) > \rho_0$.

$R1$ If a subsegment σ is encroached, then $\mathbf{v}$ is the midpoint of σ.

$R2$ If a subface σ is encroached, then $\mathbf{v}$ is the circumcenter of σ. However, if $\mathbf{v}$ encroaches upon some subsegments, then reject $\mathbf{v}$. Instead, use $R1$ to return a $\mathbf{v}$ on one of the subsegments.

$R3$ If a tetrahedron τ satisfies one of the following two cases, $R3.1$ or $R3.2$:

$R3.1$ τ has a radius-edge ratio greater than ρ_0.

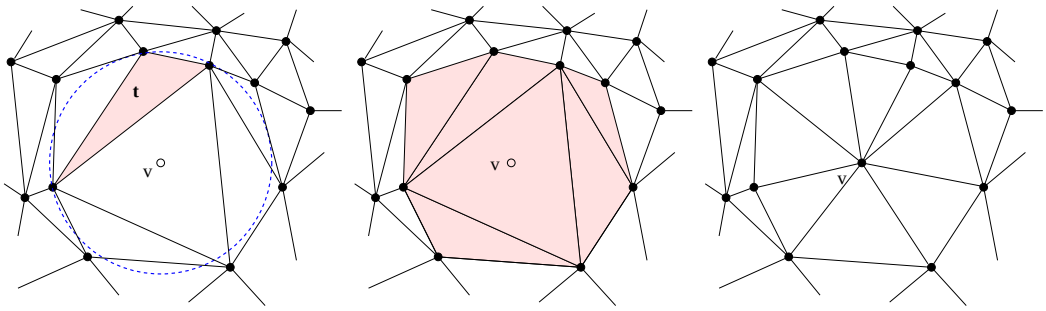


Figure 19: Left: $\mathbf{v}$ is a point found by $R2$. Middle: the vertices of the shaded region form the set P of points collected by the point accepting rule. Right: if $\mathbf{v}$ is inserted, the shaded region is re-triangulated according to the Delaunay criterion.

$R3.2$ there is a corner $\mathbf{p}$ of τ, such that $\alpha_1 H(\mathbf{p}) < r$, where r is the radius of the circumscribed ball of τ,

then $\mathbf{v}$ is inserted at the circumcenter of τ. However, if $\mathbf{v}$ encroaches upon any subsegment or subface, then reject $\mathbf{v}$. Instead, use $R1$ or $R2$ to return a $\mathbf{v}$ on one of the subfaces or subsegment.

$R3.1$ tests if τ has bad quality, and $R3.2$ checks the H-conformity of the corners of τ. $R3.1$ has a priority higher than $R3.2$. Hence $R3.2$ is triggered only if all tetrahedra have radius-edge ratio greater than ρ_0.

Point Accepting Rules Once a candidate $\mathbf{v}$ is found, $\mathbf{v}$ is not inserted immediately. Instead, the *point-accepting rule* will be called. It decides whether or not $\mathbf{v}$ can be inserted into the mesh. Let P be a set of vertices collected as follows:

- If $\mathbf{v}$ is found by $R1$, then P contains the two endpoints of the segment that $\mathbf{v}$ will split.
- If $\mathbf{v}$ is found by $R2$, then P contains the vertices of the subfaces whose diametrical circumspheres are encroached by $\mathbf{v}$ (see Fig. 19).
- If $\mathbf{v}$ is found by $R3$, then P contains the vertices of the tetrahedra whose circumspheres contain $\mathbf{v}$.

Then $\mathbf{v}$ can be inserted if $\alpha_2 H(\mathbf{p}) < \|\mathbf{v} - \mathbf{p}\|$ for all $\mathbf{p} \in P$. Otherwise, $\mathbf{v}$ is not inserted.

If $\mathbf{v}$ passes the point-accepting rule, then it is inserted into the current mesh, and the local mesh of $\mathbf{v}$ is rearranged according to the Delaunay criterion.

In the point-accepting rule, if $\mathbf{v}$ is found by $R1$ or $R2$, only the endpoints of the subsegment or subfaces of the same facet on which $\mathbf{v}$ lies have the right to accept or reject $\mathbf{v}$. $\mathbf{v}$ may be very close to some existing vertices, i.e., there may exist points

AdaptiveDelaunayRefinement ($\mathcal{T}$, ρ_0, H, α_1, α_2)
// $\mathcal{T}$ is a tetrahedral mesh of a PLS $\mathcal{X}$; ρ_0 is a radius-edge ratio bound;
// $H : |\mathcal{X}| \to \mathbb{R}^+$ is a sizing function; α_1, α_2 are two parameters.
1. initialize a queue Q of all tetrahedra of $\mathcal{T}$;
2. **while** $Q \neq \emptyset$, **do**
3. pop a tetrahedron τ from Q;
4. **if** $(\rho(\tau) > \rho_0)$ **or** $(\exists \mathbf{p} < \tau, \|\mathbf{c}_\tau - \mathbf{p}\| > \alpha_1 H(\mathbf{p}))$, **then**
5. create a vertex $\mathbf{v}$ by $R3$ (or $R1$ or $R2$);
6. **if** $\mathbf{v} \in |\mathcal{T}|$, **then**
7. collect vertices in P by the point accepting rule;
8. **if** $\forall \mathbf{p} \in P$, $\|\mathbf{v} - \mathbf{p}\| > \alpha_2 H(\mathbf{p})$, **then**
9. update $\mathcal{T}$ to be the mesh of $\mathcal{T} \cup \{\mathbf{v}\}$;
10. **if** $\mathbf{v} \neq \mathbf{c}_\tau$, **then**; $Q = Q \cup \{\tau\}$; **endif**
11. $Q = Q \cup \{\nu \in \mathcal{T} \,|\, \mathbf{v} < \nu\}$;
12. **endif**
13. **endif**
14. **endif**
15. **endwhile**
16. **return** $\mathcal{T}$;

Figure 20: The Adaptive Delaunay refinement algorithm.

$bfq \notin P$ such that $\alpha_2 H(\mathbf{q}) > \|\mathbf{v} - \mathbf{q}\|$. However, the distance $\|\mathbf{v} - \mathbf{q}\|$ is always bounded by a constant times the radius of a protecting ball.

The AdaptiveDelaunayRefinement algorithm is described in Fig. 20. It first initializes a queue Q of all tetrahedra of $\mathcal{T}$. Then it runs into a loop until Q is empty. At each step, a tetrahedron τ is removed from Q (line 3). Let $\mathbf{c}_\tau$ be its circumcenter. If τ is badly-shaped or the space at $\mathbf{c}_\tau$ is sparse (line 4), then a new point $\mathbf{v}$ is generated (line 5). Since $\mathcal{T}$ may not be a boundary conforming Delaunay mesh, $\mathbf{v}$ may lie outside $|\mathcal{T}|$, $\mathbf{v}$ can be considered for insertion if $\mathbf{v} \in |\mathcal{T}|$ (lines $6-14$). Finally, $\mathbf{v}$ can be inserted if it passes the point insertion rule (lines $8-12$). The new mesh of $\mathcal{T} \cup \{\mathbf{v}\}$ is created by the Delaunay criterion (line 9). If $\mathbf{v}$ was generated by $R1$ or $R2$, i.e., $\mathbf{v} \neq \mathbf{c}_\tau$, the insertion of $\mathbf{v}$ may not delete τ, τ is queued in Q for later process (line 10). All the newly generated tetrahedra are added to Q as well (line 11).

5.3 Analysis

The termination of this algorithm can be proved by showing that for any newly inserted vertex, the distance to its nearest mesh vertex is bounded by some positive value. Then the algorithm will stop since no arbitrarily short edge can be introduced.

Theorem 5.1 ([53]) *The algorithm terminates as long as $\alpha_2 > 0$.*

We say a segment S is *sharp* if either: *(1)* it is incident with another segment S', such that the angle between S and S' is smaller than 60°, or *(2)* it is the intersection of two facets F_1 and F_2, such that the dihedral angle between F_1 and F_2 is smaller than 69.3°. The following theorem shows that the algorithm in Fig. 20 is able to create a mesh with most of the tetrahedra having their radius-edge ratio bounded from above, while only a few poor-quality tetrahedra remain in well defined locations.

Theorem 5.2 ([53]) *Suppose H is the local feature size, $\rho_0 > 2$. There exists an $\alpha_2 > 0$, such that either output tetrahedron t has a radius-edge ratio smaller than ρ_0, or the circumcenter $\mathbf{c}_t$ of t satisfies:*

$$\|\mathbf{c}_t - \mathbf{p}\| \leq \sqrt{2}\alpha_2 H(\mathbf{p}).$$

where $\mathbf{p} \in S$ is a mesh vertex, and S is a sharp segment.

Next, we consider the mesh conformity by analyzing a special case where $H = \text{lfs}$ and $\theta_m = 90°$. The mesh quality is guaranteed with a sufficiently small α_2. Theorem 5.3 establishes bounds for these quantities for output vertices.

Theorem 5.3 ([53]) *Let $H = lfs$, $\theta_m = 90°$, $\rho_0 > 2$, and let α_2 be small enough such that all output tetrahedra have a radius-edge ratio smaller than ρ_0. Then*

(i) $S_v \geq \min\{\alpha_2, C\alpha_2 \frac{H(p(\mathbf{v}))}{H(\mathbf{v})}\}$;

(ii) $L_v \leq 2\alpha_1$.

Where $C = \sin\theta_m/\sqrt{2}$.

5.4 Sliver Removal by Delaunay Refinement

One remaining theoretical problem of our algorithm is that the resulting meshes may contain slivers which may have arbitrarily large aspect ratio but bounded radius-edge ratio. Fig. 21 highlights the remaining slivers in two bounded radius-edge ratio meshes.

Many algorithms have been proposed that take as input a good radius-edge ratio tetrahedral mesh, and refine it into a sliver-free, good aspect ratio mesh [14, 10, 24, 35]. Only the *Sliver Exudation* algorithm of Cheng *et al* [10] is implemented and experimental results are reported. Besides the theoretical work, Shewchuk [46] showed experiments that Delaunay refinement indeed is effective in removing slivers. The purpose of this section is to give more quantitative evidences, that Delaunay refinement scheme is able to generate good aspect ratio meshes if the input angle condition is satisfied.

The minimum dihedral angle, ϕ_{min}, is used to identify slivers. As suggested in [23], we fix a threshold and call a tetrahedron τ a *sliver* if its $\phi_{min}(\tau) < 5°$. Assume a bounded radius-edge ratio Delaunay mesh containing slivers to be given. Starting from that mesh, each sliver is removed by inserting a point at its circumcenter. If

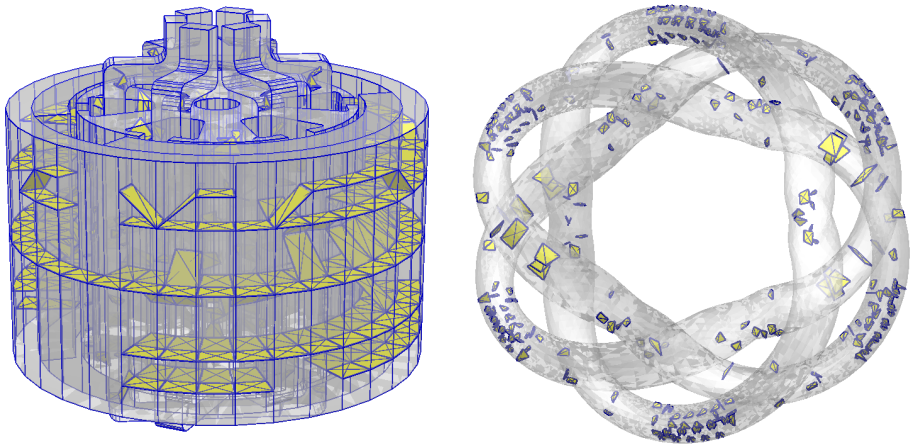


Figure 21: The remaining slivers in bounded radius edge ratio meshes are highlighted. Left: Pmdc, and Right: Hose.

a circumcenter encroaches upon any subsegment or subface, it is rejected, and the circumcenters of the encroached boundaries are inserted. A new inserted point may create new large radius-edge ratio ($> \rho_0$) tetrahedra and new slivers, they are processed by the same way. The tetrahedra with large radius-edge ratios have priority over slivers. The process repeats until neither slivers nor bad quality tetrahedra exist. This method is essentially the same as that of Li and Teng [35]. However, we omit using *picking regions* and allow the creation of *small slivers*. Hence the termination is theoretically not guaranteed.

Table 1 shows the experiments on PLS models (available from [52]). For testing purposes, these models contains no input dihedral angle smaller than 90°. Hence the basic Delaunay refinement scheme always terminates. In [35], Li and Teng claim that their algorithm needs at most $O(n)$ Steiner points to generate a sliver-free Delaunay mesh, where n is the number of input nodes of the initial bounded radius-edge ratio mesh [35]. From our experiments, we observe that the number of initial slivers and the number of Steiner points are in the same order, the latter is slightly larger and depends on the PLS models. Our tests suggest that the number of initial slivers is crucial. To find the relation between the numbers of initial slivers and Steiner points will be meaningful.

However, Delaunay refinement generally will not work for PLSs having input dihedral angle smaller than 69.3°, i.e., termination is not guaranteed. Hence, techniques like mesh smoothing and mesh optimization need to be used.

5.5 Examples

The algorithm has been integrated into the software `TetGen` [52]. The input can be either a PLS or a constrained tetrahedral mesh. A sizing function H can be

	DEMO03	Pmdc	Thepart	Hose
In Nodes	2960	12050	7954	19941
In Tets	12150	43219	35076	73720
In Min ϕ	0.007°	0.001°	0.002°	0.324°
In Max ϕ	179.98°	179.99°	179.99°	179.09°
In Max η	9435.9	71128	37316.0	194.1
PLS Facets	744	502	1995	12000
Slivers	*110*	*629*	*363*	*324*
Steiner Points	*349*	*1607*	*1075*	*691*
Out Min ϕ	5.01°	5.01°	5.02°	5.03°
Out Max ϕ	172.01°	175.60°	171.91°	171.94°
Out Max η	22.2	25.5	28.1	21.9

Table 1: Sliver removal experiments on meshes from PLS models (publicly available from [52]). The initial and final meshes have a bounded radius-edge ratio below 2. The number of slivers and the inserted Steiner points are highlighted. Initial and final mesh quality are reported by the min-max dihedral angle (ϕ) and the maximum aspect ratio (η).

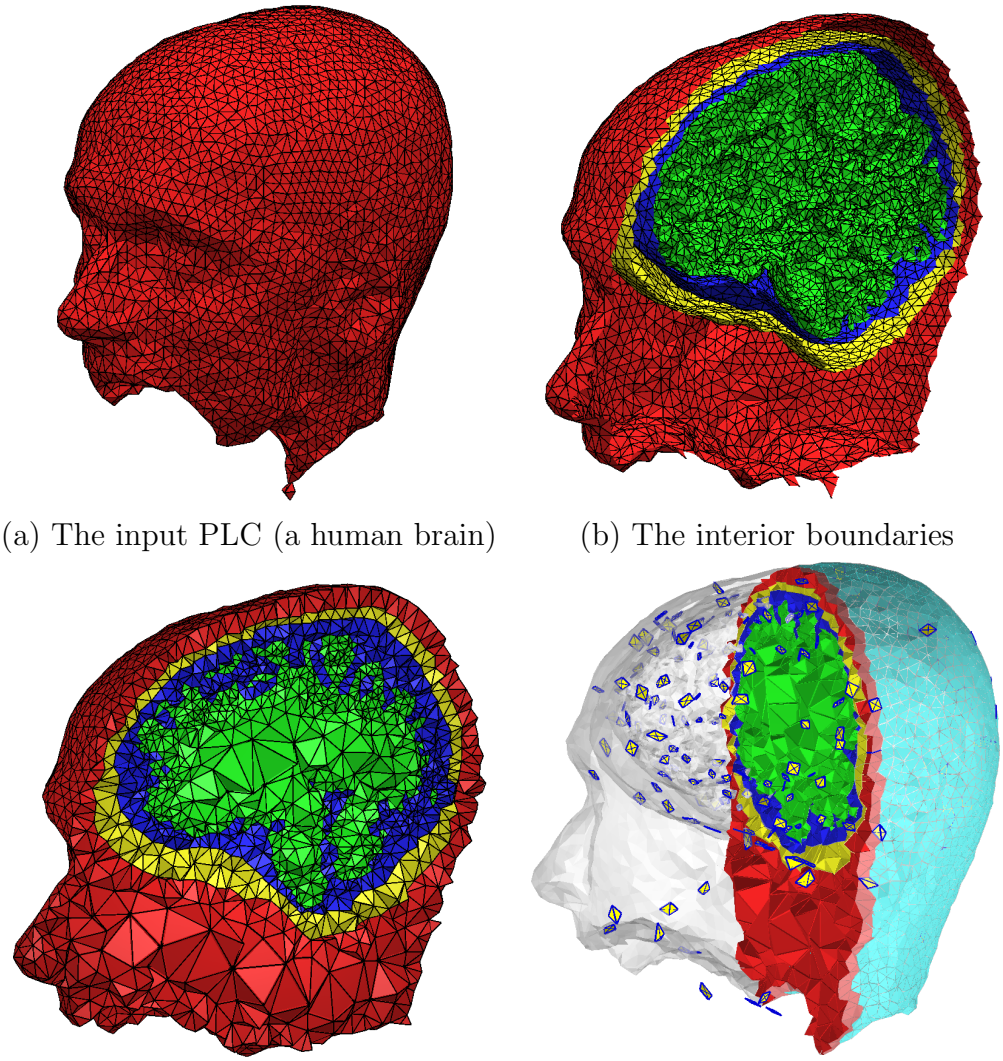
optionally specified through a background mesh. Parameters ρ_0, α_1, and α_2 are all adjustable at runtime.

The first mesh example is used in the finite element modeling of the forward problem of EEG/MEG source localization [19]. Here the mesh domain is a human head modeled by four layers, namely, the skin, the outer and inner skull, and the cortex. The PLS model of the brain contains 20301 nodes, 40638 triangles, see Fig. 22 (a). It has four sub-domains (shown in different colors) separated by the three internal layers, see Fig. 22 (b). The smallest input face angle and input dihedral angle are 10.7° and 3.8°, respectively.

The mesh refinement parameters were chosen as follows: $\rho_0 = \sqrt{2}$, $\alpha_1 = \infty$, $\alpha_2 = 1.0$. The sizing function H was automatic generated from the initial mesh. The refined mesh contains $28,543$ nodes ($6,618$ Steiner points) and $165,607$ tetrahedra. Fig. 22 (c) shows an internal view of the resulting mesh. In (d), the remaining bad quality tetrahedra (whose maximal dihedral angle are larger than 175°) are highlighted. These tetrahedra were removed after applying a mesh optimization algorithm on the mesh.

The second example is a Boeing 747 model. The input is the surface mesh of the plane ($2,874$ nodes, $5,738$ triangles) plus a bounding box. The tetrahedral mesh is constructed to compute a potential flow around the Boeing 747. The sizing function is explicitly given by a smoothed function of the Euclidian distance from the surface mesh (Fig. 23 (b)). The resulting tetrahedral mesh (Fig. 23 (c) and (d) $490,692$ nodes, $2,709,770$ tetrahedra) was generated with parameters: $\rho_0 = 2.0, \alpha_1 = 0.5, \alpha_2 = 0.25$.

Remaining bad quality tetrahedra are all close to the sharp segments of the plane's



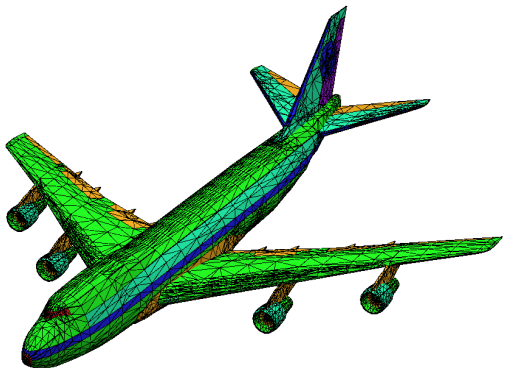
(a) The input PLC (a human brain)

(b) The interior boundaries

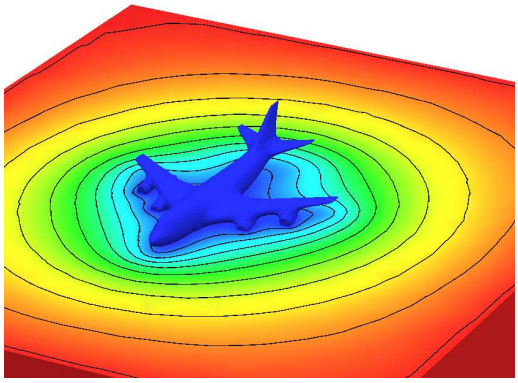
(c) Mesh detail

(d) Remaining bad tetrahedra

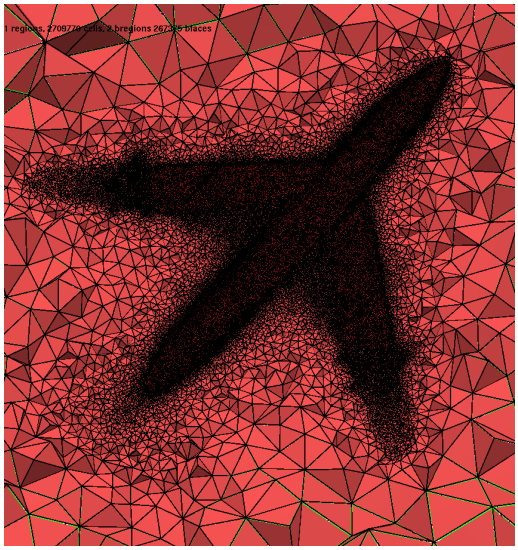
Figure 22: Tetrahedral mesh of a human brain. (Courtesy Carsten Wolters, IBB, University of Münster.)



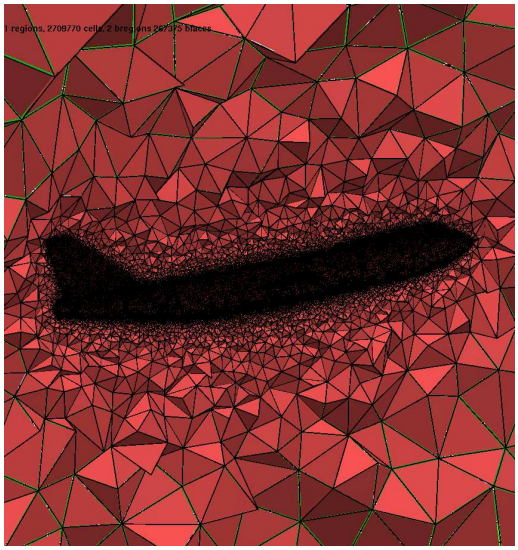
(a) The surface mesh, 2,874 nodes and 5,738 triangles.



(b) Tetrahedra size information is shown on a cut through the box, blue - small size tetrahedra, red - large size tetrahedra.



(c) Mesh detail (490,692 nodes and 2,709,770 tetrahedra)



(d) Mesh detail, refine time 59 s (44.5k tet/sec), 3.60GHz Intel PC.

Figure 23: Adaptive tetrahedral mesh of the Boeing 747 model.

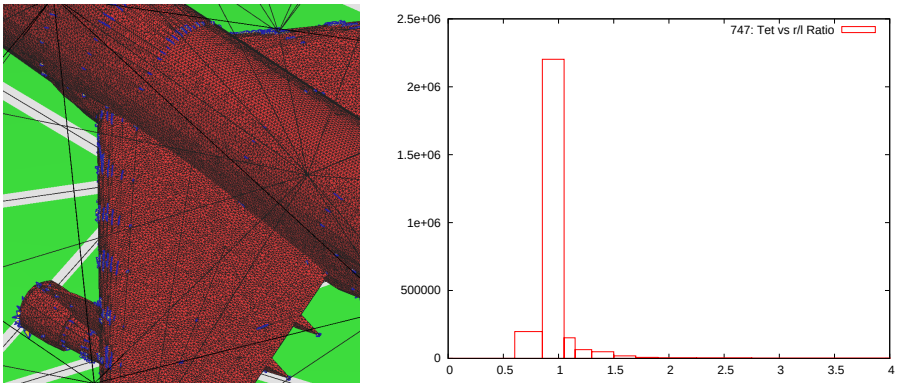


Figure 24: Mesh quality plot of the Boeing 747 model. Left: The remaining bad quality tetrahedra (shown in blue) are located close to the surface and sharp input segments. Right: The radius-edge ratio histogram.

surface (Fig. 24 left). The histogram of radius-edge ratios of the tetrahedral mesh is shown in Fig. 24 right. The mesh contains high-quality tetrahedra in the bulk of the meshed domain. For example, over 94% of the tetrahedra of the Boeing 747 mesh have radius-edge ratios between 0.612 and 1.1. Only about 0.4% of the tetrahedra are of bad-quality.

6 Open Issues

The only obstacle for obtaining an isotropic good quality Delaunay mesh is the limitation (e.g., $\geq 69.3°$) on facet angles of the input PLS [54]. It seems that the only possible way to circumvent it, is to create points at the neighborhoods of small facet angles by special constructions. Such algorithms are proposed by Cheng *et al.* [11] and Pav *et al.* [42]. However, both approaches are complicated and may introduce arbitrarily many new points. A remaining problem is how to reduce the complexity for such a special construction, so that it is efficient.

Experiments show that Delaunay refinement can remove slivers by adding the same amount of Steiner points. However, it is not proved yet. It is an open problem to determine a non-trivial lower bound (or an expected bound) on the dihedral angles of the output tetrahedra.

Another open problem is to find an upper bound on the number of Steiner points for recovering all segments of a PLS $\mathcal{X}$ in a Delaunay triangulation. So far, only Edelsbrunner *et al.* [26] provided an upper bound for the problem in two-dimensional cases. It is necessary to analyze our segment recovery algorithm further or to develop a new algorithm which has asserted bounds on the resulting size of the CDT.

A challenging problem is to remove or suppress Steiner points from the boundary (segments and facets) of a CDT. Although it is theoretically guaranteed that any Steiner point can be removed from the boundary [29], the robustness and efficiency

are the main difficulties in practice.

Acknowledgements

The author would like to thank Dr. Jürgen Fuhrmann and Dr. Klaus Gärtner for stimulating this work and helpful discussions. Special thanks to Dr. Andreas Rathsfeld for carefully reading and improving the manuscripts. Special thanks to Larry Wigton for providing the `Wing-Iso` test model.

References

[1] I. Babuška and A. K. Aziz. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, 13(2):214–226, 1976.

[2] I. Babuška, J. E. Flaherty, W. D. Henshaw, J. E. Hopcroft, J. E. Oliger, and T. Tezduyar, editors. *Modeling, mesh generation, and adaptive numerical methods for partial differential equations*, volume 75 of *The IMA Volumes in Mathematics and its Applications*. Springer, 1995.

[3] I. Babuška and W. C. Rheinboldt. Adaptive approaches and reliability estimates in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 17/18:519–540, 1987.

[4] T. Baker. Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation. *Engineering with Computers*, 5:161–175, 1989.

[5] M. W. Bern, J. E. Flaherty, and M. Luskin, editors. *Grid generation and adaptive algorithms*, volume 113 of *The IMA Volumes in Mathematics and its Applications*. Springer, 1999.

[6] P. R. Cavalcanti and U. T. Mello. Three-dimensional constrained Delaunay triangulation: a minimalist approach. In *Proc. 8th International Meshing Roundtable*, pages 119–129. Sandia National Laboratories, 1999.

[7] J. C. Cavendish, D. A. Field, and W. H. Frey. An approach to automatic three-dimensional finite element mesh generation. *International Journal for Numerical Methods in Engineering*, 21:329–347, 1985.

[8] B. Chazelle. Convex partition of a polyhedra: a lower bound and worest-case optimal algorithm. *SIAM Journal on Computing*, 13(3):488–507, 1984.

[9] B. Chazelle and L. Palios. Triangulating a nonconvex polytope. *Discrete and Computational Geometry*, 5:505–526, 1990.

[10] S.-W. Cheng, T. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng. Sliver exudation. *J. Assoc. Comput. Mach.*, 47:883–904, 2000.

[11] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray. Quality meshing for polyhedra with small angles. *International Journal on Computational Geometry and Applications*, 15:421–461, 2005.

[12] P. L. Chew. Constrained Delaunay triangulation. *Algorithmica*, 4:97–108, 1989.

[13] P. L. Chew. Guaranteed-quality triangular meshes. Technical Report TR 89-983, Department of Computer Science, Cornell University, 1989.

[14] P. L. Chew. Guaranteed-quality Delaunay meshing in 3D. In *Proc. 13th annual ACM Symposium on Computational Geometry*, pages 391–393, Nice, France, June 1997.

[15] D. Cohen-Steiner, E. C. De Verdière, and M. Yvinec. Conforming Delaunay triangulation in 3D. In *Proc. 18th annual ACM Symposium on Computational Geometry*, 2002.

[16] B. N. Delaunay. Sur la sphère vide. *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, 7:793–800, 1934.

[17] O Devillers and M. Teillaud. Perturbations and vertex removal in Delaunay and regular 3D triangulations. Technical Report 5968, INRIA, 2006.

[18] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM Journal on Numerical Analysis*, 33:1106–1124, 1996.

[19] F. Drechsler, C. H. Wolters, T. Dierkes, H. Si, and L. Grasedyck. A highly accurate full subtraction approach for dipole modeling in EEG source analysis using the finite element method. Technical Report 95, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2007.

[20] Q. Du and D. Wang. Boundary recovery for three dimensional conforming Delaunay triangulation. *Comput. Methods Appl. Mech. Engrg.*, 193:2547–2563, 2004.

[21] Q. Du and D. Wang. Constrained boundary recovery for the three dimensional Delaunay triangulations. *International Journal for Numerical Methods in Engineering*, 61:1471–1500, 2004.

[22] H. Edelsbrunner. *Geometry and topology for mesh generation.* Cambridge University Press, England, 2001.

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

[24] H. Edelsbrunner, X.-Y. Li, G. L. Miller, A. Stathopoulos, D. Talmor, S.-H. Teng, A. Üngör, and N. Walkington. Smoothing and cleaning up slivers. In *Proc. 32nd ACM Sympos. Theory Comput.*, pages 273–277, Portland, Oregon, United States, 2000.

[25] H. Edelsbrunner and M.P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithm. *ACM Transactions on Graphics*, 9(1):66–104, 1990.

[26] H. Edelsbrunner and T. S. Tan. An upper bound for conforming Delaunay triangulations. *SIAM Journal on Computing*, 22:527–551, 1993.

[27] P. J. Frey and P. L. George. *Mesh generation, application to finite elements*. Hermes Science, Oxford, United Kingdom, 2000.

[28] P. L. George, H. Borouchaki, and P. Laug. An efficient algorithm for 3D adaptive meshing. *Advances in Engineering Software*, 33:377–387, 2002.

[29] P. L. George, H. Borouchaki, and E. Saltel. últimateŕobustness in meshing an arbitrary polyhedron. *International Journal for Numerical Methods in Engineering*, 58:1061–1089, 2003.

[30] P. L. George, F. Hecht, and E. Saltel. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering*, 92:269–288, 1991.

[31] C. Hazlewood. Approximating constrained tetrahedralizations. *Computer Aided Geometric Design*, 10:67–87, 1993.

[32] C. Johnson and P. Hansbo. Adaptive finite element methods in computational mechanics. *Computer Methods in Applied Mechanics and Engineering*, 101:143–181, 1992.

[33] B. K. Karamete, M. W. Beall, and M. S. Shephard. Triangulation of arbitrary polyhedra to support automatic mesh generators. *International Journal for Numerical Methods in Engineering*, 49:167–191, 2000.

[34] D. T. Lee and A. K. Lin. Generalized Delaunay triangulations for planar graphs. *Discrete and Computational Geometry*, 1:201–217, 1986.

[35] X.-Y. Li and S.-H. Teng. Generating well-shaped Delaunay meshes in 3D. In *Proc. 12th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 28–37, Washington, DC, USA, 2001.

[36] S. H. Lo. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426, 1985.

[37] R. Löhner and P. Parikh. Three-dimensional grid generation by the advancing front method. *International Journal for Numerical Methods in Fluids*, 8:1135–1149, 1988.

[38] D. J. Mavriplis. Unstructured mesh generation and adaptivity. Technical Report 95-26, NASA Contractor Report 195069, ICASE, 1995.

[39] G. L. Miller, D. Talmor, S.-H. Teng, N. J. Walkington, and H. Wang. Control volume meshes using sphere packing: Generation, refinement and coarsening. In *Proc. 5th International Meshing Roundtable*. Sandia National Laboratories, 1996.

[40] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in higher dimensions. *SIAM Journal on Computing*, 29:1334–1370, 2000.

[41] M. Murphy, D. M. Mount, and C. W. Gable. A point-placement strategy for conforming Delaunay tetrahedralizations. In *Proc. 11th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 67–74, 2000.

[42] S. E. Pav and N. J. Walkington. Robust three dimensional Delaunay refinement. In *Proc. 13th International Meshing Roundtable*. Sandia National Laboratories, 2004.

[43] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional non-convex polyhedra. *Discrete and Computational Geometry*, 7:227–253, 1992.

[44] E. Schönhardt. Über die zerlegung von dreieckspolyedern in tetraeder. *Mathematische Annalen*, 98:309–312, 1928.

[45] J. R. Shewchuk. A condition guaranteeing the existence of higher-dimensional constrained Delaunay triangulations. In *Proc. 14th Annual Symposium on Computational Geometry*, pages 76–85, June 1998.

[46] J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *Proc. 14th Annual Symposium on Computational Geometry*, pages 86–95, Minneapolis, Minnesota, United States, June 1998.

[47] J. R. Shewchuk. Sweep algorithm for constructing higher-dimensional constrained Delaunay triangulations. In *Proc. 16th Annual Symposium on Computational Geometry*, pages 350–359, June 2000.

[48] J. R. Shewchuk. Constrained Delaunay tetrahedralization and provably good boundary recovery. In *Proc. 11th International Meshing Roundtable*, pages 193–204. Sandia National Laboratories, September 2002.

[49] J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *Proc. 11th International Meshing Roundtable*, pages 115–126, Ithaca, New York, September 2002. Sandia National Laboratories.

[50] J. R. Shewchuk. Updating and constructing constrained Delaunay and constrained regular triangulations by flips. In *Proc. 19th Annual Symposium on Computational Geometry*, pages 181–190, June 2003.

[51] J. R. Shewchuk. General-dimensional constrained Delaunay and constrained regular triangulations I: Combinatorial properties. To appear in Discrete and Computational Geometry, 2007.

[52] H. Si. TetGen. `http://tetgen.berlios.de`, 2007.

[53] H. Si. Adaptive tetrahedral mesh generation by constrained delaunay refinement. *International Journal for Numerical Methods in Engineering*, 75(7):856–880, 2008.

[54] H. Si. *Three dimensional boundary conforming Delaunay mesh generation*. PhD thesis, Faculty II - Mathematics and Natural Sciences, Technische Universität Berlin, 2008.

[55] H. Si and K. Gärtner. Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In *Proc. 14th International Meshing Roundtable*, pages 147–163, San Diego, CA, USA, 2005. Sandia National Laboratories.

[56] The Boeing Company. GGNS, the General Geometry Navier-Stokes solver. not published, 2007.

[57] R. Verfürth. *A Review of Posteriori error estimation and adaptive mesh refinement techniques*. Wiley-Teubner, 1996.

[58] N. P. Weatherill and O. Hassan. Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering*, 37:2005–2039, 1994.

[59] J. Wright and A Jack. Aspects of three-dimensional constrained Delaunay meshing. *International Journal for Numerical Methods in Engineering*, 37:1841–1861, 1994.

[60] M Yerry and M. S. Shephard. Automatic three-dimensional mesh generation by the modified-octree technique. *International Journal for Numerical Methods in Engineering*, 20:1965–1990, 1984.

[61] G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995.