Finite Elements in Analysis and Design 46 (2010) 33–46

Contents lists available at ScienceDirect

Finite Elements in Analysis and Design

journal homepage: www.elsevier.com/locate/finel

Constrained Delaunay tetrahedral mesh generation and refinement

Hang Si

Weierstrass Institute for Applied Analysis and Stochastics (WIAS), Berlin, Germany

ARTICLE INFO

Available online 22 July 2009

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 in solving these problems.

1.1. Boundary conformity

Let $\mathscr{F}$ 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 $\mathscr{T}$ conforming to $\mathscr{F}$, i.e., $\mathscr{F}$ is represented by a union of elements of $\mathscr{T}$. Additional points (so-called *Steiner points*) are allowed in $\mathscr{T}$, but the number of Steiner points should be limited as small as possible. This problem is fundamental to many applications. For example, many mesh generation methods [4,55,54,24,49] make use of such tetrahedral meshes as the intermediate meshes to obtain good quality meshes suitable for numerical simulations.

In three dimensions, this problem faces many difficulties. There are simple polyhedra which may not be tetrahedralized without Steiner points [40]. The problem of determining whether or not a non-convex polyhedron can be tetrahedralized without Steiner points is NP-complete [39]. 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, see e.g., [26,27,54,6,29,25,18,19]. A common feature of these methods is: at first, an initial Delaunay tetrahedralization of the vertex set of a polyhedron P is constructed. Next, the boundary of P will be recovered by modifying the tetrahedralization (e.g., edge/face swaps), Steiner points are added when they are needed. These methods are efficient in solving many engineering problems. However, they are not designed for arbitrary inputs. The number of Steiner points may be large for some pathological cases.

Chazelle and Palios [9] showed that any simple polyhedron (which contains no holes) P of n vertices can be decomposed into $O(n+r^2)$ tetrahedra, where r is the number of reflex edges (a quantitative measure of non-convexity) of P. However, their algorithm is neither for general non-convex polyhedra nor practical.

A theoretical approach utilizing the properties of Delaunay tetrahedralizations is to enrich the vertex set V of a polyhedron P by adding Steiner points into the boundary of P until the Delaunay tetrahedralization for the augmented vertex set V respects the boundary of P [38,36,14]. The result is a so-called *conforming Delaunay tetrahedralization*. This approach, however, may result in an unnecessarily large number of Steiner points, especially when the boundary of the polyhedron contains small angles.

E-mail address: si@wias-berlin.de

0168-874X/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.finel.2009.06.017

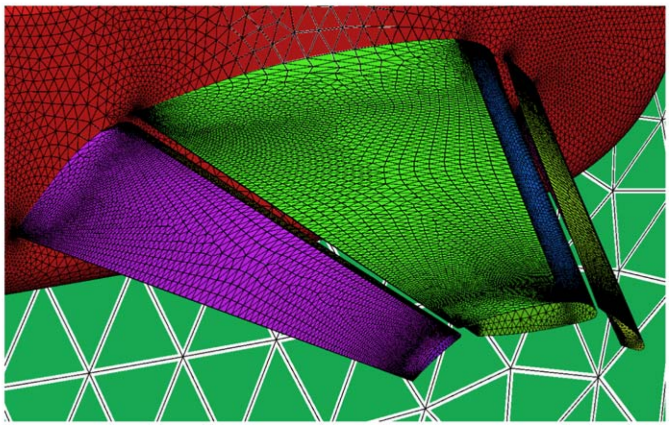
Fig. 1. A surface mesh of an airfoil.

An alternative approach for boundary conformity is to construct a *constrained Delaunay tetrahedralization* [44,51], which is a tetrahedralization for a 3D point set that respects to a set of constraints (edges and faces). A constrained Delaunay tetrahedralization contains globally non-Delaunay tetrahedra. But they are locally Delaunay, and it has properties close to those of a Delaunay tetrahedralization. This approach has both theoretical and practical features. It needs much less Steiner points than the conforming Delaunay tetrahedralization approach, especially when the boundary contains small angles. Its complexity can be theoretically analyzed.

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 $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ are *invisible* to each other if the line segment $\mathbf{xy}$ 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 $S \backslash V(P)$. A formal definition of a CDT is given in Section 3.

Remark. In the literatures, e.g., [23], a "constrained tetrahedralization" for a surface mesh $\mathcal{F}$ is defined as a tetrahedralization $\mathcal{T}$ such that all simplices of $\mathcal{F}$ are also simplices of $\mathcal{T}$. This means no Steiner points are added on $\mathcal{F}$ but possibly added in the interior of the domain Ω. In our definition of a CDT, Steiner points are allowed both on $\mathcal{F}$ and inside Ω. In this sense, one may also call it a "semi-constrained" tetrahedralization.

In general, there are infinitely many CDTs of P (by different choices of Steiner points). We are interested 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 [41] showed that if all edges of P are strongly Delaunay, then P has a 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 [44,51], 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 [43,44,46]. Among them, the flip-based facet insertion algorithm [46] has good performance. Si and Gärtner [51] 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 tetrahedra having small or large dihedral angles are bad [1,45], see Fig. 15 for examples. A notable problem is that tetrahedral meshes may contain one type of badly shaped tetrahedra, the so-called *slivers* [7], which are tetrahedra whose volumes are close to zero, see Fig. 15 right. Slivers can have both very large (near 180°) and very small (close to 0°) dihedral angles which may cause big numerical errors. It is a challenging problem for generating quality Delaunay meshes without slivers [42,37,10].

In general the nature of the exact solution of a given problem is not known beforehand. 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., [3,28,53,2,5]. Fig. 2 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 [28]. 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 [17].

Mesh adaptation, i.e., 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 [23] and [33]. In this work, we focus on how to efficiently perform step (2) in the above process.

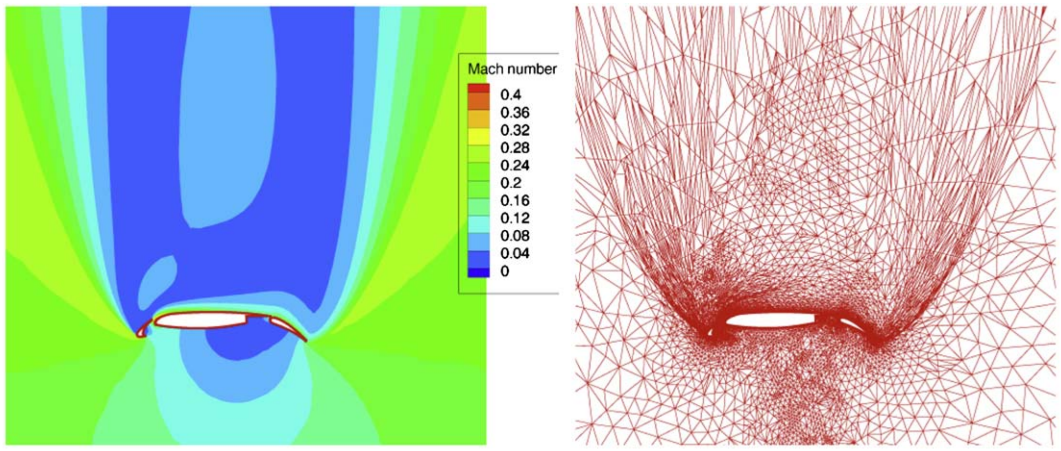

Fig. 2. Adaptive numerical simulation of the lift versus angle-of-attack for a multi-element airfoil (GGNS [52]). Left: Mach distribution at 90° angle-of-attack. Right: an intermediate adapted mesh.

Problem 1.2. Given an initial tetrahedral mesh $\mathscr{T}$ of a three-dimensional domain Ω and an isotropic mesh sizing function H defined on Ω, insert new points into $\mathscr{T}$ to form a good quality tetrahedral mesh $\mathscr{T}'$ of Ω such that the mesh size of $\mathscr{T}'$ 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 [31,32], Octree-based methods [56,35], Delaunay-based methods [34,13], and the combinations of them [54,24]. Among these methods, the Delaunay-based methods which utilize the Delaunay criterion and the Delaunay triangulation [15] are the most robust and efficient. In Section 5 we present a point insertion algorithm based on the Delaunay refinement technique.

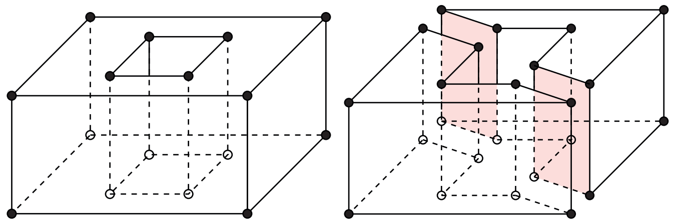

Fig. 3. Polyhedra and faces. Left: a three-dimensional polyhedron (a torus) formed by the union of four convex polytopes. It consists of 16 vertices (zero-faces), 24 edges (1-faces), 10 two-faces (the faces at top and bottom are not simply connected), and 1 three-face (which is the object). Right: two 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.

1.3. Outline

The rest of this article is organized as follows. Section 2 introduces an object called piecewise linear system which is used as an approximation of a mesh domain. A 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. The practicality of the presented algorithms is demonstrated through several examples in Section 6. We end this article by a list of open questions in Section 7.

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 subdomains 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. [34] 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 [57]. We define a general and therefore not necessarily convex *polyhedron* P as the union of a finite set $\mathscr{P}$ of convex polyhedra, i.e., $P = \bigcup_{U\in\mathscr{P}} U$, and the space of P is connected. The *dimension* $\dim(P)$ is the largest dimension of a convex polyhedron in $\mathscr{P}$. Note that P may contain holes in its interior. We require that the *underlying space* (the union of polyhedra in $\mathscr{P}$) of P must be connected. See Fig. 3 for examples.

We follow the definition of faces of a polyhedron by Edelsbrunner [20] 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 $N_\varepsilon(\mathbf{x}) = (\mathbf{x} + \mathbb{B}(\mathbf{0},\varepsilon)) \cap P$. The *face figure* of $\mathbf{x}$ is the enlarged version of this neighborhood within P, i.e., $\mathbf{x}+\bigcup_{\lambda>0}\lambda(N_\varepsilon(\mathbf{x})-\mathbf{x})$. A *face* F of a polyhedron P is the closure of a maximal connected set of points with identical face figures. See Fig. 3 for examples.

A face F of P is again a polyhedron. Particularly, $\emptyset$ is a face of P. If all convex polyhedra in $\mathscr{P}$ 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 $\mathscr{X}$ of polyhedra with the following properties:

(i) $P \in \mathscr{X} \Longrightarrow$ all faces of P are in $\mathscr{X}$,
(ii) $P, Q \in \mathscr{X} \Longrightarrow \exists K \in \mathscr{X}$, s.t. $P \cap Q = \bigcup_{K\in\mathscr{X}} K$, and
(iii) $\dim(P\cap Q) = \dim(P), P \neq Q \Longrightarrow P \subseteq Q$ and $\dim(P) < \dim(Q)$.

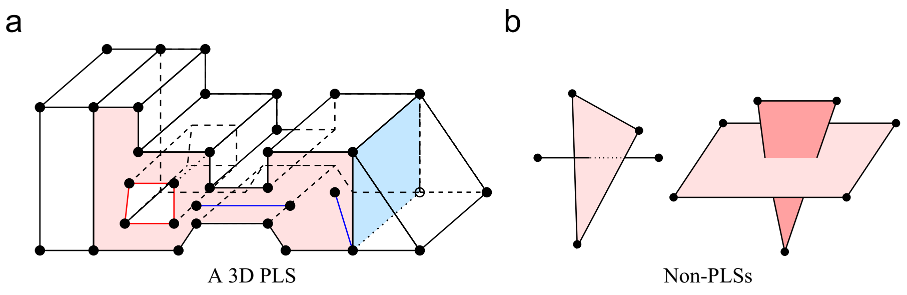

Fig. 4. (a) A three-dimensional PLS $\mathscr{X}$. The pink area highlights a facet in $\mathscr{X}$, which is a non-convex polygon with a hole in its interior. Moreover, this face contains two floated segments in $\mathscr{X}$ (shown in blue) in its interior. The light blue area shows an internal facet in $\mathscr{X}$. (b) Two non-PLC objects. They are not closed under intersections. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

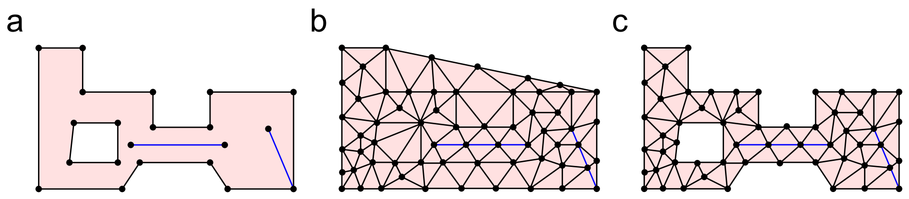

Fig. 5. Triangulation and mesh. (a) A 2D PLS $\mathscr{X}$, (b) a triangulation of $\mathscr{X}$ and (c) a mesh of $\mathscr{X}$.

This definition generalizes the one introduced by Miller et al. [34] by allowing non-convex polyhedra. PLSs are flexible for representing non-manifold objects. Properties (i) and (ii) are essential, they ensure that a PLS is closed by both taking boundaries and taking intersections. In particular, property (ii) is relaxed from that 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 $\mathscr{X}$. (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. 4 for examples.

The dimension of a PLS $\mathscr{X}$, denoted as $\dim(\mathscr{X})$, is the largest dimension of its polyhedra. A *subsystem* of $\mathscr{X}$ is a subset of $\mathscr{X}$ which is again a PLS. A particular subsystem is the *i-skeleton*, $\mathscr{X}^{(i)}$, of $\mathscr{X}$ which consists of all polyhedra of $\mathscr{X}$ whose dimensions $\leq i$. For example, $\mathscr{X}^{(0)}$ is the *vertex set*, denoted as $\mathrm{vert}(\mathscr{X})$, of $\mathscr{X}$. The *boundary system*, denoted as $\partial\mathscr{X}$, of $\mathscr{X}$ is the $(\dim(\mathscr{X})-1)$-skeleton of $\mathscr{X}$. The *underlying space* of $\mathscr{X}$ is $|\mathscr{X}| = \bigcup_{P\in\mathscr{X}} P$. Note that $|\mathscr{X}| \subseteq \mathbb{R}^d$ is a topological subspace of $\mathbb{R}^d$. The collection $\mathscr{X}$ gives a special topology on $|\mathscr{X}|$, refer to [50].

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

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

(i) $|\mathscr{T}| = \mathrm{conv}(\mathrm{vert}(\mathscr{X}))$ and
(ii) $\forall P \in \mathscr{X} \Longrightarrow \exists \mathscr{K} \subseteq \mathscr{T}$ such that $|\mathscr{K}| = P$.

Note that $\mathscr{T}$ may contain Steiner points. We define a *mesh* of $\mathscr{X}$ to be a subcomplex $\mathscr{K}$ of $\mathscr{T}$ such that $|\mathscr{K}| = |\mathscr{X}|$. 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. 5 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 [30] and Chew [12] for generating two-dimensional Delaunay-like triangulations from planar straight line graphs (a one-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 $P \in \mathscr{X}$ 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 $P \in \mathscr{X}$ at a single point. Otherwise $\mathbf{x}$ and $\mathbf{y}$ are *visible* to each other. See Fig. 6 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 $\mathscr{X}$ be a PLS in $\mathbb{R}^3$ with $\mathrm{vert}(\mathscr{X}) \subseteq S$. 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 $F \in \mathscr{X}$, such that $\mathrm{int}(\sigma) \subseteq \mathrm{int}(F)$. Let $K = S \cap \mathrm{aff}(F)$, then no vertex of K contained in the interior of B_σ is visible from any point in $\mathrm{int}(\sigma)$.

Case (i) means that every Delaunay simplex is also constrained Delaunay. In (ii), F is the lowest-dimensional polyhedron of $\mathscr{X}$ 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. 6 right for examples.

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

By this definition, a CDT of $\mathscr{X}$ may contain Steiner points, i.e., points in $S\backslash\mathrm{vert}(\mathscr{X})$. It is called a *pure* CDT if it does not contain

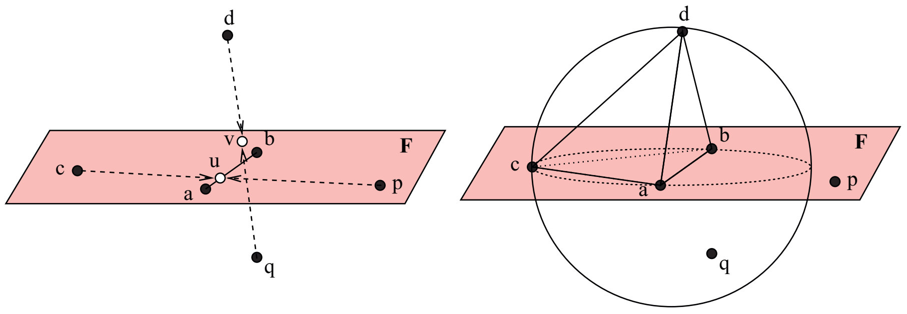

Fig. 6. Visibility and constrained Delaunay criterion. The shaded region is a facet F of a PLS $\mathscr{X}$ in $\mathbb{R}^3$, $\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{p} \in F$. **ab** is a segment of $\mathscr{X}$. Left: **d** and **q** are invisible to each other since $\mathbf{dq} \cap F = \mathbf{v}$. **c** and **p** are invisible to each other since $\mathbf{cp} \cap \mathbf{ab} = \mathbf{u}$. **u** sees both **c** and **p**. Right: a circumball of the tetrahedron **abcd** contains **q**. **abcd** is constrained Delaunay since **q** is not visible from its interior. The triangle $\mathbf{abc} \subset F$ is constrained Delaunay since **p** is outside its diametric ball.

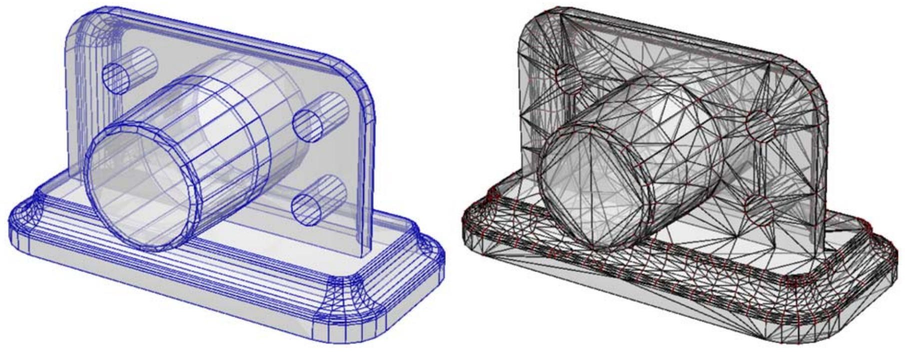

Fig. 7. A 3D PLS $\mathscr{X}$ (left) and a modified PLS $\mathscr{X}'$ (right). $\mathscr{X}'$ contains Steiner points on the boundary of $\mathscr{X}$. The underlying spaces of $\mathscr{X}$ and $\mathscr{X}'$ are the same.

Steiner points. A two-dimensional pure CDT is the same as the one defined by Lee and Lin [30] and Chew [11]. Shewchuk's definition of a CDT [47] is also a pure CDT. It is well known that a pure CDT of a three-dimensional PLS may not exist while there are infinitely many CDTs of $\mathscr{X}$ 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 $\mathscr{X}$ is very close to a conforming Delaunay triangulation of $\mathscr{X}$. The proofs are omitted, they are found in [50].

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

Theorem 3.1 (*Constrained Delaunay Lemma, Si [50]*). *If every* $(d-1)$-*simplex of* $\mathscr{T}$ *is locally Delaunay, then* $\mathscr{T}$ *is a CDT of* $\mathscr{X}$.

If a point set S in $\mathbb{R}^3$ is in general position, i.e., no five 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* $\mathscr{T}$ *be a CDT of* $\mathscr{X}$. *If vert*$(\mathscr{T})$ *is in general position, then* $\mathscr{T}$ *is the unique CDT of* $\mathscr{X}$ *with the set of vertices of* $\mathscr{T}$.

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

Theorem 3.3 (*Si [50]*). *Let* $\mathscr{X}$ *be a d-dimensional PLS.*

(i) *A CDT of* $\mathscr{X}^{(2)}$ *is a CDT of* $\mathscr{X}$.
(ii) *A CDT of* $\mathscr{X}^{(i)}$ *is a conforming Delaunay triangulation of* $\mathscr{X}^{(i)}$, *where* $i = 0, 1$.

4. The CDT algorithm

In this section, we describe an algorithm to construct a CDT from any PLS. The basic idea of this algorithm is to modify the input PLS $\mathscr{X}$ by adding few Steiner points into the boundary of $\mathscr{X}$, such that the modified PLS $\mathscr{X}'$ has the same underlying space and the same topology as $\mathscr{X}$. Moreover, the existence of a pure CDT of $\mathscr{X}'$ can be guaranteed. Hence we can construct it without using Steiner points. The result is a Steiner CDT of $\mathscr{X}$. Fig. 7 illustrates this idea.

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

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

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

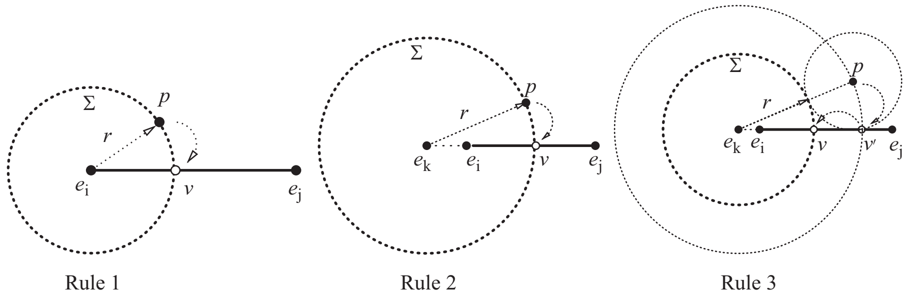

Fig. 8. Segment splitting rules.

4.1. Segment recovery

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

We need some definitions. A vertex of $\mathscr{X}$ is *acute* if at least two segments of $\mathscr{X}$ incident at it form an angle less than 60°. We distinguish two types of segments in $\mathscr{X}$: 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 into it.

Let $\mathbf{e}_i\mathbf{e}_j$ be a segment of $\mathscr{X}$ 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 even 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 $\mathscr{X}$ is missing in $\mathscr{D}_0$ and vert($\mathscr{D}_0$) is in general position, then it must be encroached by at least one vertex of $\mathscr{D}_0$.

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. 8 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. 8 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. 8 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 Rules 1 and 2 never create an edge shorter than the distance $\|R(\mathbf{e}_i) - \mathbf{v}\|$. 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 Fig. 11 (8). 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. 9. The inputs are a three-dimensional PLS $\mathscr{X}$ and a Delaunay tetrahedralization $\mathscr{D}_0$ of vert($\mathscr{X}$). Some segments of $\mathscr{X}$ may be missing in $\mathscr{D}_0$. The algorithm first initializes a queue Q of all segments of $\mathscr{X}$. 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 $\mathscr{D}_1$, a Steiner point $\mathbf{v}$ inside $\mathbf{e}_i\mathbf{e}_j$ is generated by one of the three rules described in previous section (Fig. 9, 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

```
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 $\mathbf{v} \in \mathrm{int}(\mathbf{e}_i\mathbf{e}_j)$ by Rule $i$, $i \in \{1,2,3\}$;
7.           $Q = Q \cup \{\mathbf{e}_i\mathbf{v}, \mathbf{e}_j\mathbf{v}\}$;
8.           $Q = Q \cup \{\sigma \in \mathcal{X}^{(1)}, \sigma \in \mathcal{D}_1 \mid \mathbf{v} \in \mathrm{int}(B_\sigma)\}$;
9.           update $\mathcal{D}_1$ to be the DT of vert($\mathcal{D}_1$) $\cup \{\mathbf{v}\}$;
10.      endif
11.  endwhile
12.  return $\mathcal{D}_1$;
```

Fig. 9. The segment recovery algorithm. The B_σ (in line 8) means the diametric circumball of the segment σ.

```
FacetRecovery ($\mathscr{X}$, $\mathscr{D}_1$)
// $\mathscr{X}$ is a three-dimensional PLS; $\mathscr{D}_1$ is the CDT of $\mathscr{X}^{(1)}$.
1.   $\mathscr{D}_2 = \mathscr{D}_1$;
2.   Initialize a queue $Q$ of all subfaces of $\mathscr{X}$;
3.   while $Q \neq \emptyset$ do
4.     get a subface $\sigma \in Q$; $Q = Q \setminus \sigma$ ;
5.     if $\sigma$ is missing in $\mathscr{D}_2$, then
6.       form a missing region $\Omega$ containing $\sigma$;
7.       form two cavities $C_1$, $C_2$ from $\Omega$;
8.       for each $C_i, i = 1, 2$ do
9.         TetrahedralizeCavity($\mathscr{D}_2$, $C_i$, $Q$);
10.      endfor
11.    endif
12.  endwhile
13.  return $\mathscr{D}_2$;
```

Fig. 10. The FacetRecovery algorithm.

in Q (line 7). Moreover, the insertion of $\mathbf{v}$ may cause other existing segments (subsegments) of $\mathscr{X}$ missing in $\mathscr{D}_1$, they are queued in Q and will be recovered later (line 8). Then $\mathscr{D}_1$ is updated to a Delaunay tetrahedralization of the vertex set including $\mathbf{v}$ (line 9). Fig. 11 illustrates a run of this algorithm on a two-dimensional PLS.

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 [50] for the details.

4.2. Facet recovery

The input of the facet recovery algorithm is a CDT $\mathscr{D}_1$ of $\mathscr{X}^{(1)}$. Some facets of $\mathscr{X}$ may not be represented by $\mathscr{D}_1$. Assumption 4.2, i.e., the vertex set of $\mathscr{D}_1$ is in general position, is important. It guarantees that the facets of $\mathscr{X}$ can be recovered without using Steiner points.

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

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

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

(i) the edges on the boundary of Ω are edges of $\mathscr{D}_2$ and
(ii) the edges in the interior of Ω are missing in $\mathscr{D}_2$.

Hence Ω is a connected set of missing subfaces. It may not be simply connected. Each missing subface belongs to a missing region. A facet can have more than one missing region.

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

The next step is to tetrahedralize each cavity C without using Steiner points. The TetrahedralizeCavity algorithm has two phases: (1) cavity verification (lines 2–13) and (2) cavity tetrahedralization (lines 15–20).

In phase (1), each face $\sigma \subset \mathrm{bd}(C)$ is verified. If σ is not Delaunay with respect to the vertex set of C, then C is enlarged by including the tetrahedron $\tau \in \mathscr{D}_2$, $\sigma < \tau$ and τ is removed from $\mathscr{D}_2$ (line 7). bd(C) is also changed, hence the three faces of τ are added into Q_C for later processing (line 8). If σ is a subface of $\mathscr{X}$, the enlargement of C will cause σ missing from $\mathscr{D}_2$. For this reason, we have to queue σ in Q (lines 9–11) so it will be recovered later.

Phase (2) first constructs the Delaunay tetrahedralization $\mathscr{D}_C$ of vert(C) (line 16). By Assumption 4.2, vert(C) is in general position, hence $\mathscr{D}_C$ will include all faces contained in bd(C). In other words, the interior of C is filled by a subset of tetrahedra of $\mathscr{D}_C$. The next step is to remove those tetrahedra which are not in int(C) from $\mathscr{D}_C$ (lines 16–20).

```
TetrahedralizeCavity ($\mathscr{D}_2, C, Q$)
// $\mathscr{D}_2$ is a tetrahedralization; $C$ is a cavity; $Q$ is a queue.
1.      // Phase (1): cavity verification.
2.      initialize a queue $Q_C$ of all faces contained in bd($C$);
3.      while $Q_C \neq \emptyset$ do
4.        get a face $\sigma \in Q_C$; $Q_C = Q_C \setminus \{\sigma\}$;
5.        if $\sigma \subset \mathrm{bd}(C)$ and $\sigma$ is non-Delaunay wrt. vert($C$), then
6.          get the tetrahedron $\tau \in \mathscr{D}_2$ such that $\sigma < \tau$;
7.          $C = C \cup \tau$, $\mathscr{D}_2 = \mathscr{D}_2 \setminus \{\tau\}$;
8.          $Q_C = Q_C \cup \{v < \tau | v \neq \sigma\}$;
9.          if $\sigma$ is a subface of $\mathscr{X}$, then
10.           $Q = Q \cup \{\sigma\}$;
11.         endif
12.       endif
13.     endwhile
14.     // Phase (2): cavity tetrahedralization.
15.     form the Delaunay tetrahedralization $\mathscr{D}_C$ of vert($C$);
16.     for each tetrahedron $\tau \in \mathscr{D}_C$, do
17.       if $\tau \not\subseteq \mathrm{int}(C)$, then
18.         $\mathscr{D}_C = \mathscr{D}_C \setminus \{\tau\}$;
19.       endif
20.     endfor
21.     return $\mathscr{D}_2 = \mathscr{D}_2 \cup \mathscr{D}_C$;
```

The FacetRecovery algorithm is given in Fig. 10. Fig. 13 illustrates the recovery of one missing region of this algorithm.

4.3. Correctness

In this algorithm, Steiner points are only introduced in 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($\mathscr{D}_1$) is in general position, i.e., no five vertices of vert($\mathscr{D}_1$) share a common sphere.

Although this assumption is very strong, it is easy to be satisfied by applying the techniques of symbolic perturbations [21,44,16] in

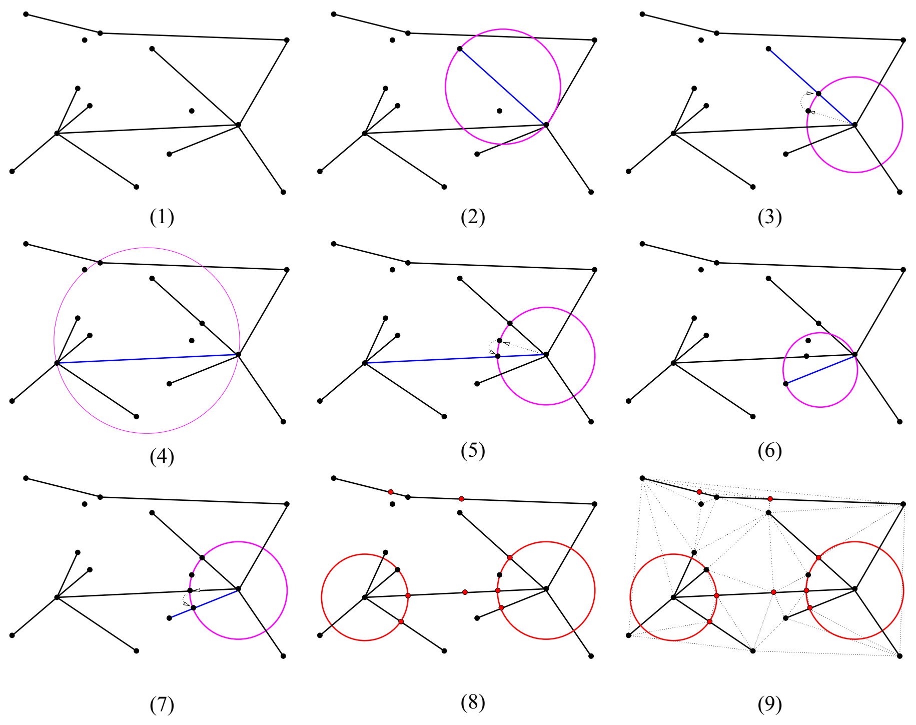

Fig. 11. A possible run of the segment recovery algorithm on the input of (1) is shown. In (2)–(7) segments in blue are split one by one by Rule 2. (8) shows the automatically formed protecting balls at two acute vertices. The Delaunay triangulation containing all subsegments is shown in (9). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

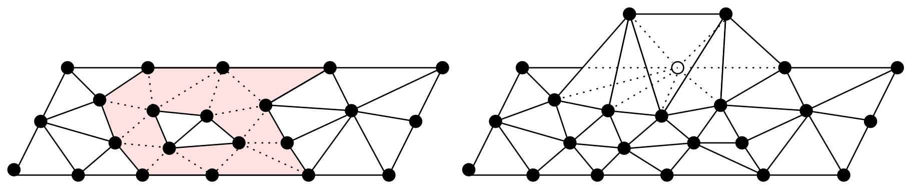

Fig. 12. Left: the shaded area highlights a missing region Ω. Right: one of the cavities resulting from a missing region is illustrated.

both steps 1 and 2. Hence (theoretically), there is no need to actually perturb the vertices.

The following theorem proved by Shewchuk [41] 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 (*Shewchuk [41]*). *If every segment of a PLS $\mathscr{Y}$ is strongly Delaunay in vert($\mathscr{Y}$), then $\mathscr{Y}$ has a CDT with no Steiner points.*

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

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. We refer the full proofs and analysis in [50].

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.

Theorem 4.4 (*Si [50]*). *Let $\mathscr{X}$ be a three-dimensional PLS which has v vertices and f facets. The* FacetRecovery *algorithm runs in time $O(fv^2 \log v)$.*

The proof that the complexity estimate is sharp we have to construct is completed by constructing a PLS which needs such running time, then showing that it is indeed the worst case. Such an example is shown in Fig. 14. In this example $\mathscr{L}_{100,5}$, 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.

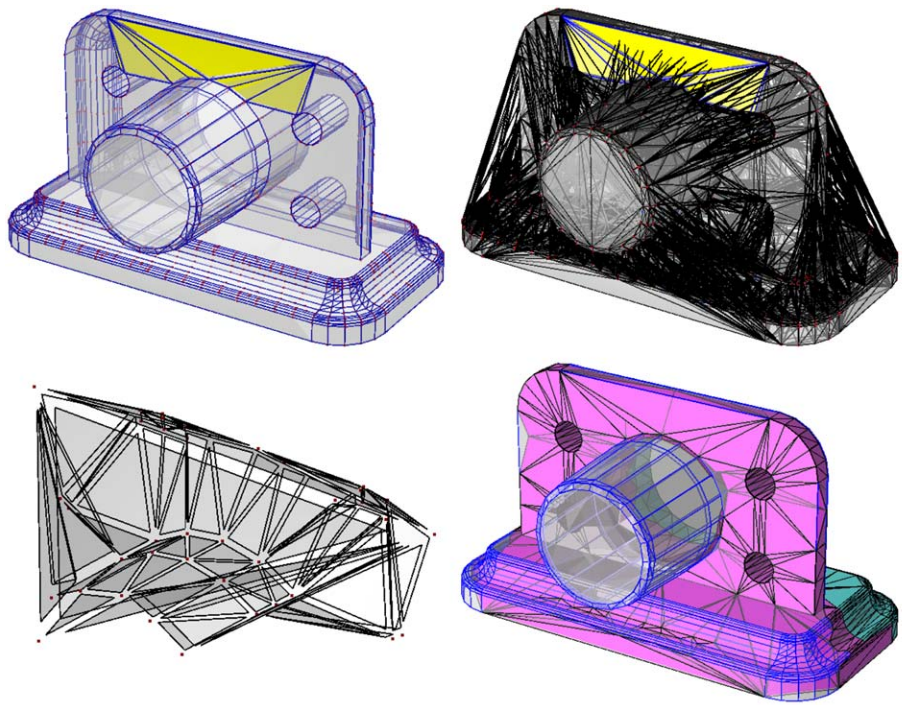
Fig. 13. An example of the facet recovery algorithm. A missing region Ω formed by nine subfaces is highlighted in the top left. This region is crossed by tetrahedra in the current constrained Delaunay tetrahedralization shown in top right. At bottom left, a cavity formed from Ω is shown. This cavity will be split into two subcavities by inserting the missing region. Bottom right is the CDT after the recovery of all missing regions.

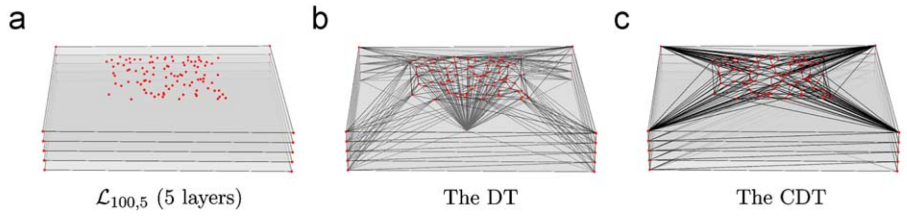

Fig. 14. Examples (layers). The PLS $\mathscr{L}_{100,5}$ shown in (a) has 5 parallel facets, 100 vertices lie above the top facet, one vertex below the center of the bottom facet. (b) and (c) respectively show the Delaunay tetrahedralization and the CDT of $\mathscr{L}_{100,5}$. (a) $\mathscr{L}_{100,5}$ (five layers), (b) the DT, and (c) the CDT.

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 [42], 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 *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 or maximum dihedral angle $\phi_{min}(\tau)$ of τ is a tetrahedron shape measure, see Fig. 15 for examples.

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

The function H is *isotropic* if the edge length does not vary with respect to the directions at $\mathbf{p}$, otherwise it is *anisotropic*. In this paper, we assume that H is isotropic. An ideal sizing function is $C^\infty, \forall \mathbf{p} \in |\mathscr{X}|$. However, in most cases, H is approximated by a discrete function

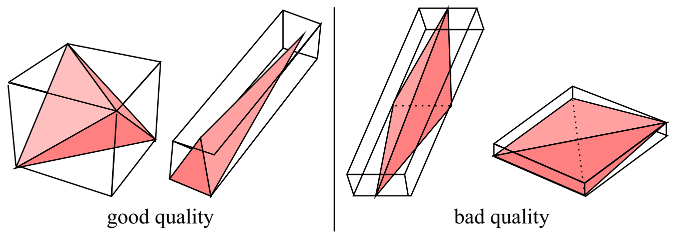

Fig. 15. Tetrahedra shape quality classified by the minimum and maximum dihedral angle criterion. The tetrahedra on the left contains no too small and too large dihedral angles, but the right ones do.

specified at some points in $|\mathscr{X}|$. 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 $\mathscr{T}$ of $\mathscr{X}$ such that the mesh size of $\mathscr{T}$ conforms to H. We use the following criterion to measure the mesh size conformity. Let $\mathbf{p}$ be a vertex in $\mathscr{T}$. 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 $\mathscr{T}$ *conforms* to H if there exist two constants C_S and C_L, where $0 < C_S \le C_L < \infty$, such that for every vertex $\mathbf{p} \in \mathscr{T}$, the following relation holds:

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

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 C_L/C_S.

5.2. The algorithm

Starting with an initial Delaunay tetrahedralization, Shewchuk's Delaunay refinement scheme [42] 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 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. 16). 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.

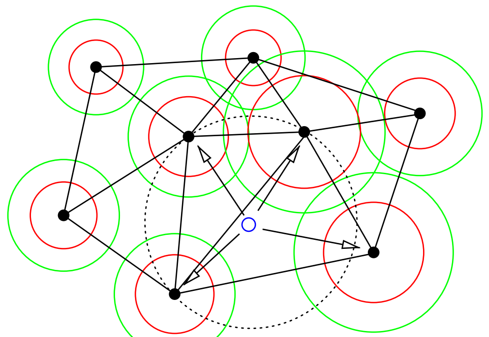

Fig. 16. For each point $\mathbf{p} \in \mathscr{T}$, assume there are two virtual balls, one protect ball (shown in red) and one sparse ball (shown in green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$R3$: 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;
$R3.2$: 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. 17).
- 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 $bfq \notin P$ 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 AdaptiveDelauneyRefinement algorithm is described in Fig. 18. It first initializes a queue Q of all tetrahedra of $\mathscr{T}$. 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 $\mathscr{T}$ may not be a boundary conforming Delaunay mesh, $\mathbf{v}$ may lie outside $|\mathscr{T}|$, $\mathbf{v}$ can be considered for insertion if $\mathbf{v} \in |\mathscr{T}|$ (lines 6–14). Finally, $\mathbf{v}$ can be inserted if it passes the point insertion rule (lines 8–12). The new mesh of $\mathscr{T} \cup \{\mathbf{v}\}$ is created by the Delaunay criterion

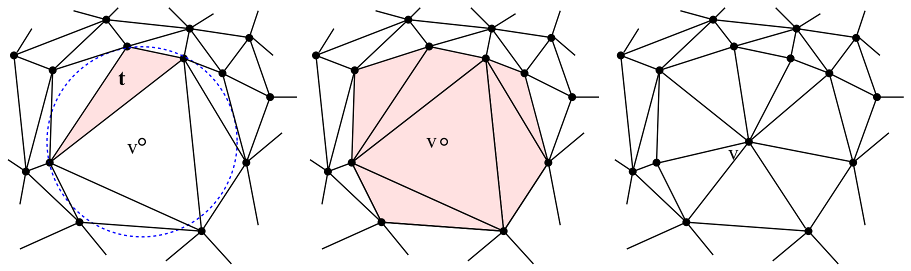

Fig. 17. Left: **v** is a point found by *R*2. Middle: the vertices of the shaded region form the set *P* of points collected by the point-accepting rule. Right: if **v** is inserted, the shaded region is re-triangulated according to the Delaunay criterion.

```
AdaptiveDelaunayRefinement ($\mathcal{T}$, $\rho_0$, $H$, $\alpha_1$, $\alpha_2$)
// $\mathcal{T}$ is a tetrahedral mesh of a PLS $\mathcal{X}$; $\rho_0$ is a radius-edge ratio bound;
// $H : |\mathcal{X}| \to \mathbb{R}^+$ is a sizing function; $\alpha_1$, $\alpha_2$ are two parameters.
1.  initialize a queue $Q$ of all tetrahedra of $\mathcal{T}$;
2.  while $Q \neq \emptyset$, do
3.      pop a tetrahedron $\tau$ 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.                 $Q = Q \cup \{\nu \in \mathcal{T} \mid \mathbf{v} < \nu\}$;
12.             endif
13.         endif
14.     endif
15. endwhile
16. return $\mathcal{T}$;
```

Fig. 18. The adaptive Delaunay refinement algorithm.

(line 9). If **v** was generated by *R*1 or *R*2, i.e., $\mathbf{v} \neq \mathbf{c}_\tau$, the insertion of **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 (*Si [49]*). *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. 18 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 (*Si [49]*). *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 (*Si [49]*). *Let $H = \text{lfs}$, $\theta_m = 90°$, $\rho_0 > 2$, and let α_2 be small enough such that all output tetrahedra have a radius-edge ratio smaller than ρ_0. Then*

(i) $S_v \geq \min\{\alpha_2, C\alpha_2 H(p(\mathbf{v}))/H(\mathbf{v})\}$;
(ii) $L_v \leq 2\alpha_1$

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

6. Examples

The algorithm presented in this paper has been implemented in the program `TetGen` [48]. The input can be either a PLS or a constrained tetrahedral mesh. A sizing function *H* can be optionally specified through a background mesh. Parameters ρ_0, α_1, and α_2 are all adjustable at runtime. In this section, we will present several examples to illustrate the practicality of the discussed algorithms.

The first mesh example is a test example for the CDT algorithm. Then the original input is a triangular surface mesh of a geological model shown in Fig. 19(a) (6116 nodes, 12,270 triangles). We manually added an internal facet between the top and the bottom surfaces, see Fig. 19b. Since the area of the facet is much larger than the vertical height of this model, this facet must be crossed by many Delaunay edges connecting from points on top and bottom. A cavity formed from the missing subfaces of this facet is shown in Fig. 19c, it has 1954 faces. The facet recovery algorithm will recover this facet. The final CDT is shown in Fig. 19d. To generate this CDT, `TetGen` added 2723 Steiner points (on segments), and 90 missing regions were recovered. The largest missing region has 21 subfaces. The total running time is 0.53 s (in detail, initial Delaunay tetrahedralization 0.21 s, process surface mesh 0.07 s, segment and facet recovery 0.25 s). The test is performed on a laptop with an Intel Core 2 Duo 2.16 Hz CPU.

The second example is a Boeing 747 model. The input is the surface mesh of the plane (2874 nodes, 5738 triangles) plus a bounding box, see Fig. 20(a) and (b). 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. The resulting tetrahedral mesh (Fig. 20(c) and (d) 490,692 nodes, 2,709,770 tetrahedra) was generated with parameters: $\rho_0 = 2.0, \alpha_1 = 0.5, \alpha_2 = 0.25$.

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. These tetrahedra are all close to the sharp segments of the plane's surface.

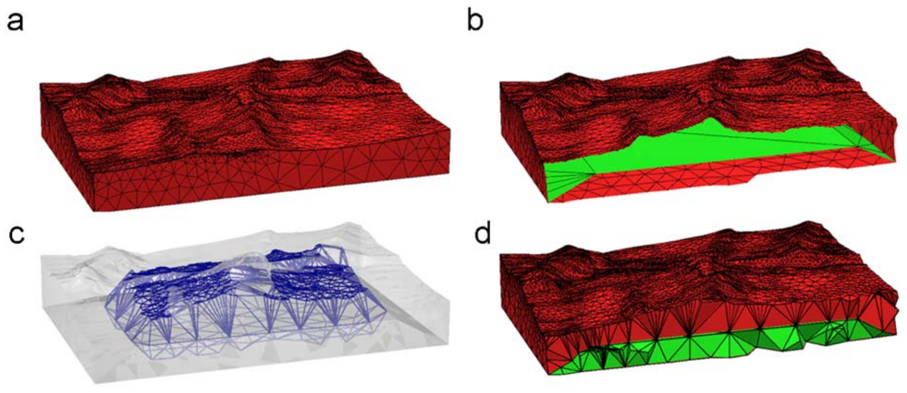

Fig. 19. Constrained Delaunay mesh of a geological model. The surface mesh is shown in (a). It also contains an internal facet between the top and bottom surfaces, see (b). (c) shows a formed cavity during the facet recovery algorithm. The size of the cavity is 1954 faces. (d) shows a view of the resulting CDT.

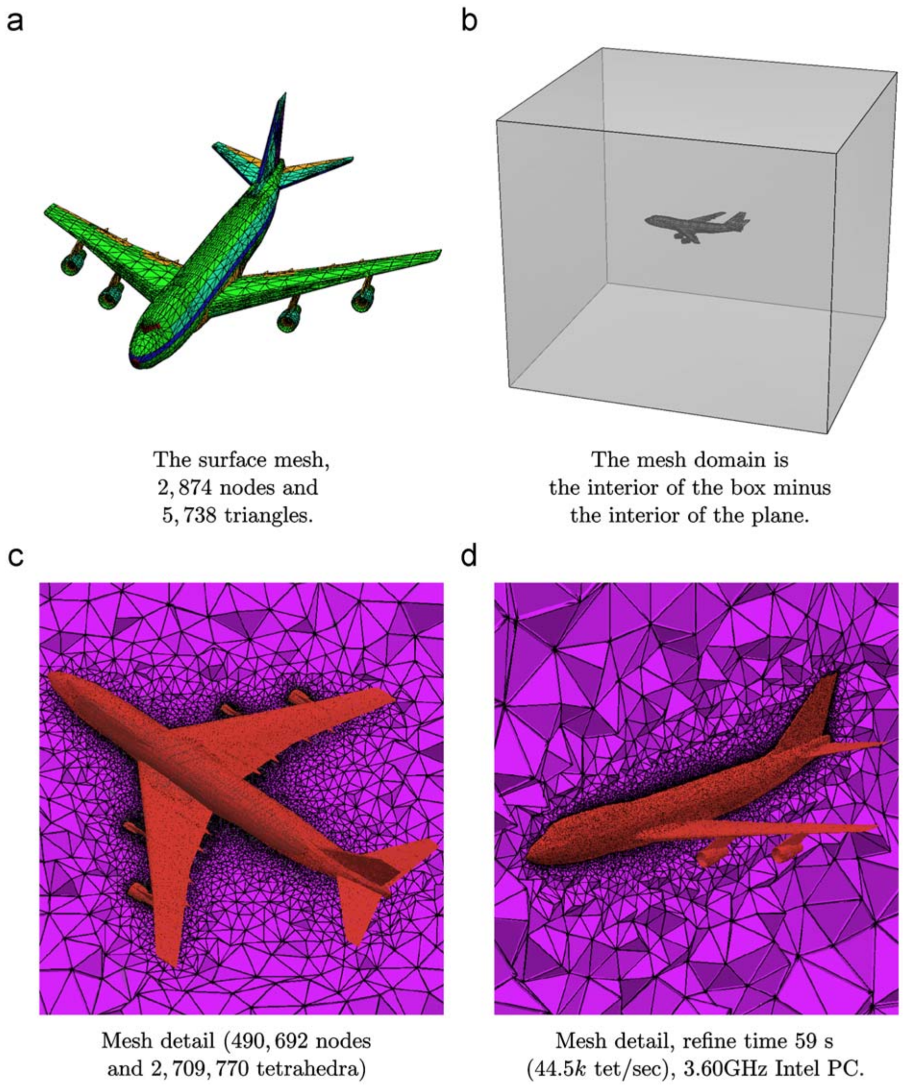
a
b
The surface mesh, 2,874 nodes and 5,738 triangles.
The mesh domain is the interior of the box minus the interior of the plane.
c
d
Mesh detail (490,692 nodes and 2,709,770 tetrahedra)
Mesh detail, refine time 59 s (44.5k tet/sec), 3.60GHz Intel PC.

Fig. 20. Adaptive tetrahedral mesh of the Boeing 747 model. (a) The surface mesh, 2874 nodes and 5738 triangles, (b) the mesh domain is the interior of the box minus the interior of the plane, (c) mesh detail (490,692 nodes and 2,709,770 tetrahedra) and (d) mesh detail, refine time 59 s (44.5k tec/s), 3.60 GHz Intel PC.

The input of the last example is a surface mesh of a rat lung (162,923 nodes, 325,842 triangles), see Fig. 21 left. The discussed CDT algorithm and the mesh refinement algorithm are applied together to generate a good quality tetrahedral mesh on it. The final tetrahedral mesh has 233,450 nodes and 888,855 tetrahedra, see Fig. 21 right. The running time of the CDT algorithm was 16.32 s

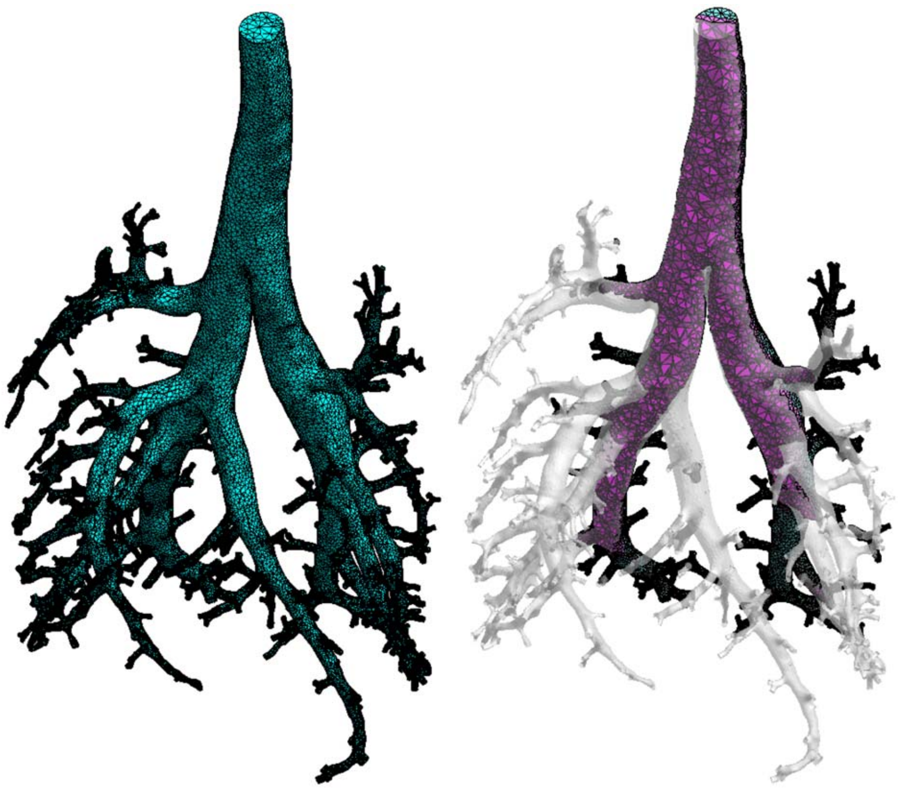

Fig. 21. The surface mesh (left) and a tetrahedral mesh (right) of a rat lung.

(in detail, initial Delaunay tetrahedralization 12.07 s, process surface mesh 1.32 s, segment and facet recovery 2.93 s). The time for mesh refinement was 25.34 s. (On a laptop with an Intel Core 2 Duo 2.16 Hz CPU.)

7. Open issues

The only obstacle for obtaining an isotropic good quality Delaunay mesh is the limitation (e.g., $\geq 69.3^\circ$) on facet angles of the input PLS [50]. 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. [10] and Pav et al. [37]. 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 $\mathscr{X}$ in a Delaunay triangulation. So far, only Edelsbrunner et al. [22] 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 [25], the robustness and efficiency are the main difficulties in practice.

Acknowledgments

The author would like to thank Dr. Jürgen Fuhrmann and Dr. Klaus Gärtner for stimulating this work and for helpful discussions. Special thanks to Dr. Andreas Rathsfeld for carefully reading and improving this manuscript. Special thanks also to Daniel Einstein for providing the rat lung test model.

References

[1] I. Babuška, A.K. Aziz, On the angle condition in the finite element method, SIAM Journal on Numerical Analysis 13 (2) (1976) 214–226.
[2] I. Babuška, J.E. Flaherty, W.D. Henshaw, J.E. Hopcroft, J.E. Oliger, T. Tezduyar, (Eds.), Modeling, mesh generation, and adaptive numerical methods for partial differential equations, in: The IMA Volumes in Mathematics and its Applications, vol. 75, Springer, Berlin, 1995.
[3] I. Babuška, W.C. Rheinboldt, Adaptive approaches and reliability estimates in finite element analysis, Computer Methods in Applied Mechanics and Engineering 17/18 (1987) 519–540.
[4] T. Baker, Automatic mesh generation for complex three-dimensional regions using a constrained Delaunay triangulation, Engineering with Computers 5 (1989) 161–175.
[5] M.W. Bern, J.E. Flaherty, M. Luskin (Eds.), Grid generation and adaptive algorithms, in: The IMA Volumes in Mathematics and its Applications, vol. 113, Springer, Berlin, 1999.
[6] P.R. Cavalcanti, U.T. Mello, Three-dimensional constrained Delaunay triangulation: a minimalist approach, in: Proceedings of 8th International Meshing Roundtable, Sandia National Laboratories, 1999, pp. 119–129.
[7] J.C. Cavendish, D.A. Field, W.H. Frey, An approach to automatic three-dimensional finite element mesh generation, International Journal for Numerical Methods in Engineering 21 (1985) 329–347.
[8] B. Chazelle, Convex partition of a polyhedra: a lower bound and worst-case optimal algorithm, SIAM Journal on Computing 13 (3) (1984) 488–507.
[9] B. Chazelle, L. Palios, Triangulating a nonconvex polytope, Discrete and Computational Geometry 5 (1990) 505–526.
[10] S.-W. Cheng, T.K. Dey, E.A. Ramos, T. Ray, Quality meshing for polyhedra with small angles, International Journal on Computational Geometry and Applications 15 (2005) 421–461.
[11] P.L. Chew, Constrained Delaunay triangulation, Algorithmica 4 (1989) 97–108.
[12] P.L. Chew, Guaranteed-quality triangular meshes. Technical Report TR 89–983, Department of Computer Science, Cornell University, 1989.

[13] P.L. Chew, Guaranteed-quality Delaunay meshing in 3D, in: Proceedings of 13th Annual ACM Symposium on Computational Geometry, Nice, France, June 1997, pp. 391–393.
[14] D. Cohen-Steiner, E.C. De Verdière, M. Yvinec, Conforming Delaunay triangulation in 3D, in: Proceedings of 18th Annual ACM Symposium on Computational Geometry, 2002.
[15] B.N. Delaunay, Sur la sphère vide, Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk 7 (1934) 793–800.
[16] O. Devillers, M. Teillaud, Perturbations and vertex removal in Delaunay and regular 3D triangulations, Technical Report 5968, INRIA, 2006.
[17] W. Dörfler, A convergent adaptive algorithm for Poisson's equation, SIAM Journal on Numerical Analysis 33 (1996) 1106–1124.
[18] Q. Du, D. Wang, Boundary recovery for three dimensional conforming Delaunay triangulation, Computer Methods in Applied Mechanics and Engineering 193 (2004) 2547–2563.
[19] Q. Du, D. Wang, Constrained boundary recovery for the three dimensional Delaunay triangulations, International Journal for Numerical Methods in Engineering 61 (2004) 1471–1500.
[20] H. Edelsbrunner, Geometry and Topology for Mesh Generation, Cambridge University Press, England, 2001.
[21] H. Edelsbrunner, M.P. Mücke, Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithm, ACM Transactions on Graphics 9 (1) (1990) 66–104.
[22] H. Edelsbrunner, T.S. Tan, An upper bound for conforming Delaunay triangulations, SIAM Journal on Computing 22 (1993) 527–551.
[23] P.J. Frey, P.L. George, Mesh Generation, Application to Finite Elements, Hermes Science, Oxford, UK, 2000.
[24] P.L. George, H. Borouchaki, P. Laug, An efficient algorithm for 3D adaptive meshing, Advances in Engineering Software 33 (2002) 377–387.
[25] P.L. George, H. Borouchaki, E. Saltel, Ultimate robustness in meshing an arbitrary polyhedron, International Journal for Numerical Methods in Engineering 58 (2003) 1061–1089.
[26] P.L. George, F. Hecht, E. Saltel, Automatic mesh generator with specified boundary, Computer Methods in Applied Mechanics and Engineering 92 (1991) 269–288.
[27] C. Hazlewood, Approximating constrained tetrahedralizations, Computer Aided Geometric Design 10 (1993) 67–87.
[28] C. Johnson, P. Hansbo, Adaptive finite element methods in computational mechanics, Computer Methods in Applied Mechanics and Engineering 101 (1992) 143–181.
[29] B.K. Karamete, M.W. Beall, M.S. Shephard, Triangulation of arbitrary polyhedra to support automatic mesh generators, International Journal for Numerical Methods in Engineering 49 (2000) 167–191.
[30] D.T. Lee, A.K. Lin, Generalized Delaunay triangulations for planar graphs, Discrete and Computational Geometry 1 (1986) 201–217.
[31] S.H. Lo, A new mesh generation scheme for arbitrary planar domains, International Journal for Numerical Methods in Engineering 21 (1985) 1403–1426.
[32] R. Löhner, P. Parikh, Three-dimensional grid generation by the advancing front method, International Journal for Numerical Methods in Fluids 8 (1988) 1135–1149.
[33] D.J. Mavriplis, Unstructured mesh generation and adaptivity, Technical Report 95-26, NASA Contractor Report 195069, ICASE, 1995.
[34] G.L. Miller, D. Talmor, S.-H. Teng, N.J. Walkington, H. Wang, Control volume meshes using sphere packing: generation, refinement and coarsening, in: Proceedings of 5th International Meshing Roundtable, Sandia National Laboratories, 1996.
[35] S.A. Mitchell, S.A. Vavasis, Quality mesh generation in higher dimensions, SIAM Journal on Computing 29 (2000) 1334–1370.
[36] M. Murphy, D.M. Mount, C.W. Gable, A point-placement strategy for conforming Delaunay tetrahedralizations, in: Proceedings of 11th Annual ACM-SIAM Symposium on Discrete Algorithms, 2000, pp. 67–74.
[37] S.E. Pav, N.J. Walkington, Robust three dimensional Delaunay refinement, in: Proceedings of 13th International Meshing Roundtable, Sandia National Laboratories, 2004.
[38] P. Pébay, P.J. Frey, A priori Delaunay-conformity, in: Proceedings of 7th International Meshing Roundtable, Sandia National Laboratories, 1998, pp. 321–333.
[39] J. Ruppert, R. Seidel, On the difficulty of triangulating three-dimensional non-convex polyhedra, Discrete and Computational Geometry 7 (1992) 227–253.
[40] E. Schönhardt, Über die zerlegung von dreieckspolyedern in tetraeder, Mathematische Annalen 98 (1928) 309–312.
[41] J.R. Shewchuk, A condition guaranteeing the existence of higher-dimensional constrained Delaunay triangulations, in: Proceedings of 14th Annual Symposium on Computational Geometry, June 1998, pp. 76–85.
[42] J.R. Shewchuk, Tetrahedral mesh generation by Delaunay refinement, in: Proceedings of 14th Annual Symposium on Computational Geometry, Minneapolis, Minnesota, USA, June 1998, pp. 86–95.
[43] J.R. Shewchuk, Sweep algorithm for constructing higher-dimensional constrained Delaunay triangulations, in: Proceedings of 16th Annual Symposium on Computational Geometry, June 2000, pp. 350–359.
[44] J.R. Shewchuk, Constrained Delaunay tetrahedralization and provably good boundary recovery, in: Proceedings of 11th International Meshing Roundtable, Sandia National Laboratories, September 2002, pp. 193–204.
[45] J.R. Shewchuk, What is a good linear element? Interpolation, conditioning, and quality measures, in: Proceedings of 11th International Meshing Roundtable, Sandia National Laboratories, Ithaca, NY, September 2002, pp. 115–126.
[46] J.R. Shewchuk, Updating and constructing constrained Delaunay and constrained regular triangulations by flips, in: Proceedings of 19th Annual Symposium on Computational Geometry, June 2003, pp. 181–190.
[47] J.R. Shewchuk, General-dimensional constrained Delaunay and constrained regular triangulations I: combinatorial properties, Discrete and Computational Geometry 39 (2008) 580–637.
[48] H. Si, TetGen, ⟨http://tetgen.berlios.de⟩, 2007.
[49] H. Si, Adaptive tetrahedral mesh generation by constrained Delaunay refinement, International Journal for Numerical Methods in Engineering 75 (7) (2008) 856–880.
[50] H. Si, Three dimensional boundary conforming Delaunay mesh generation, Ph.D. Thesis, Institute of Mathematics, Technische Universität Berlin, 2008.
[51] H. Si, K. Gärtner, Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations, in: Proceedings of 14th International Meshing Roundtable, Sandia National Laboratories, San Diego, CA, USA, 2005, pp. 147–163.
[52] The Boeing Company, GGNS, The General Geometry Navier–Stokes Solver, 2007, unpublished.
[53] R. Verfürth, A Review of Posteriori Error Estimation and Adaptive Mesh Refinement Techniques, Wiley-Teubner, 1996.
[54] N.P. Weatherill, O. Hassan, Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints, International Journal for Numerical Methods in Engineering 37 (1994) 2005–2039.
[55] J. Wright, A. Jack, Aspects of three-dimensional constrained Delaunay meshing, International Journal for Numerical Methods in Engineering 37 (1994) 1841–1861.
[56] M. Yerry, M.S. Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, International Journal for Numerical Methods in Engineering 20 (1984) 1965–1990.
[57] G.M. Ziegler, Lectures on polytopes, in: Graduate Texts in Mathematics, vol. 152, Springer, New York, 1995.