

An Introduction to Algorithms for Constructing Conforming Delaunay Tetrahedrizations

Marcelo Siqueira

Department of Computer and Information Science
School of Engineering and Applied Sciences
University of Pennsylvania

June 2003

Abstract

In many computational applications involving 3D geometric modeling, a space decomposition of the problem domain whose elements satisfy some quality constraints is often required. The Delaunay tetrahedrization is a popular choice due to its many important geometric properties and to the existence of efficient algorithms to compute and to maintain it. However, the Delaunay tetrahedrization has a serious limitation: Its underlying space is convex, and thus it can only induce a space subdivision of convex domains. The adaptation of the Delaunay tetrahedrization, defined over a finite set of points, to more complicated domains, such as arbitrary polyhedra, has proven to be a major challenge for both the mesh generation and computational geometry communities. The conforming Delaunay tetrahedrization is such an adaptation.

A conforming Delaunay tetrahedrization of an arbitrary polyhedron is the Delaunay tetrahedrization of some finite set of points in which the boundary of the polyhedron is represented by the vertices, edges, and facets of the tetrahedrization. The primary goal of any algorithm for constructing such a tetrahedrization is to find an appropriate set of points for which the Delaunay tetrahedrization conforms to the boundary of the input domain. It turns out that this goal poses many algorithmic challenges. For instance, it is not known if there is a polynomial upper bound for the size of the set of vertices of the tetrahedrization with respect to the number of vertices of the input polyhedron. Furthermore, there is no known solution that can fully reconcile boundary conformity and quality constraints for some of the most common quality metrics.

In this paper we review four algorithms for constructing a Delaunay tetrahedrization that conforms to the boundary of an arbitrary polyhedron and to an optional set of isolated vertices, edges, and facets lying in the interior of the polyhedron. All these algorithms are provably guaranteed to terminate and to satisfy some quality constraints. We present the proofs of correctness and termination of the algorithms, compare their contributions, advantages and weaknesses, report on their complexity issues, and highlight some of the most important unsolved questions related to the problem of obtaining a conforming Delaunay tetrahedrization.

Contents

1	Introduction	1
2	Mathematical Preliminaries	4
2.1	Euclidean Distance, Convex Hull, and Polyhedra	4
2.2	Simplex and Simplicial Complex	5
2.3	Piecewise-Linear Complex	7
2.4	Power Distance	7
2.5	Local Feature Size	8
2.6	Delaunay Complex	9
3	Conforming Delaunay Tetrahedrizations	13
3.1	Boundary Conformity	14
3.2	Existence and Complexity	14
3.3	Delaunay Refinement	16
3.3.1	Angle Constraint	16
3.3.2	Refinement Algorithm	16

3.3.3	Termination, Correctness and Complexity	19
3.4	PLCs with Small Angles	23
3.4.1	Small Angles	23
3.4.2	Protecting Balls	24
3.4.3	Refinement Algorithm	27
3.4.4	Termination and Correctness	28
3.5	Concluding Remarks	32
4	Provably Good Quality Tetrahedrizations	33
4.1	Quality Metrics	34
4.2	Sliver-Free Delaunay Refinement	35
4.2.1	Picking Region	35
4.2.2	Delaunay Refinement	37
4.2.3	Bounds on Small Slivers	38
4.2.4	Termination	39
4.3	Bounded Radius-Edge Ratio and PLCs with Small Angles	42
4.3.1	Augmented Complex Q	43
4.3.2	Delaunay Refinement	45
4.3.3	Boundary Conformity	47
4.3.4	Termination	50
4.4	Concluding Remarks	53
5	Conclusion	55

List of Figures

2.1	Examples of simplices of dimension 0, 1, 2, and 3 in \mathbb{R}^3 .	5
2.2	Collections of simplices in \mathbb{R}^2 . (a) and (b) are not simplicial complexes, but (c) is.	6
2.3	Examples of piecewise-linear complexes.	7
2.4	Two orthogonal spheres.	8
2.5	Example of the local feature size function in \mathbb{R}^2 .	9
2.6	Example of Voronoi diagram in the plane. The sites are represented by black points.	10
2.7	Example of a Voronoi region in \mathbb{R}^3 .	11
2.8	Example of a Delaunay triangulation and its dual.	12
3.1	The Schönhardt polyhedron.	15
3.2	Angle between (a) two edges, (b) an edge and a facet, and (c) two facets.	17
3.3	(a) An input PLC. (b) Its tetrahedrization.	17
3.4	Example of a “chain reaction” of mutual encroachments.	23
3.5	The neighborhood of a protecting ball.	25
3.6	The procedure split-on-a-sphere creates the isosceles sub-sectors avc and cvb from the isosceles sub-sector avb .	26

4.1	Tetrahedra with very small angles.	34
4.2	Picking region of a tetrahedron, triangle, and segment.	36
4.3	The forbidden region of a triangle.	38
4.4	Flat and curved facets, and linear and curved edges of the augmented complex.	44
4.5	The circumcap of a helper arc.	45
4.6	Helper triangles and non-helper triangles.	46

Chapter 1

Introduction

In many computational applications, such as surface interpolation, graphics rendering, and finite-element analysis, it is often required to find a space decomposition of the problem domain into smaller, disjoint sub-domains. Due to their topological and geometric simplicity, simplicial decompositions are in general the most used form of space decomposition [1]. In three-dimensions, a simplicial decomposition is induced by a *tetrahedrization*, i.e., a collection of openly disjoint tetrahedra and their boundary triangles, edges and vertices that covers the domain.

Most applications that use tetrahedrizations demand more than a mere tetrahedrization of the domain of interest. They usually require every tetrahedra to be “well-shaped”, having small aspect ratios or bounds on their smallest angle and largest size, so that the numerical computations involving these tetrahedra are guaranteed to be robust. Whenever the quality of a tetrahedrization is an issue, a popular choice is the *Delaunay tetrahedrization*. This tetrahedrization tends to favor “round” tetrahedra over “skinny” ones, and there are algorithms to construct them, called *Delaunay refinement* algorithms, that are provably guaranteed to meet constraints on tetrahedra aspect ratio and size [2, 3, 4].

However, a serious limitation of the Delaunay tetrahedrization is the fact that its domain is a convex polyhedron. While most domains of interest can be satisfactorily approximated by an arbitrary polyhedron, the convexity restriction narrows the range of applications of the Delaunay tetrahedrization. When the domain of interest is a non-convex polyhedron, possibly with an additional set of isolated vertices, edges and facets inside it, a better choice is to use a *conforming Delaunay tetrahedrization*.

Let R be a non-convex polyhedron possibly with an additional set F of isolated vertices, edges and facets inside it.

A Delaunay tetrahedrization that conforms to R and F is the Delaunay tetrahedrization of some set P of point in 3D such that the vertices of the boundary of R and those of F are vertices of the tetrahedrization, and the edges and facets of the boundary of R and those of F are the union of a subset of edges and triangles of the tetrahedrization, respectively. If we are able to obtain such a tetrahedrization, we can further remove its elements lying outside R so that the tetrahedra of the resulting tetrahedrization cover R exactly.

Conforming Delaunay tetrahedrizations can also be built by Delaunay refinement algorithms. These algorithms start with the Delaunay tetrahedrization of a set of points whose convex hull encloses the polyhedral domain R of interest. Unless we are lucky, this initial tetrahedrization will not conform to the boundary of R nor to F . While this is not the case, the algorithm keeps inserting vertices into the Delaunay tetrahedrization and updating it until the current Delaunay tetrahedrization conforms to the boundary of R and to F . There are two major issues in this kind of algorithm: Where to insert the next vertex and how to guarantee termination. Furthermore, conforming Delaunay refinement algorithms can also attempt to reconcile boundary conformity with quality constraints with respect to some formally specified quality metrics.

In this critical paper, we review four conforming Delaunay refinement algorithms. For all these four algorithms, the input consists of a set of vertices, edges, and faces defining the boundary of a bounded polyhedron R , and an optional set of vertices, edges, and faces inside R that the tetrahedrization is also supposed to conform to. The set of vertices, edges, and faces in the input defines a *piecewise-linear complex* (PLC). The output is a Delaunay tetrahedrization of some convex region enclosing the input PLC that is provably guaranteed to conform to its vertices, edges, and faces. For three out of the four algorithms, the output is also provably guaranteed to satisfy some quality constraints.

The first algorithm is due to Jonathan Shewchuk [5]. This algorithm is guaranteed to terminate only if the input PLC satisfies the *angle constraint*, i.e., the angle defined by a face and an edge adjacent to it, two adjacent edges, or two adjacent faces of the input PLC is no less than $\frac{\pi}{2}$. Shewchuk's algorithm also generates conforming Delaunay tetrahedrizations whose tetrahedra have bounded *radius-edge ratio*, a useful metric used by several applications to measure the quality of the tetrahedrization. Tetrahedra with small radius-edge ratio have no small angles, except for one type of tetrahedron: the *sliver*. The second algorithm is due to Xiang-Yang Li [6], and it consists of an extension of Shewchuk's algorithm that is free of slivers. Like Shewchuk's algorithm, Li's algorithm is also limited to PLCs satisfying the angle constraint. The third algorithm is due to Cohen-Steiner, Verdière, and Yvinec [7]. This algorithm produces conforming Delaunay tetrahedrization of input PLCs that may not satisfy the angle constraint. However, the output tetrahedrization has no quality guarantee. The fourth algorithm is due to Siu-Wing Cheng and Sheung-Hung Poon [8]. Their algorithm is an extension of Shewchuk's algorithm to handle input PLCs that may not satisfy the angle constraint. Like Shewchuk's algorithm, Cheng and Poon's algorithm generates conforming

Delaunay tetrahedrizations whose tetrahedra have bounded radius-edge ratio, but they are not free of slivers.

The algorithms by Shewchuk [5], Xiang-Yang Li [6], and Siu-Wing Cheng and Sheung-Hung Poon [8] are also provably guaranteed to generate good graded tetrahedrizations. That is, the size of a tetrahedron is adapted to the local features of the input domain. This is also a very desirable property in practice as tetrahedrizations that employ roughly same-sized tetrahedra in general have more tetrahedra than are necessary to represent well the local features, and thus they can impose an overburden on the applications that use them. For the sake of space, however, we will not cover the grading property of these three aforementioned algorithms.

Given an input PLC \mathcal{C} that may not satisfy the angle constraint, it remains an open problem to develop an algorithm for generating a conforming Delaunay tetrahedrization of \mathcal{C} such that the tetrahedrization is free of slivers and its tetrahedra have bounded radius-edge ratio. It also remains an open problem to find a polynomial upper bound, if any, for the number of insertion points needed to enforce boundary conformity of any input PLC. In particular, all four algorithms aforementioned can generate an exponential number of insertion points. The time complexity of these four algorithms is not analyzed in the papers describing them, and this is also an interesting problem to investigate.

This paper is organized as follows. In Chapter 2, we provide several concepts useful to understand notations and relevant definitions in the following chapters. In Chapter 3, we focus on the problem of boundary conformity, and describe a simplified version of Shewchuk's algorithm and the algorithm by Cohen-Steiner, Verdière, and Yvinec. The simplified version of Shewchuk's algorithm only deals with the boundary conformity problem. In Chapter 4, we formally define radius-edge ratio and slivers, and study the algorithms by Li and Cheng and Poon, which reconcile boundary conformity and quality constraints. In Chapter 5, we summarize the main contributions of each algorithm, compare their contributions, and discuss future research directions.

Chapter 2

Mathematical Preliminaries

This chapter introduces notations and basic concepts used throughout this paper. The material in this chapter can be found in Boissonnat and Yvinec [9], Edelsbrunner [10], Gallier [11], Okabe, Boots, Sugihara and Chiu [12].

2.1 Euclidean Distance, Convex Hull, and Polyhedra

We denote the Euclidean space by \mathbb{R}^m , and we define the *Euclidean distance* $d(x, y)$ between any two points $x = (x_1, \dots, x_n)$ and $y = (y_1, \dots, y_n)$ in \mathbb{R}^m to be

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}.$$

Given a subset P of \mathbb{R}^m , we say that P is *convex* if for any two points $q, r \in P$, we have $p \in P$ for every point $p = (1 - \lambda)q + \lambda r$, with $0 \leq \lambda \leq 1$, $\lambda \in \mathbb{R}$. The empty set is trivially convex, every unit set is convex, and the entire space \mathbb{R}^m is convex. Given any non-empty subset P of \mathbb{R}^m , there is a smallest set containing P denoted by $\text{conv}(P)$ and called the *convex hull of P* . If P is any subset of $n > 0$ points p_1, \dots, p_n of \mathbb{R}^m then the set of all *convex combinations* $\sum_{i=1}^n \lambda_i p_i$, where $\sum_{i=1}^n \lambda_i = 1$ and $\lambda_i \geq 0$, is the convex hull $\text{conv}(P)$ of the set P .

We call a subset R of points of \mathbb{R}^m a *polyhedron* if it is a non-empty, connected bounded¹ set obtained from a finite number of intersection and union operations on a finite number of half-spaces. A polyhedron can be convex or nonconvex. The boundary of a polyhedron in \mathbb{R}^m consists of subsets of points in \mathbb{R}^m , called $(m - 1)$ -*faces*, that are

¹There are several references that omit the word ‘bounded’ from the definition of polyhedron.

isomorphic to polyhedra in \mathbb{R}^{m-1} ; the boundary of an $(m-1)$ -face consists of subsets of points in \mathbb{R}^{m-2} , called $(m-2)$ -faces, that are isomorphic to polyhedra in \mathbb{R}^{m-2} ; and so on. The 0-faces of a polyhedron are points and they are also called *vertices*, the 1-faces are line segments and they are also called *edges*, and the $(m-1)$ -faces are called *facets*.

2.2 Simplex and Simplicial Complex

Given a collection of $n+1$ affinely independent points in \mathbb{R}^m , $V = \{v_0, v_1, \dots, v_n\}$, the n -simplex (or simplex) $\sigma = [v_0, v_1, \dots, v_n]$ spanned by V is the convex hull of V , $\text{conv}(V)$. That is, the set of all convex combinations $\sum_{i=0}^n \lambda_i v_i$, where $\sum_{i=0}^n \lambda_i = 1$ and $\lambda_i \geq 0$, with $0 \leq i \leq n$. The *dimension* of σ , $\dim(\sigma)$, is n . In \mathbb{R}^m , the largest number of affinely independent points is $m+1$, and we have simplices of dimension $0, 1, \dots, m$. Figure 2.1 shows examples of the four types of simplices in \mathbb{R}^3 . A 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron.

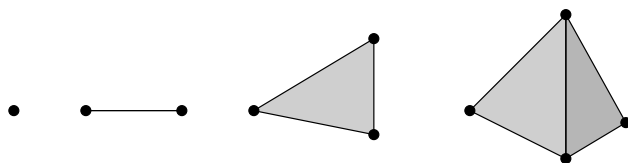


Figure 2.1: Examples of simplices of dimension 0, 1, 2, and 3 in \mathbb{R}^3 .

Note that the convex hull of any nonempty subset $V' \subseteq V$ is again a simplex. The simplex spanned by V' , say τ , is called a *face* of σ and the inclusion relation is denoted as $\tau \prec \sigma$. If $\dim(\tau) = d$ then τ is called an d -face of σ . If d is 0 then τ is called a *vertex*. If d is 1 then τ is called an *edge*. If $n = m$ and $\dim(\tau) = m-1$ then τ is called a *facet* of σ . If $\tau = \sigma$ then τ is an *improper* face, and all others are *proper* faces of σ . The *boundary* $bd(\sigma)$ of a simplex σ is the union of its proper faces. The *relative interior* of a simplex σ , denoted by $\text{int}(\sigma)$, arises by removing its boundary. The number of d -faces of σ is equal to the number of ways we can choose $d+1$ from $n+1$ points, which is $\binom{n+1}{d+1}$. Thus, the total number of faces of σ is

$$\sum_{d=0}^n \binom{n+1}{d+1} = 2^{n+1} - 1.$$

A *simplicial complex* \mathcal{K} is a finite collection of simplices satisfying the following two conditions:

1. If $\sigma \in \mathcal{K}$ and $\tau \prec \sigma$ then $\tau \in \mathcal{K}$;

2. If $\sigma, \tau \in \mathcal{K}$ then if $\sigma \cap \tau \neq \emptyset$, we have $\sigma \cap \tau \prec \sigma$ and $\sigma \cap \tau \prec \tau$.

Figure 2.2 shows three sets of simplices in \mathbb{R}^2 . The set on the left is not a simplicial complex because it is missing an edge and a vertex. The set in the middle contains two simplices that intersect each other but the intersection is not a face of either one, and therefore it cannot be a simplicial complex. The set on the right is a simplicial complex.

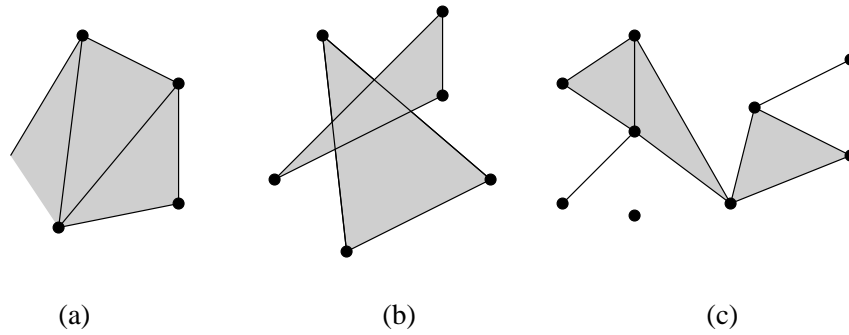


Figure 2.2: Collections of simplices in \mathbb{R}^2 . (a) and (b) are not simplicial complexes, but (c) is.

Let \mathcal{K} be a simplicial complex. The *dimension* $\dim(\mathcal{K})$ of \mathcal{K} is the largest dimension of a face in \mathcal{K} , i.e., $\dim(\mathcal{K}) = \max\{\dim(\sigma) \mid \sigma \in \mathcal{K}\}$. The set consisting of the union of all points in the simplices of \mathcal{K} is called the *underlying space* of \mathcal{K} , or the *polyhedron* of \mathcal{K} , or the *geometric realization* of \mathcal{K} , and it is denoted by $|\mathcal{K}|$. Since \mathcal{K} is a finite collection of simplices, its underlying space $|\mathcal{K}|$ is a compact set.

A *subcomplex* of a simplicial complex in \mathcal{K} is a subset of \mathcal{K} that is itself a simplicial complex. An important example of a subcomplex is the d -skeleton $\mathcal{K}^{\leq d}$ of a simplicial complex \mathcal{K} . It consists of all simplices of \mathcal{K} of dimension at most d , i.e., $\mathcal{K}^{\leq d} = \{\sigma \in \mathcal{K} \mid \dim(\sigma) \leq d\}$. Further we define $\mathcal{K}^d = \{\sigma \in \mathcal{K} \mid \dim(\sigma) = d\}$ to be the subset of simplices in \mathcal{K} of dimension d . In particular, \mathcal{K}^0 is the set of vertices of \mathcal{K} . Note that \mathcal{K}^d is not a simplicial complex.

Given a collection of $n+1$ affinely independent points $V = \{v_0, v_1, \dots, v_n\}$ in \mathbb{R}^m , the *open n -simplex* σ' spanned by V is the relative interior $\text{int}(\sigma)$ of the n -simplex $\sigma = [v_0, v_1, \dots, v_n]$ defined by V . If \mathcal{K} is a simplicial complex then the collection K' consisting of the relative interior of all simplices in \mathcal{K} induces a space partition of $|\mathcal{K}|$. As we shall see later, the main problem discussed in this paper can be viewed as the one of obtaining a *simplicial decomposition* of a polyhedron $R \subseteq \mathbb{R}^3$. That is, to obtain a simplicial complex \mathcal{K} such that the collection K' consisting of the relative interior of all simplices in \mathcal{K} is a space partition R . If such a simplicial complex exists, then clearly $|\mathcal{K}| = R$.

2.3 Piecewise-Linear Complex

We use the term *cell* to denote objects in \mathbb{R}^3 which are either 0-dimensional cells called vertices, 1-dimensional cells called edges, and 2-dimensional cells called facets. A *vertex* is just a point in \mathbb{R}^3 , an *edge* is a line straight segment in \mathbb{R}^3 , and a *facet* is a polygonal region in \mathbb{R}^3 possibly with holes and isolated edges and vertices in its interior. A *piecewise-linear complex* C , called for short PLC, is a finite collection of cells in \mathbb{R}^3 such that C satisfies the following two conditions:

1. The boundary of any cell of C is the union of cells of C .
2. The intersection of any two cells of C is either empty or a union of cells of C .

Figure 2.3 illustrates three piecewise-linear complexes. All conforming Delaunay refinement algorithms we study in the next chapters take a PLC as input.

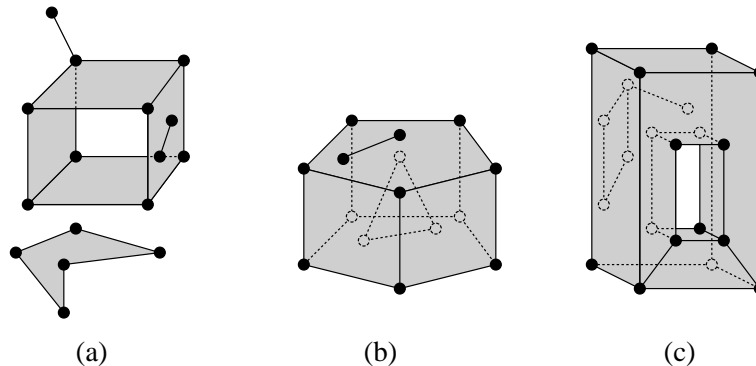


Figure 2.3: Examples of piecewise-linear complexes.

2.4 Power Distance

Let $S \subset \mathbb{R}^3$ be a sphere centered at $c \in \mathbb{R}^3$ and with radius $r > 0$. The *power or weighted distance function* of S is the map $\pi_S : \mathbb{R}^3 \mapsto \mathbb{R}$ given by

$$\pi_S(x) = d(x, c)^2 - r^2,$$

for every $x \in \mathbb{R}^3$, where $d(y, z)$ is the Euclidean distance function in \mathbb{R}^3 between $y, z \in \mathbb{R}^3$. The power distance function of S is positive for points outside S , zero for points on S , and negative for points inside S . The various cases

permit intuitive geometric interpretation. For example, if a point x is outside S then $\pi_S(x)$ is the square length of a tangent line segment connecting x with a point on S . The locus of points with equal distance from two spheres in \mathbb{R}^3 is a plane H . If the two spheres intersect each other then H passes through their intersection circle, and if the two spheres are disjoint and lie outside each other then H separates the two spheres.

Given two spheres $S_1 \subseteq \mathbb{R}^3$ and $S_2 \subseteq \mathbb{R}^3$ centered at c_1 and c_2 and with radius $r_1 > 0$ and $r_2 > 0$, respectively, we generalize the power distance function to the symmetric form

$$\pi_{S_1, S_2} = d(c_1, c_2)^2 - r_1^2 - r_2^2.$$

We say that S_1 and S_2 are *orthogonal* if $\pi_{S_1, S_2} = 0$. Note that two spheres are orthogonal if and only if they meet in a circle C and the tangent plane H_1 of S_1 and the tangent plane H_2 of S_2 at every point $x \in C$ form a right angle. Note also that the center c_1 (resp. c_2) of S_1 (resp. S_2) has to lie outside S_2 (resp. S_1). Figure 2.4 shows two orthogonal spheres. Orthogonality of spheres is a key concept to understand the algorithm by Cheng and Poon in Chapter 4.

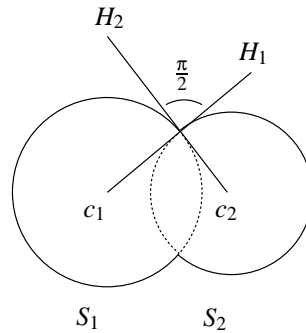


Figure 2.4: Two orthogonal spheres.

2.5 Local Feature Size

Given a PLC C , the *local feature size* at any point $p \in \mathbb{R}^3$ with respect to C is a function, $lfs : \mathbb{R}^3 \mapsto \mathbb{R}$, where $lfs(p)$ is the radius of the smallest ball centered at p that intersects two disjoint cells of C . Figure 2.5 illustrates this notion in two dimensions (with no facets) by showing such balls for several points. The points are represented by little white balls, and the balls are in gray. By definition, for any two vertices u and v of the PLC C , we have $lfs(u) \leq d(u, v)$ and $lfs(v) \leq d(u, v)$. The local feature size function lfs satisfies a one-sided Lipschitz inequality, which implies that lfs is continuous, as the following lemma shows:

Lemma 2.5.1. [10] Let C be a PLC, and let $lfs : \mathbb{R}^3 \mapsto \mathbb{R}$ be the local feature size at any point $p \in \mathbb{R}^3$ with respect to C . Then, the function lfs satisfies the Lipschitz condition $|lfs(x) - lfs(y)| \leq d(x, y)$.

Proof. Let us proceed by contradiction, i.e., say $lfs(y) \geq lfs(x)$, and assume that there are points x and y in \mathbb{R}^3 such that $lfs(x) < lfs(y) - d(x, y)$. Since $lfs(x) + d(x, y) < lfs(y)$, the ball B_1 centered at x with radius $lfs(x)$ is enclosed by the ball B_2 centered at y with radius $lfs(y)$. Thus, the ball B_2 also intersects the same two disjoint cells intersected by B_1 . Thus, there is a ball B_3 centered at y with radius no larger than $lfs(x) + d(x, y)$ that intersects two disjoint cells of C , which contradicts the definition of $lfs(y)$. \square

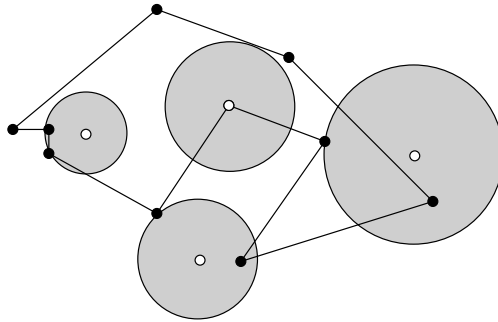


Figure 2.5: Example of the local feature size function in \mathbb{R}^2

The local feature size of the input PLC plays a crucial role in the proofs of termination of the conforming Delaunay refinement algorithms we study in the following chapters.

2.6 Delaunay Complex

Let $P \subset \mathbb{R}^m$ be a finite set of points in \mathbb{R}^m . The *Voronoi diagram* $\mathcal{V}(P)$ generated by P associates to each $q \in P$ a *Voronoi region* $V(q)$ such that

$$V(q) = \{x \in \mathbb{R}^m \mid d(x, q) \leq d(x, r), \forall r \in P\}.$$

We call a point $q \in P$ *site* or *generator* of $\mathcal{V}(P)$ in order to distinguish them from the other points in \mathbb{R}^m . An example of a Voronoi diagram $\mathcal{V}(P)$ for a set P of seven points in the Euclidean plane is given in Figure 2.6.

Consider the locus of points $B(q, r) = \{x \in \mathbb{R}^m \mid d(x, q) = d(x, r)\}$ of two distinct sites q and r of $P \subset \mathbb{R}^m$ of a Voronoi diagram $\mathcal{V}(P)$. The set $B(q, r)$ is a hyperplane perpendicular to the line segment \overline{qr} that bisects this segment in \mathbb{R}^m . The bisector hyperplane $B(q, r)$ divides the Euclidean space into two half-spaces. Denote the half-space that

contains the point q as $H(q, r) = \{x \in \mathbb{R}^m \mid d(x, q) \leq d(x, r)\}$. Note that $H(q, r)$ contains q and all points $x \in \mathbb{R}^m$ whose distance from q is shorter than or equal to the distance from r . So, an equivalent definition of the Voronoi region $V(q)$ of q is $V(q) = \bigcap_{r \in P} H(q, r)$. Because $V(q)$ is the intersection of a finite collection of half-spaces, the Voronoi region $V(q)$ is a convex polyhedron in \mathbb{R}^m .

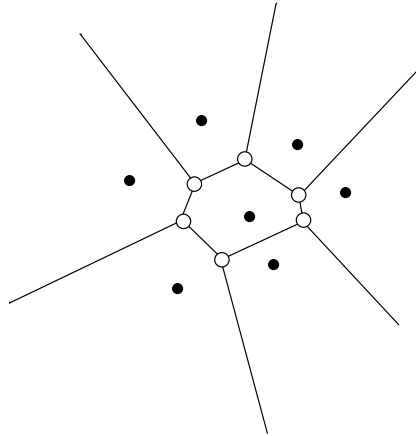


Figure 2.6: Example of Voronoi diagram in the plane. The sites are represented by black points.

The convex polyhedron representing a Voronoi region $V(q)$ of a site $q \in P$ can be unbounded. In fact, it can be shown that $V(q)$ is unbounded if and only if the site $q \in P$ belongs to the convex hull $\text{conv}(P)$ of P , [12]. Furthermore, if P has n sites then the boundary of $V(q)$ consists of up to $n - 1$ $(m - 1)$ -dimensional faces; the boundary of a $(m - 1)$ -dimensional face of $V(q)$ consists of $(m - 2)$ -dimensional faces, and so on. In particular, a 1-dimensional face is called a *Voronoi edge* and a 0-dimensional face is called a *Voronoi vertex*. Figure 2.7 shows the convex polyhedron (a cube) corresponding to the Voronoi region of a site q of a set $P \subset \mathbb{R}^3$ with seven elements. The point q is placed at the origin of an orthogonal reference frame. The other sites of P are the six points equi-distant from q and lying along the axes of the reference frame.

A point $p \in \mathbb{R}^m$ that belongs to n Voronoi regions is equally far from the n sites of these Voronoi regions. It follows that the n sites lie on a common $(m - 1)$ -sphere in \mathbb{R}^m . It can be shown that, if P is in *general position*, i.e., if no $m + 2$ sites of P lie on a common $(m - 1)$ -sphere, then $n \leq m + 1$. Furthermore, if p is a vertex of the Voronoi diagram $\mathcal{V}(P)$ generated by P and P is in general position, then p is the common point of exactly $m + 1$ Voronoi regions. If p is not in general position then p is the common point of at least $m + 1$ Voronoi regions. Note that the sphere centered at a vertex p of $\mathcal{V}(P)$ and containing the sites of the Voronoi regions that meet at p cannot contain any sites of P inside it. This is known as the *empty-sphere property* of Voronoi diagrams.

If $P \subset \mathbb{R}^3$ and P is in general position and no four points of P lie in a common circle in \mathbb{R}^3 , every vertex p of a

Voronoi region $V(q)$ in $\mathcal{V}(P)$ belongs to exactly three three-dimensional faces and three edges of $V(q)$. The Voronoi diagram $\mathcal{V}(P)$ also satisfies the *nearest-neighbor property*: If $q \in P$ is the nearest-neighbor of $r \in P$ then the Voronoi regions $V(q)$ and $V(r)$ share a common $(m - 1)$ -dimensional face.

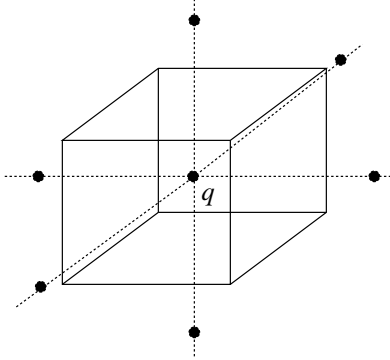


Figure 2.7: Example of a Voronoi region in \mathbb{R}^3 .

Let $\mathcal{V}(P)$ be the Voronoi diagram generated by a finite set $P \subset \mathbb{R}^m$ of n distinct points in \mathbb{R}^m such that $n \geq m + 1$ and not all points in P lie in a common hyperplane of \mathbb{R}^m . Then, we define the *Delaunay complex* $\mathcal{D}(P)$ of P to be the dual of $\mathcal{V}(P)$. That is, if $V = \{v_1, \dots, v_k\}$ is the set of Voronoi vertices of $\mathcal{V}(P)$, then the Delaunay complex is the collection $\mathcal{D}(P)$ consisting of k convex sets T_i and their boundary elements such that each T_i is given by

$$T_i = \{x \in \mathbb{R}^m \mid x = \sum_{j=1}^{n_i} \lambda_j q_{i,j}, \sum_{j=1}^{n_i} \lambda_j = 1, \lambda_j \geq 0\},$$

where $\{q_{i,1}, \dots, q_{i,n_i}\} \subseteq P$ is the set of sites of the Voronoi regions of $\mathcal{V}(P)$ meeting at the Voronoi vertex v_i , and n_i ($n_i \leq n$) is the number of elements of $\{q_{i,1}, \dots, q_{i,n_i}\}$. Note that if $n_i = m + 1$ and $\{q_{i,1}, \dots, q_{i,n_i}\}$ is a set of $m + 1$ affinely independent points then T_i is a m -simplex. In fact, if $n_i = m + 1$ for every $T_i \in \mathcal{D}(P)$ then it can be shown that every T_i is a m -simplex and $\mathcal{D}(P)$ is a simplicial complex. In this case, the Delaunay complex $\mathcal{D}(P)$ is called the *m -dimensional Delaunay tessellation* $\mathcal{DT}_m(P)$ of P . Furthermore, if $n_i > m + 1$ for some T_i , we have that T_i is a convex polyhedron and we can partition T_i into $(n_i - m)$ openly disjoint m -simplices by cutting T_i with hyperplanes passing through its vertices. So, we can always define a m -dimensional Delaunay tessellation $\mathcal{DT}_m(P)$ of P from the Delaunay complex \mathcal{D} . However, there can be more than one partition of T_i into $(n_i - m)$ m -simplices, and thus we can define more than one m -dimensional Delaunay tessellation $\mathcal{DT}_m(P)$ of P from $\mathcal{D}(P)$ if $n_i > m + 1$ for some $T_i \in \mathcal{D}$. Otherwise, the m -dimensional Delaunay tessellation $\mathcal{DT}_m(P)$ of P is unique.

A 2-dimensional Delaunay tessellation is known as a *Delaunay triangulation*, and a 3-dimensional Delaunay tessellation is known as a *Delaunay tetrahedrization*. Figure 2.8 shows the Delaunay triangulation $\mathcal{DT}_2(P)$ of the same set P of points that generated the Voronoi diagram in Figure 2.6. The duality between Delaunay complex

and Voronoi diagram is also illustrated in Figure 2.8. The vertices of $\mathcal{DT}_2(P)$ are the sites of $\mathcal{V}(P)$, each edge of $\mathcal{DT}_2(P)$ connects two sites $q, r \in P$ and is the dual of the edge of $\mathcal{V}(P)$ defined by the bisector line of p and q , and each triangle of $\mathcal{DT}_2(P)$ is the dual of the vertex of $\mathcal{V}(P)$ that is the common vertex of the Voronoi regions whose sites are three vertices of the triangle. Similarly, the vertices of a Delaunay tetrahedrization $\mathcal{DT}_3(P)$ are the sites of $\mathcal{V}(P)$, the edges of $\mathcal{DT}_3(P)$ are the dual of 2-dimensional faces of $\mathcal{V}(P)$, the facets of $\mathcal{DT}_3(P)$ are the dual of the edges of $\mathcal{V}(P)$, and the tetrahedra of $\mathcal{DT}_3(P)$ are the dual of the vertices of $\mathcal{V}(P)$.

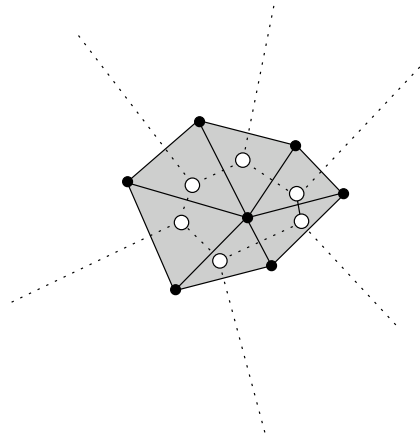


Figure 2.8: Example of a Delaunay triangulation and its dual.

Due to the empty-property of Voronoi diagrams, the *circumsphere* of any m -simplex σ of a m -dimensional Delaunay tessellation $\mathcal{DT}_m(P)$, i.e., the (unique) $(m-1)$ -sphere defined by the $m+1$ vertices of σ , contains no points of P inside it. Note that the center of the circumsphere (circumcenter) of m -simplex $\sigma \in \mathcal{DT}_m(P)$ is the Voronoi vertex that is the dual of σ in $\mathcal{V}(P)$. Likewise, the nearest-neighbor property of Voronoi diagrams implies that the edges of the m -dimensional Delaunay tessellation $\mathcal{DT}_m(P)$ of P connect the nearest vertices of $\mathcal{DT}_m(P)$.

The underlying space of the m -dimensional Delaunay tessellation $\mathcal{DT}_m(P)$ of P is a polytope in \mathbb{R}^m , which corresponds to the convex hull $\text{conv}(P)$ of P . It follows that $\mathcal{DT}_m(P)$ can only induce a simplicial subdivision of convex regions. In \mathbb{R}^3 , this limitation is the main motivation for the study of conforming Delaunay tetrahedrizations. Finally, there are several algorithms for computing the m -dimensional Delaunay tessellation $\mathcal{DT}_m(P)$ of P , and if the set P has n points, this simplicial complex can be computed in time $O(n \log n + n^{\lceil m/2 \rceil})$, and this is optimal in the worst case.

Chapter 3

Conforming Delaunay Tetrahedrizations

In Chapter 2 we defined the Delaunay tetrahedrization of a finite set P of points in \mathbb{R}^3 . The Delaunay tetrahedrization is very popular among several applications that require space subdivision. This is mainly due to the existence of algorithms that are provably guaranteed to generate good quality Delaunay tetrahedrizations [2, 3]. However, the Delaunay tetrahedrization has a serious limitation: Its underlying space is convex. This means that Delaunay tetrahedrizations can only decompose convex regions. In many applications, the problem domain is a polyhedron possibly, with a set isolated vertices, edges, and facets inside it.

Fortunately, even if the problem domain is not a convex region, we still can subdivide it by using a *conforming* Delaunay tetrahedrization. The idea is to build a Delaunay tetrahedrization of some set of points whose convex hull encloses the problem domain and such that this tetrahedrization *conforms* to the boundary of the domain and to any set of isolated vertices, edges, and facets inside it. That is, the vertices of the boundary of the domain (and the ones lying inside it) are vertices of the tetrahedrization, and the edges and facets of the boundary of the domain (and the ones lying inside it) are the union of edges and facets of the tetrahedrization. Finally, we can remove the simplices of the tetrahedrization that lie outside the domain so that the resulting tetrahedrization is a simplicial decomposition of the given domain. Note that, after removing the simplices lying outside the domain from the conforming Delaunay tetrahedrization, the resulting tetrahedrization may no longer be a Delaunay tetrahedrization of the set of the remaining vertices.

In this chapter we study two algorithms for constructing conforming Delaunay tetrahedrization of domains specified by piecewise-linear complexes. That is, the input is a PLC that represents the boundary of a polyhedron and the possible set of isolated vertices, edges, and faces inside it. The output is a Delaunay tetrahedrization of some set

of points that conforms to the input PLC. The underlying idea of both algorithms is to keep inserting new vertices into the tetrahedrization until it conforms to the input PLC. One algorithm is a simplified version of the algorithm by Jonathan Shewchuk [5], and the other one is due to Cohen-Steiner, Verdière, and Yvinec [7].

Section 3.1 formally states the problem of boundary conformity. Section 3.2 discusses existence and complexity of algorithms to generate conforming tetrahedrizations of polyhedral domains. Section 3.3 presents our simplified version of Shewchuk’s algorithm. Section 3.4 describes Cohen-Steiner, Verdière, and Yvinec’s algorithm.

3.1 Boundary Conformity

A tetrahedrization \mathcal{T} is said to *conform* to a PLC C if every vertex of C is a vertex of \mathcal{T} , every edge of C is the union of edges of \mathcal{T} , and every face of C is the union of faces of \mathcal{T} . A tetrahedrization \mathcal{T} is a *conforming Delaunay tetrahedrization* of C if and only if \mathcal{T} conforms to C and \mathcal{T} is also a Delaunay tetrahedrization of the set of vertices of \mathcal{T} .

Now, we can formally restate the problem we are interested in studying in this chapter: *Given a PLC C , find a conforming Delaunay tetrahedrization of C .* Recall that if we are given a conforming Delaunay tetrahedrization of C , we can eliminate the simplices of the tetrahedrization that lie outside the polyhedron whose boundary is described by C such that the underlying space of the resulting tetrahedrization matches the polyhedron. Of course, this is only possible if the PLC C describes the boundary of a polyhedron with possibly a set of isolated vertices, faces and edges inside it. For instance, the PLC in Figure 2.3(a) does not, while the PLCs in Figure 2.3(b) and Figure 2.3(c) do. The algorithms we describe later generates a conforming Delaunay tetrahedrization of the input PLC, but they do not provide any quality guarantee on the aspect ratio of the tetrahedra.

3.2 Existence and Complexity

We will see in Section 3.4 an incremental algorithm for the problem of constructing a conforming Delaunay tetrahedrization of any given PLC C . By proving the correctness and termination of this algorithm, we prove that any such a PLC C admits a tetrahedrization. Interestingly, there are PLCs whose tetrahedrization have necessarily more vertices than the vertices of the PLC. For instance, consider the Schönhardt polyhedron showed in Figure 3.1 [10]. It can be obtained from a triangular prism by a slight rotation of one triangular face relative to the other. After this rotation, each of the three square faces is broken along a diagonal into two triangular faces, and the polyhedron is no

longer convex, as the three diagonals are reflex. As a result of the rotation, the upper left corner and lower right corner of each formerly square face are no longer visible to each other. Since any four vertices of the polyhedron include the upper left corner and lower right corner of some formerly square face of the polyhedron, any tetrahedron whose vertices are vertices of the polyhedron will have an edge that does not lie entirely within the polyhedron. Therefore, the Schönhardt polyhedron cannot be tetrahedrized by using tetrahedra spanned by its vertices only. However, we can easily obtain a tetrahedrization of the Schönhardt polyhedron by adding a vertex to its center.

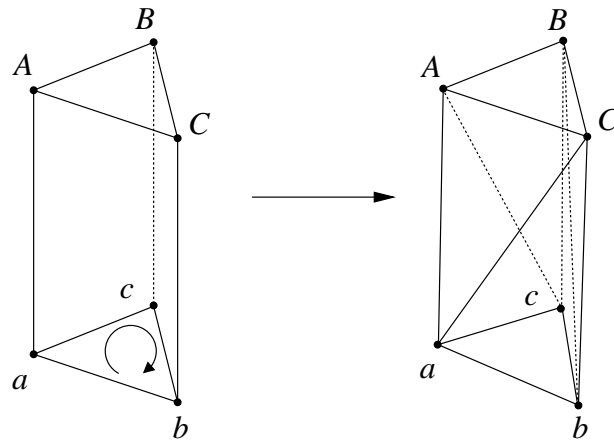


Figure 3.1: The Schönhardt polyhedron.

Ruppert and Seidel [13] built on the construction of the Schönhardt polyhedron and proved that the problem of deciding if a polyhedron can be tetrahedrized without inserting extra vertices is NP-hard. They also showed that the problem of deciding if a polyhedron can be tetrahedrized with only k additional vertices is NP-hard. However, Bern and Eppstein [14] showed that any polyhedron can be tetrahedrized by adding at most $O(n^2)$ extra vertices, where n is the number of vertices of the polyhedron. So, the demand for extra vertices is not unlimited. Bern and Eppstein also showed that any polyhedron can be tetrahedrized with $O(n^2)$ tetrahedra, and Chazelle [15] provided a polyhedron Q such that any tetrahedrization of Q has at least $(n + 1)^2$ tetrahedra.

Unfortunately, a PLC C can be much more complicated than a polyhedron as C may have isolated vertices, edges, and faces in the interior of its associated polyhedron. Besides, we are interested in a conforming Delaunay tetrahedrization of C as opposed to any tetrahedrization. Even if a PLC C exactly defines the boundary of a polyhedron, an upper bound for the number of extra vertices required to build a conforming Delaunay tetrahedrization of C is still unknown. Also, if we relax the conformity restriction and ask for a Delaunay tetrahedrization of C that conforms only to the vertices and edges of C , it is not known if we can add only $O(mn)$ extra vertices to build such a tetrahedrization, where m and n are the number of vertices and the number of edges of C , respectively, [4].

3.3 Delaunay Refinement

This section presents an algorithm for obtaining a conforming Delaunay triangulation of a restricted family of PLC: The PLCs satisfying the *angle constraint* described in Subsection 3.3.1. This algorithm is due to Jonathan Shewchuk [5].

3.3.1 Angle Constraint

Let C be a PLC. For the purpose of Shewchuk's algorithm, we measure the angle between two adjacent d -faces of C with $d \in \{1, 2\}$ as follows (see Figure 3.2):

1. Let $\sigma = [u, v]$ and $\tau = [v, w]$ be two edges of C such that $\sigma \cap \tau = \{v\}$. Then, *the angle between σ and τ* is $\min\{\angle uvw, \angle wvu\}$.
2. Let $\sigma = [u, v]$ be an edge of C and let τ be a facet of C such that $u \in \tau$. If σ and τ are not coplanar then let H be the plane through σ perpendicular to the supporting plane of τ . We define *the angle between σ and τ* to be $\min\{\angle puv \mid p \in (H \cap \text{int}(\tau))\}$, where $\text{int}(\tau)$ is the relative interior of τ . If $(H \cap \text{int}(\tau)) = \emptyset$ or if σ and τ are coplanar then the angle between σ and τ is undefined.
3. Let σ and τ be two facets of C such that $\sigma \cap \tau \neq \emptyset$. If σ and τ are not coplanar then let H_σ and H_τ be the supporting planes of σ and τ , respectively. For each point $p \in (H_\sigma \cap H_\tau)$, let H_p be the plane perpendicular to $H_\sigma \cap H_\tau$. We define *The angle between σ and τ* to be $\min\{\angle opq \mid p \in (H_\sigma \cap H_\tau), o \in (H_p \cap \text{int}(\sigma)), q \in (H_p \cap \text{int}(\tau))\}$. If $(H_p \cap \text{int}(\sigma)) = \emptyset$ and $(H_p \cap \text{int}(\tau)) = \emptyset$ for every $p \in (H_\sigma \cap H_\tau)$ or if σ and τ are coplanar then the angle between σ and τ is undefined.

A PLC C is said to satisfy *the angle constraint* if and only if the angle between any two adjacent d -faces of C , with $d \in \{1, 2\}$, is either undefined or equal to or greater than $\frac{\pi}{2}$.

3.3.2 Refinement Algorithm

Given a PLC C that satisfies the angle constraint as input, Shewchuk's algorithm produces a conforming Delaunay tetrahedrization of C . In the following we describe this algorithm. To avoid confusion between elements of C and those of the resulting tetrahedrization, we refer to the edges and 2-faces of C as *segments* and *facets*, respectively.

The first step of the algorithm builds a Delaunay tetrahedrization \mathcal{DT}_3 of the set of vertices of \mathcal{C} . Unless we are lucky, there will be segments of \mathcal{C} that are not covered by edges of \mathcal{DT}_3 , and there will be facets of \mathcal{C} that are not covered by triangles of \mathcal{DT}_3 . To recover these segments and facets, the algorithm adds new vertices to the set of vertices of \mathcal{DT}_3 and updates \mathcal{DT}_3 to include the new vertices.

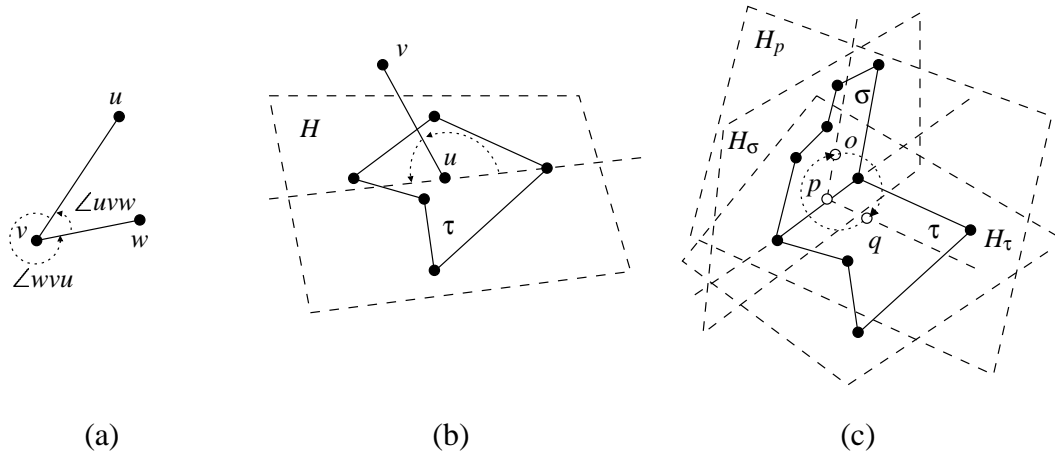


Figure 3.2: Angle between (a) two edges, (b) an edge and a facet, and (c) two facets.

The process of recovering missing edges and facets of \mathcal{C} necessarily decomposes each segment of \mathcal{C} into smaller edges called *subsegments*. Each bold edge of the tetrahedrization illustrated in Figure 3.3(b) is a subsegment of some segment of the input PLC shown in Figure 3.3(a); edges that are not bold are not subsegments. Similarly, each facet is decomposed into triangular faces called *sub-facets*. All of the triangles visible in Figure 3.3(b) are sub-facets, but most of the triangles in the interior of the tetrahedrization are not.

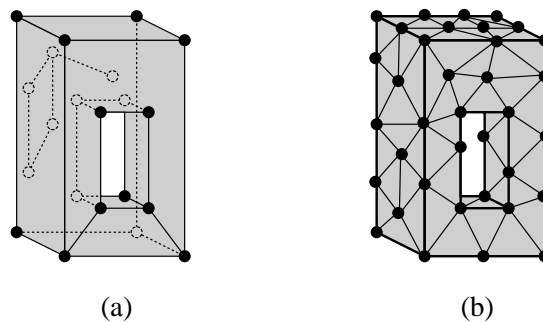


Figure 3.3: (a) An input PLC. (b) Its tetrahedrization.

The new vertices are added to the tetrahedrization according to three rules. Before describing them, we need a few definitions. The *diameter sphere* of a subsegment is the (unique) smallest sphere that encloses the subsegment. The

equator sphere of a triangular sub-facet is the (unique) smallest sphere that passes through the three vertices of the sub-facet. A tetrahedron of the Delaunay tetrahedrization is said to be *skinny* if its circumradius is equal to or greater than B times the length of its shortest edge, where B is a positive constant we will define later.

A vertex *encroaches upon* a subsegment if it is enclosed by the diameter sphere of that subsegment and it is not a vertex of the subsegment. Likewise, a vertex *encroaches upon* a sub-facet if it is enclosed by the equator sphere of that sub-facet and it is not a vertex of the sub-facet. It is a property of the Delaunay tetrahedrization that if a subsegment is missing from the tetrahedrization then it is encroached. Similarly, it is a property of the Delaunay tetrahedrization that if a sub-facet does not appear in the tetrahedrization nor is it covered by other faces that share the same equatorial sphere, then it is encroached. Note that an encroached subsegment or sub-facet may indeed be part of the tetrahedrization, as encroachment is a restriction stronger than the “empty sphere” one.

The algorithm starts by building the delaunay tetrahedrization \mathcal{DT}_3 of the set of vertices of C , and then recover the missing segments of C , if any. Note that if a segment is missing from \mathcal{DT}_3 then this segment is encroached upon some vertex of \mathcal{DT}_3 . So, to recover missing segments, the algorithm uses a list L_1 of subsegments, which is initialized with all segments of C , and then applies the following rule:

1. For any subsegment s of L_1 such that s is encroached upon some vertex of \mathcal{DT}_3 , insert the midpoint of s into the set of points of \mathcal{DT}_3 , remove s from L_1 and insert the subsegments s_1 and s_2 into L_1 , where s_1 and s_2 are the subsegments resulting from the splitting s at its midpoint. Repeat this rule until no subsegment in L_1 is encroached upon any vertex of \mathcal{DT}_3 .

We are assured of the eventual termination of the splitting process of rule 1 because the Delaunay tetrahedrization always connects a vertex to its nearest neighbor. So, once the distance between consecutive vertices along a segment of C is sufficiently small, this segment will be in the tetrahedrization. After recovering every missing segment of C , the algorithm starts recovering its missing facets. Note that at this point the boundary segments of the all facets of the input PLC C are subdivided by edges of the Delaunay tetrahedrization.

Unlike subsegments, it is difficult to know what the sub-facets of a particular facet are as they do not exist yet. One way of going around this problem is to maintain a two-dimensional Delaunay triangulation of the set of vertices of each facet. That is, for every facet f of C , the algorithm builds the Delaunay triangulation $\mathcal{DT}_2(P \cap Q)$, where P is the set of vertices of the tetrahedrization and Q is the supporting plane of f . The Delaunay triangulation $\mathcal{DT}_2(P \cap Q)$ is maintained by the algorithm independently of the tetrahedrization in which we hope the missing facets of $\mathcal{DT}_2(P \cap Q)$ will eventually appear. To recover missing facets, the algorithm applies the following rule:

2. For each facet f of C , consider $\mathcal{DT}_2(P \cap Q)$. If there is a sub-facet f' in $\mathcal{DT}_2(P \cap Q)$ that is encroached upon by a vertex of \mathcal{DT}_3 then inserts the circumcenter (the center of the equator sphere) of f' into the set of points of \mathcal{DT}_3 , and update \mathcal{DT}_3 and $\mathcal{DT}_2(P \cap Q)$. However, if the circumcenter of f' encroaches upon any subsegment in L_1 , do not insert this circumcenter; instead, apply rule 1 again as the circumcenter had been inserted. Repeat this rule until no sub-facet of $\mathcal{DT}_2(P \cap Q)$ is encroached upon by a vertex of \mathcal{DT}_3 .

It can be shown that if the circumcenter of a sub-facet is outside the facet then this circumcenter encroaches upon a segment of the facet. Since the algorithm does not insert the circumcenter of a sub-facet if it encroaches upon a segment, all circumcenters are inserted inside the facet. Furthermore, note that the insertion of the circumradius of a sub-facet f' of $\mathcal{DT}_2(P \cap Q)$ into \mathcal{DT}_3 will remove f' from $\mathcal{DT}_2(P \cap Q)$. Note that rule 2 always terminates as it demands the insertion of a finite amount of vertices (either the circumcenter of a sub-facet or the midpoints of the subsegments encroached by it). When there is no encroached subsegment in L_1 and there is no encroached sub-facet in the Delaunay triangulation of each facet of C , the algorithm removes all exterior tetrahedra (lying in the convex hull of the input vertices, but outside the region enclosed by C) from \mathcal{DT}_3 .

3.3.3 Termination, Correctness and Complexity

Throughout this subsection, assume that C is a PLC satisfying the angle constraint, and \mathcal{DT}_3 is the Delaunay tetrahedrization of Shewchuk's algorithm on input C at any given time. We start the description of the termination proof of Shewchuk's algorithm by stating an important lemma that holds only if the input PLC C satisfies the angle constraint.

Given a facet or sub-facet f and a point v , the orthogonal projection $\text{proj}_f(v)$ of v onto f is the point that is coplanar with f and lies in the line that passes through v orthogonally to f . The projection exists whether or not it falls in f . Similarly, the orthogonal projection $\text{proj}_s(v)$ of v onto a segment or subsegment s is the point that is collinear with s and lies in the plane through v orthogonal to s .

Lemma 3.3.1 (Projection Lemma). *Let f' be a sub-facet of the Delaunay triangulation of the set of points lying on the supporting plane of the facet f . Suppose that f' is encroached upon by some vertex v , but v does not encroach upon any subsegment of f . Then $\text{proj}_f(v)$ lies in the facet f , and v encroaches upon a sub-facet of f that contains $\text{proj}_f(v)$.*

The importance of the Projection Lemma is two-folded. First, it assures us that if a vertex v encroaches upon a sub-facet of a facet f but f does not contain $\text{proj}_f(v)$ then v has to encroach upon some boundary segment of f . Be-

cause the algorithm eliminates encroached subsegments before eliminating encroached sub-facets, all subsegments encroached upon by v are split until none of them remain and the Projection Lemma also ensures us that all sub-facets of f encroached upon by v are eliminated in this process. Second, the Projection Lemma assures us that if a vertex v encroaches upon more than one sub-facet of a facet f then one of the encroached sub-facets, say f' , contains $\text{proj}_f(v)$. The algorithm should apply rule 2) to f' . The reason for that is that the distance between v and the nearest vertex of f' is bounded by $\sqrt{2}$ times the radius of the equator sphere of f' , and this bound makes it possible the existence of a stronger guarantee of quality for the output tetrahedrization.

For each vertex v of \mathcal{DT}_3 , we define the *insertion radius* r_v of v as follows. If v is a vertex of C then r_v is the minimum distance from v to another vertex of C . If v is a vertex inserted into \mathcal{DT}_3 or rejected by Shewchuk's algorithm then r_v is the minimum distance to a vertex in \mathcal{DT}_3 at the time when v is inserted or rejected. Recall that a vertex v is rejected by Shewchuk's algorithm if v is the circumcenter of an encroached sub-facet, and rule 2 is applied to insert v but v encroaches upon a subsegment. As a result the vertex v cannot be inserted into the set of vertices of \mathcal{DT}_3 .

Consider the time when Shewchuk's algorithm inserts or rejects a vertex v . This can happen only by the application of either rule 1 or rule 2. We say that the vertex v *has type 1* if it is inserted by rule 1, and we say that the vertex v *has type 2* if it is inserted or rejected by rule 2. Let v be a vertex inserted or rejected by the algorithm. We define the *parent* p of v as the vertex responsible for the attempt of insertion of v , i.e., the vertex v is considered for insertion only if some subsegment s' or sub-facet f' is encroached upon by at least one vertex in \mathcal{DT}_3 . Let V be the set of encroaching vertices upon s' or f' . Then, the parent p of v is a vertex $p \in V$ such that $d(p, v) \leq d(q, v)$ for all $q \in V$. Finally, if v is a vertex of C then the parent of v is undefined.

Lemma 3.3.2. *Let C be a PLC satisfying the angle constraint, and let \mathcal{DT}_3 be the Delaunay tetrahedrization of Shewchuk's algorithm on input C at any given time. If v is a vertex of \mathcal{DT}_3 then the following hold:*

1. *If $v \in C$ then $r_v \geq lfs(v)$, where r_v is the insertion radius of v and $lfs(v)$ is the value of the local feature size function at v .*
2. *If v has type 1 or type 2 and the parent p of v is a vertex of C then $r_v \geq lfs(p)$, where r_v is the insertion radius of v and $lfs(p)$ is the value of the local feature size function at p .*
3. *If v has type 1 or type 2 and the parent p of v has been rejected by the algorithm then $r_v \geq \frac{1}{\sqrt{2}} \cdot r_p$, where r_v and r_p are the insertion radius of v and p , respectively.*
4. *If v has type 1 or type 2 and the parent p of v has been inserted by the algorithm then $r_v \geq lfs(p)$, where r_v is*

the insertion radius of v and $lfs(p)$ is the value of the local feature size function at p .

Proof. If v is a vertex of C then, by definition, $lfs(v)$ is less than or equal to the distance from v to the nearest other vertex of C . Hence, $r_v \geq lfs(v)$ and claim 1 holds. If v is not a vertex of C then v is a vertex of type 1 or type 2 inserted or rejected by the algorithm. Thus, the vertex v has a parent p . If p is a vertex of C then it is not contained in the same segment or facet that contains v . Thus, $r_v = d(v, p) \geq lfs(p)$ and claim 2 holds. If p is not a vertex of C then p encroaches upon a subsegment s' whose v is the midpoint or upon a sub-facet f' whose v is the circumcenter. In the former case, the vertex p is inside the diameter sphere of s' . In the latter case, the vertex p is inside the equator sphere of f' . In either case, the distance $d(p, q)$ from p to the closest vertex q of s' or f' is at most $\sqrt{2}$ times the distance $d(v, q)$ of v from the same vertex. If p was rejected by the algorithm then $r_p \leq d(p, q)$ and $r_v = d(v, q)$. Hence, we have that $r_v \geq \frac{1}{\sqrt{2}} \cdot r_p$ and claim 3 holds. If p was inserted into \mathcal{DT}_3 by the algorithm then p is either the midpoint of an encroached segment s'' or the circumcenter of an encroached sub-facet f'' (which is possible only if v is the circumcenter of an encroached sub-facet). If v is of type 1 then p is of type 1 (midpoint as of an encroached segment s'') and, from the fact that C satisfies the angle constraint, the segment containing p cannot be adjacent to the segment containing v . If v is of type 2 then p is of type 1 (midpoint as of an encroached segment s'') or type 2 (circumcenter of an encroached sub-facet f'') and, from the fact that C satisfies the angle constraint and from the Projection Lemma, the segment or facet containing p cannot be adjacent to the segment containing v . So, in either case, the vertices v and p must be in non-incident faces of C . Hence, $r_v = d(v, p) \geq lfs(p)$ and claim 4 holds. \square

Lemma 3.3.2 limits how quickly the insertion radius can decrease as the algorithm inserts new vertices. If vertices with ever-smaller insertion radii cannot be generated by the algorithm then edges shorter than existing ones cannot be introduced, and consequently the algorithm must terminate. Before formally proving this fact, we need one more result.

Theorem 3.3.1. *Let C be a PLC satisfying the angle constraint, and let c be the minimum distance between any two disjoint faces of C . Then, Shewchuk's algorithm on input C will terminate and generate a conforming Delaunay tetrahedrization of C with no edge shorter than $\frac{1}{\sqrt{2}} \cdot c$.*

Proof. Let us proceed by contradiction. Assume that the algorithm introduces one or more edges shorter than $\frac{1}{\sqrt{2}} \cdot c$ into the tetrahedrization. Let e be the first such edge introduced by the algorithm. Clearly, the endpoints of e cannot both be vertices of C nor can they lie on non-incident feature points. Let v be the most recently inserted endpoint of e , let p be the parent of v . From the fact that C satisfies the angle constraint and from the Projection Lemma, if v is the midpoint of a segment s then p cannot be in a segment or facet adjacent to s , and if v is the circumcenter of a

sub-facet of a facet f then p cannot be in a facet adjacent to f . Furthermore, if p was inserted by the algorithm and v and p lie on non-incident segments or facets of \mathcal{C} then, from Lemma 3.3.2, $r_v \geq lfs(p) \geq c$. So, we are left with the following cases:

1. The vertex v is the midpoint of an encroached subsegment or the circumcenter of an encroached facet and p is a vertex of \mathcal{C} .
2. The vertex v is the midpoint of an encroached subsegment s' and p is the (rejected) circumcenter of an encroached sub-facet f' .
3. The vertex v is the midpoint of an encroached sub-facet f' and p is not a vertex of \mathcal{C} .

In case 1, since p is a vertex of \mathcal{C} , Lemma 3.3.2 assures us that $r_v \geq lfs(p)$. But, by hypothesis, $lfs(p) \geq c$. Hence, $r_v \geq c$. In case 2, we can use Lemma 3.3.2 to claim that $r_v \geq \frac{1}{\sqrt{2}} \cdot r_p$. Since p was rejected by the algorithm, it also has a parent, say the vertex g , which is either a vertex of \mathcal{C} or a vertex inserted by the algorithm. In either case, the vertices p and g do not belong to the same facet. So, from Lemma 3.3.2, we have that $r_p \geq lfs(g)$. But, by hypothesis, $lfs(g) \geq c$. Hence, $r_v \geq \frac{1}{\sqrt{2}} \cdot r_p \geq \frac{1}{\sqrt{2}} \cdot lfs(g) \geq \frac{1}{\sqrt{2}} \cdot c$. In case 3, the vertex p is the circumcenter of a sub-facet in a facet non-incident to the one containing f' or the vertex p is the midpoint of a subsegment that may be adjacent to the containing f' . In both cases, Lemma 3.3.2 assures us that $r_v \geq lfs(p)$. So, in all cases above, we have that $r_v \geq \frac{1}{\sqrt{2}} \cdot c$. Thus, the length of the edge e cannot be smaller than c , which is a contradiction. The correctness of the algorithm follows from its termination and from the correctness of the algorithms for constructing Delaunay triangulation and tetrahedrization of a set of points. \square

As we may expect, the size complexity of the conforming Delaunay tetrahedrization of Shewchuk's algorithm might not be polynomial with respect to the size of the input PLC \mathcal{C} . After building the initial Delaunay tetrahedrization, the theoretical worst-case running time of Shewchuk's algorithm is $O(n^2)$ time per vertex insertion. However, as Shewchuk points out [5], he was not able to devise an input PLC for which the algorithm would present its worst behavior. In addition, in practice, he noticed that after a few initial insertions of vertices, further insertions tend to be well-localized, which makes it possible to achieve constant time per insertion.

3.4 PLCs with Small Angles

This section describes an algorithm for constructing conforming Delaunay tetrahedrizations of PLCs that may not satisfy the angle constraint. This algorithm is due to Cohen-Steiner, Verdière, and Yvinec [7]. Their algorithm uses a scheme to “protect” the edges of the input PLC, called *protecting balls*, that guarantees the termination of the algorithm even if the input PLC does not satisfy the angle constraint. The underlying idea of this edge protection scheme can be understood by studying the behavior of Shewchuk’s algorithm on an input PLC that does not satisfy the angle constraint.

3.4.1 Small Angles

What could go wrong with Shewchuk’s algorithm if the input PLC does not satisfy the angle constraint? Consider the two adjacent segments \overline{av} and \overline{au} in Figure 3.4. The angle between them is less than $\frac{\pi}{2}$. Initially, vertex v encroaches upon the segment \overline{au} , which is split at its midpoint w . Vertex w encroaches upon the segment \overline{av} , which is split at its midpoint x , and so on. This “chain reaction” of mutual encroachments may produce ever-smaller subsegments incident to the vertex a , and causes the algorithm to run forever.

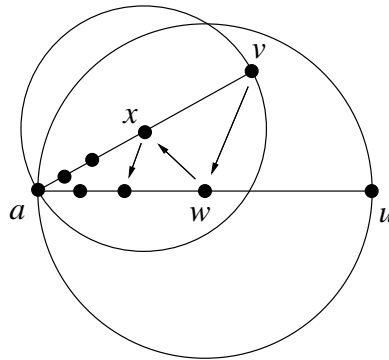


Figure 3.4: Example of a “chain reaction” of mutual encroachments.

Note that the chain reaction of mutual encroachments described in Figure 3.4 involves two adjacent segments of \mathcal{C} . If the angle constraint is satisfied, the Projection Lemma holds, and consequently vertices in a segment of \mathcal{C} cannot encroach upon subsegments and sub-facets of adjacent segments or facets, and vertices in a facet of \mathcal{C} cannot encroach upon sub-facets of adjacent facets. Hence, a situation like the one in Figure 3.4 cannot happen if the angle constraint is satisfied. In addition, Lemma 3.3.2 and Theorem 3.3.1 assure that chain reaction of mutual encroachments cannot occur between two d -faces ($d \in \{1, 2\}$) of \mathcal{C} that are disjoint. These observations led to the

development of the protecting balls scheme.

3.4.2 Protecting Balls

Given a PLC \mathcal{C} , we employ the protecting ball scheme to *conceptually* divide the domain of each facet f of \mathcal{C} into two disjoint regions: The protected region and the unprotected region of f . The protected region contains the boundary edges of f , and the unprotected region of f is disjoint from the unprotected region of any other facet of \mathcal{C} . As we shall see later, after the insertion of a few special vertices in the protected regions of the input PLC, the conforming Delaunay tetrahedrization algorithm in [7] only allows the insertion of new vertices inside and on the boundary of unprotected regions. Since unprotected regions are disjoint from each other, vertices inserted in unprotected regions cannot cause a chain reaction of mutual encroachments.

Unfortunately, the protecting balls scheme is restricted to another particular class of PLCs: The ones for which every vertex is an endpoint of an edge, and every edge belongs to the boundary of at least one facet of the PLC. Of course, this restriction is also a limitation of the tetrahedrization algorithm in [7]. However, in practice, most of the domains of interest do not contain isolated vertices and edges lying in the interior of facets or outside them.

Let \mathcal{C} be a PLC such that every vertex is an endpoint of an edge and every edge of \mathcal{C} belongs to the boundary of at least one facet of \mathcal{C} , and let \mathcal{B} be a set of closed balls, called *protecting balls*, satisfying the following conditions:

1. The union of the balls in \mathcal{B} covers the 1-skeleton of \mathcal{C} .
2. The balls are centered on points which are in the 1-skeleton of \mathcal{C} .
3. If two balls intersect then their centers belong to the same edge of \mathcal{C} .
4. if a face of \mathcal{C} intersects a ball in \mathcal{B} then it contains the center of this ball.
5. The intersection of any three balls in \mathcal{B} is empty.
6. Any two balls are not tangent.
7. The center of any ball is inside no other ball.

Note that conditions 1 and 4 imply that any vertex in \mathcal{C} is the center of a ball in \mathcal{B} . An algorithm for constructing \mathcal{B} is given in [7].

Let B be a ball of \mathcal{B} with center v . The point v is called a c -point. Let V be the set of balls in \mathcal{B} that intersect B . By condition 5, the intersections of B with the elements of V are disjoint. For each element B_i of V , consider the radical plane H of B and B_i . The plane H intersects the line passing through the centers of B and B_i at a point h_i . The point h_i is on an edge of C by condition 3, and it is called an h -point. By condition 4, any face of C that intersects $B \cap B_i$ contains the centers of B and B_i , and thus it can be either the edge of C including the segment $\overline{vv_i}$, where v_i is the center of B_i , or a facet incident to this edge. For each facet f of C intersecting $B \cap B_i$, the intersection points between the circle $bd(B) \cap bd(B_i)$ and f are called p -points. Figure 3.5 shows c -points, h -points, and p -points in a neighborhood of the a ball B incident to three other balls.

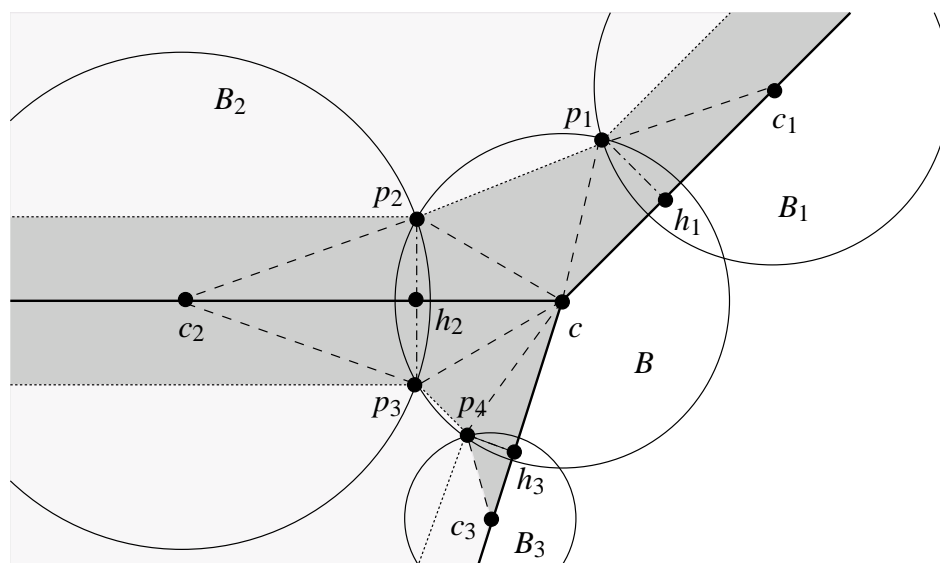


Figure 3.5: The neighborhood of a protecting ball.

Let Q be the supporting plane of a facet f of C intersecting B . Note that the center v of B belongs to Q . The edges of C incident to f split the disk $Q \cap B$ into one or more sectors. We are interested in the sectors included in f only. As illustrated by Figure 3.5, these sectors are further subdivided into sub-sectors by the p -points on the boundary of B . We call *right-angled sub-sectors* the sub-sectors delimited by an edge of C and a p -point, and *isosceles sub-sectors* the sub-sectors limited by two p -points.

Let $\mathcal{S}(C, \mathcal{B})$ be the collection of all right-angled and isosceles sub-sectors of C with respect to \mathcal{B} . For each isosceles sub-sector avb of $\mathcal{S}(C, \mathcal{B})$, where v is the center of the ball B in \mathcal{B} containing avb and a, b are the points of avb on the boundary of B , we define the *angle of avb* as the angle formed by avb at v . Only isosceles sub-sectors whose angle is less than $\frac{\pi}{2}$ are desirable for the conforming Delaunay tetrahedrization algorithm we will describe next. So, if an isosceles sub-sector of $\mathcal{S}(C, \mathcal{B})$ has an angle equal to or greater than $\frac{\pi}{2}$, we replace it by two or more isosceles

sub-sectors whose angles are less than $\frac{\pi}{2}$ using a procedure called *split-on-a-sphere* as follows.

Let avb be an isosceles sub-sector of $\mathcal{S}(C, \mathcal{B})$, and let B be the ball to which avb belong. If a and b belong to only one ball B of \mathcal{B} then we define c to be the midpoint of the shortest geodesic arc \widehat{ab} on $bd(B)$. Otherwise, either a or b is a p -point and belongs to $bd(B) \cap bd(B_i)$ for some $B_i \in \mathcal{B}$. Without loss of generality, assume that a is a p -point. Then, we compute c using the *concentric shell splitting*¹. We define d to be the value of $d(a, b)$ divided by 2 and rounded to the nearest power of 2, 2^k , $k \in \mathbb{Z}$, and we define c to be the point of the shortest geodesic arc \widehat{ab} on $bd(B)$ at a distance d from a . Finally, remove avb from $\mathcal{S}(C, \mathcal{B})$ and add the isosceles sectors avc and cvb to $\mathcal{S}(C, \mathcal{B})$. The point c is called a *sos-point*. Figure 3.6 illustrates the creation of the isosceles sub-sectors avc and cvb by the split-on-a-sphere procedure.

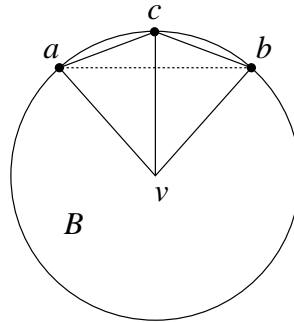


Figure 3.6: The procedure split-on-a-sphere creates the isosceles sub-sectors avc and cvb from the isosceles sub-sector avb .

By using the split-on-a-sphere procedure, we can replace every isosceles sub-sector avb of $\mathcal{S}(C, \mathcal{B})$ whose angle is equal to or greater than $\frac{\pi}{2}$ with two or more isosceles sub-sectors whose angles are less than $\frac{\pi}{2}$. In order to do that, we execute split-on-a-sphere on the isosceles sub-sectors of $\mathcal{S}(C, \mathcal{B})$ whose angle is equal to or greater than $\frac{\pi}{2}$ until every isosceles sub-sectors of $\mathcal{S}(C, \mathcal{B})$ has an angle less than $\frac{\pi}{2}$. From now on, we assume that $\mathcal{S}(C, \mathcal{B})$ contains isosceles sub-sectors whose angle is less than $\frac{\pi}{2}$ only.

Let avb be an isosceles sub-sector of $\mathcal{S}(C, \mathcal{B})$. The line segment \overline{ab} joining the points a and b of avb is called a *shield edge*. We associate with each isosceles sub-sector avb in $\mathcal{S}(C, \mathcal{B})$ the isosceles triangle $[a, v, b]$ defined by the c -point v and the shield edge \overline{ab} . Similarly, we associate with each right-angled sub-sector avb in $\mathcal{S}(C, \mathcal{B})$ the right triangle $[a, v, b]$ spanned by a , v , and b , where a is the h -point, v is the c -point, and b is the p -point of the right-angled sub-sector avb . We define the *protected region of C with respect to \mathcal{B}* as the union of the isosceles and right triangles associated with the isosceles and right-angled sub-sectors of $\mathcal{S}(C, \mathcal{B})$. The *unprotected region of C with respect to*

¹The reason for using the concentric shell splitting will be clear in the proof of Lemma 3.4.6.

\mathcal{B} is the complex \mathcal{C} minus the protected region. Figure 3.5 illustrates both concepts.

3.4.3 Refinement Algorithm

Let \mathcal{C} be a PLC such that every vertex is an endpoint of an edge, and every edge belongs to the boundary of at least one facet of the PLC. Given such a PLC as input, the first step of the algorithm is to build a set \mathcal{B} of protecting balls for \mathcal{C} that satisfies the seven conditions in Subsection 3.4.2. For the sake of space, we do not describe this step here. A detailed description of this step can be found in [7]. The second step of the algorithm consists of creating a set P of vertices containing all c -points, h -points, p -points, and sos -points of the collection $\mathcal{S}(\mathcal{C}, \mathcal{B})$ of all right-angled and isosceles sub-sectors of \mathcal{C} with respect to \mathcal{B} after the execution of the split-on-a-sphere procedure. By definition of \mathcal{C} and \mathcal{B} , the set P contains all vertices of \mathcal{C} .

The third step is known as the *protection procedure*. This procedure ensures that shield edges and isosceles triangles associated with isosceles sub-sectors of $\mathcal{S}(\mathcal{C}, \mathcal{B})$ are not encroached upon by any vertex of P . Like before, we say that a *shield edge* \overline{ab} is *encroached upon* by a vertex $v \in P$ if v belongs to the interior of the diameter sphere of \overline{ab} , and we say that an *isosceles triangle* avb is *encroached upon* by a vertex $v \in P$ if v belongs to the interior of the equator sphere of avb . The procedure works as follows: While there is an encroached shield edge \overline{ab} or an encroached isosceles triangle $[a, v, b]$, where avb is an isosceles sub-sector of $\mathcal{S}(\mathcal{C}, \mathcal{B})$, execute the split-on-a-sphere procedure on avb , and insert the new vertex generated by this procedure into P .

The first, second, and third steps comprise the initialization stage of the algorithm. The rest of the algorithm is as follows. For each facet f of \mathcal{C} , we consider the supporting plane Q of f and build the Delaunay triangulation $\mathcal{DT}_2(P \cap Q)$ of the point set P that lie in Q , and we also build the Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of P . Because the initialization stage ensures that no shield edge \overline{ab} in $\mathcal{S}(\mathcal{C}, \mathcal{B})$ is encroached upon by a vertex in P , the Delaunay triangulation $\mathcal{DT}_2(P \cap Q)$ contains all shield edges \overline{ab} in $\mathcal{S}(\mathcal{C}, \mathcal{B})$ such that a and b are in Q . As a result the Delaunay triangulation $\mathcal{DT}_2(P \cap Q)$ conforms to the unprotected region f_u of f for every $f \in \mathcal{C}$. So, we are left with the task to ensure that every triangle in $\mathcal{DT}_2(P \cap Q)$ included in f_u appears in $\mathcal{DT}_3(P)$. This is done by the *main procedure* as follows.

While there is a triangle (sub-facet) t in $\mathcal{DT}_2(P \cap Q)$, where Q is the supporting plane of a facet f of \mathcal{C} , such that t is included in the unprotected region f_u of f and t does not appear in $\mathcal{DT}_3(P)$, do the following:

1. If the circumcenter v of t does not encroach upon any shield edge in $\mathcal{S}(\mathcal{C}, \mathcal{B})$, insert v into P and update $\mathcal{DT}_2(P \cap Q)$ and $\mathcal{DT}_3(P)$ accordingly.

2. Otherwise, do not insert v into P ; instead, apply the split-on-a-sphere procedure to each isosceles sub-sector whose shield edge is encroached upon by v , and then execute the third step of the algorithm again (the protection procedure).

Note that the execution of the split-on-a-sphere and protection procedures in the statement 2 above causes the unprotected region f_u to shrink. Note also that if we view the collection of unprotected regions of C as the input PLC of Shewchuk's algorithm, the statements 1 and 2 above are very similar to rules 1 and 2 of Shewchuk's algorithm. The only difference is the fact that the unprotected regions may shrink, while facets do not. As we prove in the next subsection, the main procedure is guaranteed to terminate and it also ensures that $\mathcal{DT}_3(P)$ conforms to the protected region of every facet f of C , and thus $\mathcal{DT}_3(P)$ is a conforming Delaunay tetrahedrization of C .

3.4.4 Termination and Correctness

We start proving the termination of the conforming Delaunay tetrahedrization algorithm in Subsection 3.4.3 by stating some properties of the algorithm.

Lemma 3.4.1. [7] *At the beginning and the end of the execution of the main procedure, there is no encroached shield edge in $\mathcal{S}(C, \mathcal{B})$.*

Proof. After the execution of the protection procedure, there cannot be any encroached shield edge in $\mathcal{S}(C, \mathcal{B})$. Since this procedure is executed in the initialization stage, there cannot be any encroached shield edge before the execution of the main procedure. In each loop of the main procedure, either statement 1 or statement 2 is executed. Statement 1 does not cause a shield edge encroachment, and statement 2 executes the protection procedure again. So, the desired result follows. \square

Lemma 3.4.2. [7] *Every circumcenter (inserted or rejected) by the algorithm lies in the unprotected region, outside the protecting balls.*

Proof. Let t be a triangle (sub-facet) of a facet f of C , and assume that the circumcenter v of t is considered for insertion at some step of the algorithm. The triangle t lies in the unprotected region f_u of f , and belongs to the Delaunay triangulation $\mathcal{DT}_2(P \cap Q)$ of $P \cap Q$, where Q is the supporting plane of f . Assume that v lies outside f_u , and consider any point m in t . Since f_u is enclosed by shield edges, the line segment \overline{vm} must intersect a shield edge \overline{ab} enclosing f_u . Because t belongs to $\mathcal{DT}_2(P \cap Q)$, its circumsphere is empty and thus it cannot contain a

and b in its interior. But, in this case, the diameter sphere of \overline{ab} contains t as v is inside this diameter sphere, and consequently m encroaches upon \overline{ab} , which is a contradiction. So, the vertex v must lie in f_u . \square

Corollary 3.4.1. *No vertex is inserted inside the protecting balls after the initialization step of the algorithm.*

Proof. This follows from Lemma 3.4.2 and from the fact that the diameter sphere of shield edges cover the intersection of the unprotected region with the protecting balls. \square

Lemma 3.4.3. [7] *At the beginning and the end of the execution of the main procedure, no isosceles triangle associated with an isosceles sub-sector in $\mathcal{S}(C, \mathcal{B})$ is encroached.*

Proof. Due to the protection procedure, our claim is certainly true after the initialization step and after the execution of statement 2 of the main procedure. So, it remains to show that our claim also holds after the execution of statement 1 of the main procedure. Let p be the circumcenter of some triangle t such that p is inserted into P by statement 1. Suppose that p encroaches upon an isosceles triangle $[a, v, b]$ associated with an isosceles sub-sector avb in $\mathcal{S}(C, \mathcal{B})$. By hypothesis, the vertex p does not encroach upon any shield edge in $\mathcal{S}(C, \mathcal{B})$. So, the vertex p does not belong to the diameter sphere of the shield edge \overline{ab} . From Corollary 3.4.1, the vertex p does not belong to the protecting ball B whose center is v . Because the angle of avb is less than $\frac{\pi}{2}$, the equator sphere of $[a, v, b]$ is contained in the union of the diameter sphere of \overline{ab} and B . So, the vertex v cannot belong to the equator sphere of $[a, v, b]$, which is a contradiction. Hence, our claim holds. \square

Lemma 3.4.4. [7] *Let B be a ball with center v , and let p be a point on the boundary of B . Then, if the line segment \overline{vp} is encroached upon during any step of the algorithm, the encroaching point is a h -point h_i on the radical plane of B and B_i , and p belongs to $bd(B) \cap int(B_i)$.*

Proof. Because the diameter sphere of \overline{vp} is inside B , the line segment \overline{vp} can only be encroached upon by a vertex in B that is not the center v of B . From the description of the algorithm and from Corollary 3.4.1, the only vertices inside B other than its center are h -points. Suppose that \overline{vp} is encroached upon by a h -point h_i belonging to B and B_i . So, the diameter sphere of \overline{vp} contains h_i in its interior. As a result we have that the angle $\angle vh_i p$ is greater than $\frac{\pi}{2}$. But, every point $q \in bd(B)$ satisfying $\angle vh_i p > \frac{\pi}{2}$ lies in $bd(B) \cap int(B_i)$. Thus, the point p is in $int(B_i)$. \square

Lemma 3.4.5. [7] *At any step of the algorithm, no right triangle associated with a right-angled sub-sector in $\mathcal{S}(C, \mathcal{B})$ is encroached.*

Proof. Suppose that the right triangle $[h_i, v, p]$ associated with the right-angled sub-sector $h_i v p$ of $\mathcal{S}(\mathcal{C}, \mathcal{B})$ is encroached at some step of the algorithm. By definition of a right-angled sub-sector, the h -point h_i is on the radical plane of two balls, B and B_i , the c -point v is the center of B , and the vertex p is either a p -point or a sos -point on the boundary of B and B_i . Since the diameter sphere of the line segment \overline{vp} is the same as the equator sphere of $[h_i, v, p]$, by Lemma 3.4.4, the encroaching point is a h -point and p has to belong to the interior of a third ball B_j , which is impossible by the definition of \mathcal{B} . Hence, our claim follows. \square

Note that Lemma 3.4.3 and Lemma 3.4.5 imply that the Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of P conforms to the protected region of \mathcal{C} . So, if the algorithm ends, the main procedure ensures that $\mathcal{DT}_3(P)$ also conforms to the unprotected region of \mathcal{C} . By assuming that the algorithms for constructing Delaunay triangulation and Delaunay tetrahedrization of a set of points are correct, the proof of correctness of the algorithm in Subsection 3.4.3 follows from the proof of termination. The termination of this algorithm depends on the termination of the protection and main procedures. If we assume that the protection procedure always terminates, we can prove that the main procedure always terminates by showing that the algorithm can only insert finitely many vertices in the unprotected region of \mathcal{C} .

Lemma 3.4.6. [7] *The protection procedure always terminates.*

Proof. We just sketch the proof here. See [7] for details. The idea behind this proof is to show that, for each call of the protection procedure, there exists an angle $\phi > 0$ such that no isosceles triangle $[a, v, b]$ whose angle $\angle avb$ is less than ϕ can be encroached, where v is the center of a ball B and a and b lie in $bd(B)$. If such an angle $\phi > 0$ exists, the protection procedure will call the split-on-a-sphere procedure finitely many times. As a result it always terminates. Suppose that the isosceles triangle $[a, v, b]$ associated with the isosceles sub-sector avb is encroached upon by a vertex p . Let B be the ball with center at v (and to which $[a, v, b]$ belongs), and let Q be the supporting plane of the facet of \mathcal{C} in which $[a, v, b]$ lies. In order to prove the existence of ϕ , we consider three possible cases: (1) the point p is on the boundary of B , (2) the point p is in the interior of B , and (3) the point p is outside B . In case (1), we can show that p can only be a vertex of an isosceles triangle that does not lie in Q . So, the distance d from p to the shield edge \overline{ab} is strictly positive and bounded from below. Thus, there exists $\phi > 0$ depending on d such that if $\angle avb < \phi$, the equator sphere of $[a, v, b]$ cannot contain p . In case (2), we know that p can only be a h -point. By using an argument similar to the one in the proof of Lemma 3.4.5, we can show that p cannot lie in Q . Hence, the vertex p is a h -point lying in a plane adjacent to Q . In this case, we can show that the distance from p to the arcs of any isosceles sub-sector in the plane Q is strictly positive and bounded from below. So, by an argument similar to the one in case 1, we show that there exists $\phi > 0$ such that if $\angle avb < \phi$, the equator sphere of $[a, v, b]$ cannot contain

p . In case (3), since p is outside B , if p encroaches upon $[a, v, b]$ then it also encroaches upon \overline{ab} . At each call of the protection procedure, the set of points outside the protecting balls is fixed. In addition, the distance between points on the boundary of distinct balls that do not share a p -point is strictly positive and bounded from below. Thus, there exists $\phi > 0$ such that if $\angle avb < \phi$, the diameter sphere of \overline{ab} cannot contain p , except if p belongs to Q and to the boundary of a ball B_i that intersects B , and a (or b) is one of the two points in $bd(B) \cap bd(B_i) \cap Q$. But, in this case, due to the use of concentric shell splitting by the split-on-a-sphere procedure, after a few splits, the edges incident to a will have the same length and will be unable to encroach upon each other. \square

Theorem 3.4.1. *The algorithm always terminates and when it reaches this state, the Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of P conforms to the input PLC C .*

Proof. Lemma 3.4.3 and Lemma 3.4.5 ensure that the Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of P conforms to the protected region of C . From Lemma 3.4.6, we know that the protection procedure always terminates. So, it remains to show that the main procedure always terminates. We prove the termination of the main procedure by proving that the algorithm can only insert finitely many circumcenters and that there are only a finite number of shield edges encroached upon by circumcenters rejected by the algorithm. Both claims can be proved by realizing that the unprotected region is a disjoint union of planar regions. Let f_u be such a region. Although f_u may evolve during the execution of the algorithm, it always shrinks. Consequently, the distance between f_u and the other regions as well as the distance between f_u and the set of c -points and h -points inserted into the protected regions can be bounded from below by a constant $\delta_f > 0$. Let t be a triangle (sub-facet) in f_u whose circumcenter is considered for insertion in P by the algorithm. Then, the triangle t belongs to $\mathcal{DT}_2(P \cap Q)$, where Q is the supporting plane of f_u , but it does not belong to $\mathcal{DT}_3(P)$, and therefore there exists at least one vertex p in P such that p is inside the equator sphere S_t of t . Such a point can be inside a protecting ball (a c -point or a h -point), on the boundary of a protecting ball (and thus on the boundary of another region as $t \in \mathcal{DT}_2(P \cap Q)$), or an added circumcenter in another unprotected region. Therefore, the equator sphere S_t contains a vertex added to the interior of a protecting ball or it intersects another unprotected region. In either case, the radius of S_t is larger than δ_f . Since $t \in \mathcal{DT}_2(P \cap Q)$, the sphere S_t cannot enclose any point of $P \cap Q$. Because the area of f_u is finite, there cannot be an infinite number of spheres S_t with radius larger than δ_f such that S_t does not enclose a point of $P \cap Q$. So, the number of added circumcenters is finite. Now, let us show that the total number of edges encroached upon by rejected circumcenters is also finite. Let \overline{ab} be a shield edge encroached upon by the circumcenter p of a triangle t of f_u . Since $t \in \mathcal{DT}_2(P \cap Q)$, the diameter sphere S_t cannot contain any point of $P \cap Q$. By hypothesis, the radius of S_t is greater than δ_f , which implies that \overline{ab} has length at least $\sqrt{2} \cdot \delta_f$. Hence, the number of such edges is finite. \square

3.5 Concluding Remarks

The description of Shewchuk’s algorithm in Subsection 3.3.2 is actually a simplified version of the original algorithm in [5]. The latter includes one more rule to refine tetrahedra. The goal of this rule is to improve the “quality” of the tetrahedrization, which is the subject of the next chapter and is an important issue in the context of several applications.

The algorithm in [7] described in subsection 3.4.3 was not the first algorithm for constructing a conforming Delaunay tetrahedrization of an input PLC that may not satisfy the angle constraint. To my knowledge, Murphy, Mount, and Gable [16] gave the first such an algorithm available in the literature. Their algorithm also uses an edge protection scheme, but it seems to insert too many extra vertices in the initial tetrahedrization, and therefore it may not be very attractive in practice.

Cohen-Steiner, Verdière, and Yvinec [7] implemented their algorithm and showed some complex examples of tetrahedrizations generated by it. Like Shewchuk’s algorithm, Cohen-Steiner, Verdière, and Yvinec’s algorithm may produce a tetrahedrization whose size complexity is not polynomial on the size of the input PLC. The time complexity of their algorithm remains an open question that was not addressed in [7]. Besides, they did not address the fact that their algorithm cannot handle input PLCs with isolated vertices or edges.

Chapter 4

Provably Good Quality Tetrahedrizations

This chapter examines a variation of the conforming Delaunay tetrahedrization problem studied in Chapter 3. Here, we are also interested in the “quality” of the conforming tetrahedrization. That is, we formally define two quality metrics, the radius-edge ratio and the aspect ratio, and we ask for a conforming Delaunay tetrahedrization of an input PLC in which the radius-edge ratio and the aspect ratio of every tetrahedron is bounded by some given constants. In many practical applications, such as finite-element analysis, the smaller the radius-edge ratio and the aspect ratio are, the better the tetrahedron is [17]. In such applications, the goodness of a tetrahedron is directly related to the accuracy and performance of calculations involving the tetrahedron.

The original conforming Delaunay tetrahedrization algorithm by Shewchuk [5] is provably guaranteed to generate tetrahedrization with bounded radius-edge ratio. In this chapter we present an extension of this algorithm, due to Xiang-Yang Li [6] that is provably guaranteed to generate tetrahedrization with bounded radius-edge ratio and bounded aspect ratio. Both algorithms, however, can only deal with input PLCs that satisfy the angle constraint. Next, we describe an algorithm by Siu-Wing Cheng and Sheung-Hung Poon [8] which is also built upon Shewchuk’s algorithm and is provably guaranteed to generate conforming Delaunay tetrahedrizations with bounded radius-edge ratio for input PLCs that may not satisfy the angle constraint.

Section 4.1 formally defines the quality metrics used to measure the quality of the tetrahedra of a Delaunay tetrahedrization. Section 4.2 describes Li’s algorithm. Section 4.3 introduces Cheng and Poon’s algorithm.

4.1 Quality Metrics

The Delaunay triangulation of a set of points in the plane enjoys the property of maximizing the minimum angle in the triangulation among all possible triangulations of the same set of points. That is probably the main reason for the use of the Delaunay triangulation in practical applications in 2D, as triangulations with bounded small angles lead to improvements on the accuracy and performance of calculations involving their triangles. Unfortunately, the max-min angle optimality does not generalize to higher dimensions, and the Delaunay tetrahedrization of a set of points can indeed contain tetrahedra with very small dihedral angles. These tetrahedra can degenerate the accuracy and performance of calculations involving them in practical applications [17].

A tetrahedron has very small angles either because its vertices are close to a line, or, if that is not the case, its vertices are close to a plane. In the former case, the tetrahedron is said to be *skinny*. In the latter case it is said to be *flat*. Skinny tetrahedra include the *needle* and the *wedge* tetrahedra illustrated in Figure 4.1, while flat tetrahedra include the *cap* and the *sliver* tetrahedra also illustrated in Figure 4.1. A flat tetrahedron has small angles but is not skinny. The needle, the wedge, and the cap have circumradius much larger than their shortest edge, but the sliver does not. However, the ratio of the circumradius of a sliver to the radius of its inscribed sphere could be as large as possible.

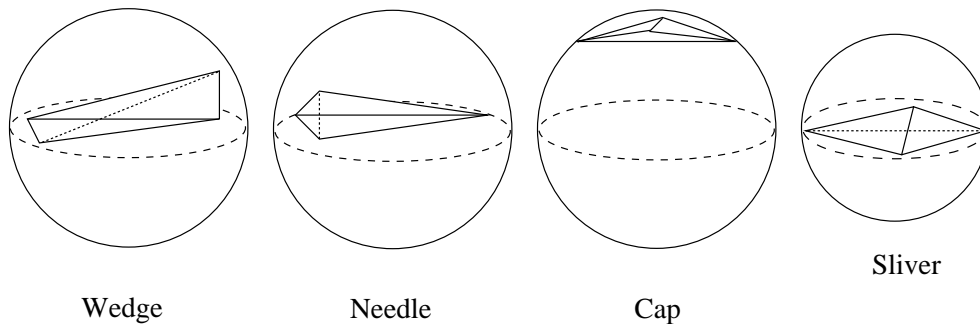


Figure 4.1: Tetrahedra with very small angles.

Let σ be a tetrahedron. We know that σ has a unique circumsphere S_σ . Let R be the radius of S_σ , and let L be the shortest edge of σ . Then, we define the *radius-edge ratio* ρ of σ to be

$$\rho = \rho(\sigma) = \frac{R}{L}.$$

A tetrahedron σ minimizes the radius-edge ratio ρ if and only if it is regular, in which case ρ is $\sqrt{6}/4 = 0.642\dots$. The needle, wedge, and cap tetrahedra have large radius-edge ratio, but the sliver does not.

Now, consider the radius r of the inscribed sphere of σ . We define the *aspect ratio* ϑ of σ to be

$$\vartheta = \vartheta(\sigma) = \frac{R}{r}.$$

A tetrahedron with small aspect ratio has small radius-edge ratio, but not vice-versa. A sliver has small radius-edge ratio but it can have an aspect ratio as large as possible.

The aspect ratio of σ can be bounded from above by its radius-edge ratio $\rho(\sigma)$ and the ratio $\varphi(\sigma)$ of the volume to the shortest edge of σ ,

$$\varphi(\sigma) = V/L^3,$$

where V is the volume of σ . That is, there exist $\rho_0 > 0$ and $\varphi_0 > 0$ such that, for every tetrahedron σ satisfying $\rho(\sigma) \leq \rho_0$ and $\varphi(\sigma) \geq \varphi_0$, the aspect ratio $\vartheta(\sigma)$ is at most $\frac{\sqrt{3} \cdot \rho_0^3}{\varphi_0}$ [6]. For technical reasons, in the next section we use the metrics ρ and φ instead of the aspect ratio ϑ to classify a tetrahedron as a sliver.

4.2 Sliver-Free Delaunay Refinement

We now describe Li's algorithm [6]. This algorithm takes as input a PLC satisfying the angle constraint, and then produces a Delaunay tetrahedrization that conforms to it and whose radius-edge ratio $\rho(\sigma)$ and ratio $\varphi(\sigma)$ of each tetrahedron σ is such that $\rho(\sigma) \leq \rho_0$ and $\varphi(\sigma) \geq \varphi_0$, for some fixed value of $\rho_0 > 0$ and $\varphi_0 > 0$. Li's algorithm extends Shewchuk's algorithm as follows: Shewchuk's algorithm, as described in [5], has one more rule in addition to the two rules showed in Subsection 3.3.2. This rule accounts for the elimination of tetrahedra with large radius-edge ratio by inserting its circumcenter as a new vertex to the tetrahedrization. Li [6] modified this rule so that his algorithm could also eliminate slivers. Li's algorithm allows the new vertex inserted by the algorithm to be chosen from a region, called *picking region*, that is a small sphere centered at the circumcenter of a diameter sphere of an encroached segment, equator sphere of an encroached triangle, or circumsphere of a tetrahedron. Vertices chosen from picking regions may avoid creating new slivers.

4.2.1 Picking Region

Let σ be a tetrahedron, and let S_σ be the circumsphere of σ . Then, the *picking region* $(c_\sigma, \delta R_\sigma)$ of σ is a sphere centered at c_σ and with radius δR_σ , where c_σ and R_σ are the center and the radius of S_σ , respectively, and $\delta < 1$ is a parameter of the algorithm whose value will be defined later. Similarly, we can define the picking region of a

segment and a triangle. Let s be a segment, and let S_s be the diameter sphere of σ . Then, the *picking region* $(c_s, \delta R_s)$ of s is the segment on s centered at c_s with length $2\delta R_s$, where c_s and R_s are the center and radius of S_s , respectively. Let t be a triangle (sub-facet), and let S_t be the equator sphere of t . Then, the *picking region* $(c_t, \delta R_t)$ of t is the disk in the supporting plane of t with center at c_t and radius δR_t , where c_t and R_t are the center and radius of S_t , respectively. Figure 4.2 illustrates the picking region of a tetrahedron, triangle, and segment.

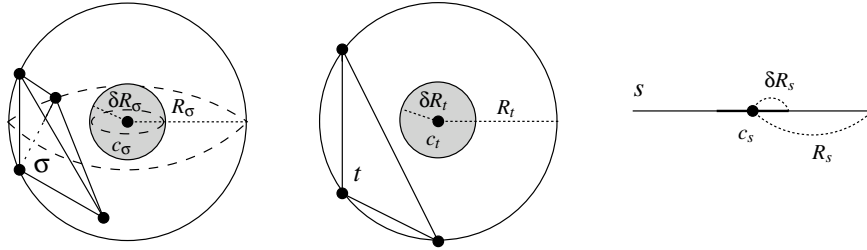


Figure 4.2: Picking region of a tetrahedron, triangle, and segment.

During its Delaunay refinement process, Li's algorithm always chooses new vertices from the picking region of a tetrahedron, triangle or segment. Since the circumcenter of the circumsphere of a tetrahedron, equator sphere of a triangle, or diameter sphere of a segment, belongs to the picking region, it's possible for the circumcenter to be chosen by the algorithm, but the chosen vertex does *not* have to be the circumcenter (as it is the case with Shewchuk's algorithm).

A *bad* tetrahedron, i.e., a tetrahedron with a large radius-edge ratio or a sliver, is eliminated by the insertion of a vertex v from its picking region or from the picking region of an encroached triangle or segment. The insertion of v removes the bad tetrahedron, but it may also create many new bad tetrahedra. We may try to avoid creating any new bad tetrahedra or at least avoid creating new slivers. Unfortunately, this is not always possible, as the insertion of any v in the picking region will create a new sliver. However, it is possible to avoid creating a certain kind of new sliver: The small sliver. If we avoid creating new small slivers then the other bad tetrahedra will eventually disappear. This observation is crucial for proving the termination of the algorithm.

We formally define a small sliver as follows: Let τ be a segment, triangle or tetrahedron, and assume that a new vertex v is inserted from the picking region $(c_\tau, \delta R_\tau)$ of τ by the algorithm. Then, a *small sliver* is any sliver tetrahedron σ of the Delaunay tetrahedrization obtained by inserting v that is incident to v and whose radius R_σ of its circumsphere is no greater than bR_τ , where b is another parameter of the algorithm to be defined later in this section. If the radius R_σ of the circumsphere of σ is greater than bR_τ then we call σ a *large sliver*.

4.2.2 Delaunay Refinement

Li's algorithm takes as input a PLC C satisfying the angle constraint as well as the four parameters ρ_0 , φ_0 , δ , and b . These parameters have to be carefully chosen so that the algorithm always terminates. The output of the algorithm is a Delaunay tetrahedrization that conforms to C and that does not contain any bad tetrahedra. The algorithm starts with a Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of the set P of vertices of C , and repeatedly applies the following three rules until a Delaunay tetrahedrization that conforms to C is obtained:

1. For any diameter sphere of a subsegment of C , if it contains any vertex inside it, then add its midpoint as a new vertex to P and update $\mathcal{DT}_3(P)$.
2. For any equator sphere of a sub-facet t of C , if it contains any vertex inside it, then add its circumcenter as a new vertex to P and update $\mathcal{DT}_3(P)$. However, if its circumcenter encroaches upon any subsegment of C , we split the subsegment instead of adding that circumcenter.
3. For any bad tetrahedron σ , add a point v to P from the picking region of σ such that v avoids creating small slivers. However, if the circumcenter c_σ of σ encroaches upon any sub-facet or subsegment of C , do not add v ; instead, apply the following rules:
 - (a) For any equatorial sphere of a sub-facet τ , if it contains the circumcenter c_σ of σ inside it, then add a point v to P from the picking region $(c_\tau, \delta R_\tau)$ of τ that avoids creating new small slivers. However, if the circumcenter c_τ of τ encroaches upon any subsegment of C , do not insert v ; instead, apply the following rule.
 - (b) For any diameter sphere of a subsegment s , if it contains c_σ or c_τ inside it, then add a point v to P from the picking region $(c_s, \delta R_s)$ of s that avoids creating new small slivers.

The key part the above algorithm is to find a point v in the picking region of a bad tetrahedron (or triangle or segment) that avoids creating new small slivers. Let $(c_\tau, \delta R_\tau)$ be the picking region from where v is to be chosen. The algorithm randomly selects v from $(c_\tau, \delta R_\tau)$ and builds a local tetrahedrization. Here, the local tetrahedrization is the one containing all tetrahedra incident to v . If there is a sliver among them, i.e., if there is any tetrahedron incident to v with circumradius less than or equal to bR_τ , the algorithm discards v and reselects a new point. This procedure is continued until the local tetrahedrization is free of small slivers. As we shall see next, by properly defining the parameters ρ_0 , φ_0 , δ , and b , this procedure is expected to terminate after a finite and small number of steps.

4.2.3 Bounds on Small Slivers

We now prove the existence of a point in the picking region of each element such that the insertion of this point will not introduce small slivers. This point can, though, introduce new large slivers and tetrahedra with large radius-edge ratio. We start the proof by studying the conditions under which a given triangle $\tau = [q, r, s]$ can be combined with a vertex p to create a sliver $\sigma = [p, q, r, s]$. The tetrahedron $\sigma = [p, q, r, s]$ is a sliver if $\rho(\sigma) \leq \rho_0$ and $\phi(\sigma) < \phi_0$, and, if this is the case, we have the following result:

Lemma 4.2.1. [6] *Let $\sigma = [p, q, r, s]$ be a tetrahedron such that $\rho(\sigma) \leq \rho_0$ and $\phi(\sigma) < \phi_0$ (σ is a sliver). Then, if $R_{[q,r,s]}$ is the radius of the circumcircle of the triangle $[q, r, s]$ then the distance d from the vertex p of σ to the closest point in the circumcircle of $[q, r, s]$ is at most $48\rho_0\phi_0R_{[q,r,s]}$.*

Note that, for any triangle $[q, r, s]$, there is a set of points P such that $[p, q, r, s]$ is a sliver for $p \in P$. From Lemma 4.2.1, the set P must be contained in a solid torus defined by the points at distance at most $d = 48\rho_0\phi_0R_{[q,r,s]}$ from the circumcircle of $[q, r, s]$ with radius $R_{[q,r,s]}$. We call the set of points P the *forbidden region* $FR_{[q,r,s]}$ of the triangle $[q, r, s]$. Figure 4.3 shows the forbidden region of a triangle. The volume of the torus is the perimeter of the circle times the area of the sweeping disk. So, the volume of the forbidden region $FR_{[q,r,s]}$ is at most $2\pi R_{[q,r,s]} \cdot \pi d^2 = c_1 R_{[q,r,s]}^3$, where $c_1 = 2\pi^2(48\rho_0\phi_0)^2$.

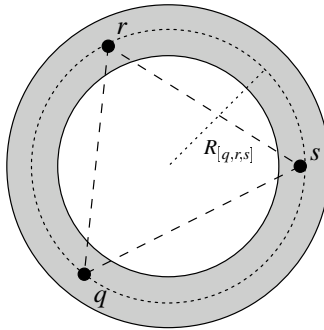


Figure 4.3: The forbidden region of a triangle.

Now, suppose that we select a point p in the picking region of a tetrahedron, sub-facet or subsegment τ . The point p will not create new small slivers if and only if p does not belong to the union $\bigcup FR_{[q,r,s]}$ of the forbidden region $FR_{[q,r,s]}$ of every triangle $[q, r, s]$ that will form a small sliver with p . In order to prove that there exists such a point p in any picking region, we start by stating some results related to the intersection of the forbidden region $FR_{[q,r,s]}$ with any plane or line around the triangle $[q, r, s]$. Later, we state that, for any point p in a picking region, there is at most a finite number $W > 0$ of triangles $[q, r, s]$ that can form a small sliver with p . Then, the result follows by

proving that the union of any W such forbidden regions cannot entirely cover the picking region to which p belongs.

Lemma 4.2.2. [6] *Let τ be a triangle and let H be a plane. Then, the area of the intersection of the forbidden region FR_τ with H is at most $c_2 R_\tau^2$, where $c_2 = 192\pi\rho_0\phi_0$ and R_τ is the radius of the circumcircle of τ .*

Lemma 4.2.3. [6] *Let τ be a triangle and let L be a line. Then, the length of the intersection of the forbidden region FR_τ with L is at most $c_3 R_\tau$, where $c_3 = 16\sqrt{3\rho_0\phi_0}$ and R_τ is the radius of the circumcircle of τ .*

Lemma 4.2.4. [6] *Let p be any point in the picking region of a tetrahedron, sub-facet or subsegment τ . Then, there is at most a constant number of new small slivers incident to p .*

Let W be the maximum number of triangles $[q, r, s]$ that can form new small slivers $[p, q, r, s]$ with a point p from the picking region of a tetrahedron, sub-facet or subsegment τ . From Lemma 4.2.4, we have that such W exists and is finite. By definition of small sliver, we have that $R_{[q, r, s]} \leq R_{[p, q, r, s]} \leq bR_\tau$, where $R_{[q, r, s]}$ and $R_{[p, q, r, s]}$ are the radius of the circumcircle of $[q, r, s]$ and the radius of the circumsphere of $[p, q, r, s]$, respectively. If τ is a tetrahedron, from the volume of the forbidden region, we have that the volume of $\bigcup FR_{[q, r, s]}$ is at most $W \cdot c_1 (bR_\tau)^3$, where $c_1 = 2\pi^2(48\rho_0\phi_0)^2$. If τ is a sub-facet, from Lemma 4.2.2, the area of the intersection of $\bigcup FR_{[q, r, s]}$ with the supporting plane of τ is at most $W \cdot c_2 (bR_\tau)^2$, where $c_2 = 192\pi\rho_0\phi_0$. If τ is a subsegment, from Lemma 4.2.4, the sum of the length of the segments resulting from the intersection of $\bigcup FR_{[q, r, s]}$ and the supporting line of τ is at most $W \cdot c_3 (bR_\tau)$, where $c_3 = 16\sqrt{3\rho_0\phi_0}$. So, to prove the existence of a point p that does not create any new small sliver, it suffices to show that the following conditions simultaneously hold: $W \cdot c_1 (bR_\tau)^3 < (\delta R_\tau)^3$, $W \cdot c_2 (bR_\tau)^2 < (\delta R_\tau)^2$, and $W \cdot c_3 (bR_\tau) < (\delta R_\tau)$. This can be done by setting appropriate values to the constants ρ_0 , ϕ_0 , δ , and b .

4.2.4 Termination

The proof of termination of Li's algorithm uses a similar argument to the one used to prove the termination of Shewchuk's algorithm: The shortest edge of the Delaunay tetrahedrization cannot decrease after a finite number of steps. However, there are two additional complications in Li's algorithm that make its proof of termination harder:

- If σ is a bad tetrahedron whose circumcenter encroaches upon a sub-facet or subsegment τ , the algorithm picks up a point v from the picking region of τ . However, while the circumcenter of τ is guaranteed to be in the circumsphere of σ , there is no guarantee that v will be. So, the insertion of v in τ might not cause the elimination of σ .

- The elimination of a bad tetrahedron can cause the appearance of new bad tetrahedra, except for new small slivers.

We start by proving that every bad tetrahedron whose circumcenter encroaches upon a sub-facet or subsegment can be eliminated after a finite number of sub-facet or subsegment splittings. This fact follows from the following lemma.

Lemma 4.2.5. *Let σ be a bad tetrahedron whose circumcenter c_σ encroaches upon at least one sub-facet or subsegment. Let R_σ be the circumradius of σ , and let R be the largest circumradius among the circumradii of all equator or diameter spheres enclosing c_σ . If $(1 + \delta)R \leq R_\sigma$ then the tetrahedron σ is eliminated by a vertex from either its picking region or a picking region of a sub-facet or subsegment encroached upon by c_σ .*

Proof. Let c_σ be the circumcenter c_σ of σ . If c_σ does not encroach upon any sub-facet or subsegment, the algorithm selects a point p from the picking region of σ . Since v is inside the circumsphere of σ , its insertion causes the elimination of the bad tetrahedron σ , and it also avoids creating small slivers incident to p . If c_σ encroaches upon any sub-facet or subsegment, a point v must be selected from the picking region $(c_\tau, \delta R_\tau)$ of a subsegment or sub-facet τ encroached upon by c_σ . Since τ is encroached upon by c_σ , we have that $d(c_\sigma, c_\tau) < R_\tau$. For any point v inside the picking region of τ , we have that $d(c_\sigma, v) \leq d(c_\sigma, c_\tau) + d(c_\tau, v) < (1 + \delta)R_\tau$. By hypothesis, $R_\tau \leq R \leq R_\sigma / (1 + \delta)$. Hence, $d(c_\sigma, v) < R_\sigma$, which implies that v is inside the circumsphere of σ . So, the tetrahedron σ must be eliminated with the insertion of v . \square

Now, notice that there are only finitely many points that can be inserted into sub-facets and subsegments encroached by the circumcenter of a bad tetrahedron σ such that $(1 + \delta)R > R_\sigma$, where R_σ is the circumradius of σ and R is the largest circumradius among the circumradii of all equator or diameter spheres enclosing the circumcenter c_σ of σ . So, from Lemma 4.2.5, the tetrahedron σ must be eliminated after a finite number of vertex insertions.

We now prove the termination of the algorithm by showing that the shortest edge of the tetrahedrization cannot decrease too much after the elimination of a bad tetrahedron. For the sake of clarity, we classify the bad tetrahedra into original slivers, created slivers, and tetrahedra with large radius-edge ratio. *Original slivers* are all slivers present in the tetrahedrization after the application of rules 1 and 2. *Created slivers* are (large) slivers created during the application of rule 3. A tetrahedron σ with large radius-edge is a tetrahedron for which $\rho(\sigma) > \rho_0$.

Lemma 4.2.6. [6] *After eliminating all original slivers, the length of the shortest edge of the tetrahedrization is no smaller than a factor of $(1 - \delta)/4$ of the length of the shortest edge of the original tetrahedrization.*

Proof. As we mentioned in Subsection 4.1, for any tetrahedron τ , the radius-edge ratio $\rho(\tau)$ of τ satisfies $\rho(\tau) = R_\tau/L_\tau \geq \sqrt{6}/4 > 1/2$, which implies that $R_\tau > L_\tau/2$, where R_τ and L_τ are the circumradius and the shortest edge of τ , respectively. Let σ be an original sliver and assume that point p is inserted for eliminating σ . If the point p is chosen from the picking region of σ , then for every vertex v of the tetrahedrization, $d(v, p) \geq (1 - \delta)R_\sigma > (1 - \delta)L_\sigma \geq (1 - \delta)L/2$, where R_σ is the circumradius of σ , L_σ is the shortest edge of σ , and L is the shortest edge of tetrahedrization before the insertion of p . Otherwise, if p is chosen from the picking region of an encroached sub-facet or subsegment ω , then we have $R_\omega \geq R_\sigma/\sqrt{2} > R_\sigma/2$. Since $R_\sigma > L_\sigma/2 \geq L/2$, for every vertex v of the tetrahedrization, $d(v, p) \geq (1 - \delta)R_\omega > (1 - \delta)L/4$. Hence, our claim follows. \square

Lemma 4.2.7. [6] *Assume that tetrahedron σ has large radius-edge ratio or it is a (large) sliver created after a point insertion. Then the length of the shortest edge length of the tetrahedrization after eliminating σ is no shorter than a factor of*

$$\min\left\{\frac{(1 - \delta)\rho_0}{2}, \frac{(1 - \delta)b}{4}\right\}$$

of the length of the shortest edge before σ is eliminated.

Proof. Let σ be a bad tetrahedron. Then, assume that the circumcenter c_σ of σ does not encroach upon any sub-facet or subsegment. So, the tetrahedron σ is eliminated by the insertion of a point p from its picking region. We have two cases: Either (1) σ is a tetrahedron with large radius-edge ratio or (2) σ is a (large) sliver created by the insertion of a point in the tetrahedrization. In case (1), we have that $\rho(\sigma) = R_\sigma/L_\sigma > \rho_0$, where R_σ and L_σ are the circumradius and the shortest edge of σ , respectively. The shortest edge introduced after inserting p is no shorter than $(1 - \delta)R_\sigma > (1 - \delta)\rho_0L_\sigma$. Thus, if L_1 and L_2 are the shortest edge of the tetrahedrization before and after the insertion of p , respectively, we have that $L_2 > (1 - \delta)\rho_0L_1$. In case (2) the tetrahedron σ was generated by the insertion of a point from the picking region of a tetrahedron, sub-facet or subsegment τ . By the definition of large sliver (see Subsection 4.2.3), we know that $R_\sigma > bR_\tau$. The shortest edge introduced after inserting p is no shorter than $(1 - \delta)R_\sigma$, which is no shorter than $(1 - \delta)bR_\tau$. Since, for every tetrahedron τ , $R_\tau > L_\tau/2$, where L_τ is the shortest edge of τ , we have $(1 - \delta)R_\sigma > \frac{(1 - \delta)b}{2}L_\tau$, and thus $L_2 > \frac{(1 - \delta)b}{2}L_1$. It remains to show that the theorem also holds when c_σ encroaches upon a sub-facet or subsegment. Assume that c_σ encroaches upon a sub-facet or subsegment τ . So, c_σ is inside the diameter (equator) sphere of τ if τ is a subsegment (sub-facet). Thus, $R_\tau > R_\sigma/\sqrt{2} > R_\sigma/2$, and the shortest edge introduced after inserting p from $(c_\tau, \delta R_\tau)$ is no shorter than $\frac{(1 - \delta)}{2}R_\sigma$. If σ is a tetrahedron with large radius-edge ratio, we have $R_\sigma > \rho_0L_\sigma$. It follows that $L_2 > \frac{(1 - \delta)}{2}\rho_0L_\sigma \geq \frac{(1 - \delta)}{2}\rho_0L_1$. Otherwise, the tetrahedron σ is a (large) sliver, and we have $R_\sigma > bR_\tau$. As a result we get $L_2 > \frac{(1 - \delta)b}{4}L_\tau \geq \frac{(1 - \delta)b}{4}L_1$. So, our claim holds. \square

From Lemma 4.2.6, we have that, after eliminating all original slivers, the length of the shortest edge of the tetrahedrization may decrease, but it does not decrease too much, and it is bounded from below. From Lemma 4.2.7, we have that, given $\rho_0 > 2$, if we choose $\delta \leq 1 - \frac{\rho_0}{2}$ to define the picking regions, and select $b \geq 2\rho_0$ to define large slivers, the shortest edge length of the tetrahedrization will never decrease after the elimination of all original slivers. In addition, these choices will not affect the satisfiability of the conditions for bounding small slivers, $W \cdot c_1(bR_\tau)^3 < (\delta R_\tau)^3$, $W \cdot c_2(bR_\tau)^2 < (\delta R_\tau)^2$, and $W \cdot c_3(bR_\tau) < (\delta R_\tau)$, as we still have one degree of freedom: The parameter φ_0 that defines what a sliver is. Hence, termination follows from the fact that we cannot insert an infinite amount of vertices in a bounded region if they have to be separate from each other from a distance of at least $d > 0$.

4.3 Bounded Radius-Edge Ratio and PLCs with Small Angles

We now describe the conforming Delaunay tetrahedrization algorithm due to Cheng and Poon [8]. The main contribution of this algorithm is to extend Shewchuk's algorithm, as described in [5], to handle input PLCs that may not satisfy the angle constraint. That is, Cheng and Poon's algorithm generates a conforming Delaunay tetrahedrization with bounded radius-edge ratio for input PLCs that may not satisfy the angle constraint. Like Cohen-Steiner, Verdière, and Yvinec's algorithm described in Section 3.4, Cheng and Poon's algorithm also employs an edge protection scheme to guarantee the termination of their algorithm.

The edge protection scheme of Cheng and Poon's algorithm differs from the one in Cohen-Steiner, Verdière, and Yvinec's algorithm in that the former creates a new complex by adding new facets, edges, and vertices to \mathcal{C} . The new facets and edges are spherical facets and arcs, respectively, resulting from the intersection of the facets and edges of \mathcal{C} with a collection of protecting balls covering the edges of \mathcal{C} . We denote the resulting augmented complex by Q .

The idea is to apply Delaunay refinement to Q to create a tetrahedrization of the space outside the protecting balls without adding new vertices to the space inside the protecting balls. The tetrahedrization of the space inside the protecting balls is an automatic consequence of the tetrahedrization of the outside space. By construction of Q , termination follows from the fact that, in the space outside the protecting balls, the angle between two adjacent faces of Q is either undefined or equal to $\frac{\pi}{2}$. In this case, the Projection Lemma (see Subsection 3.3.3) holds for the outside space, and the termination conditions are the same as the ones in Shewchuk's algorithm [5]. The edge protection scheme of Cheng and Poon's algorithm is also restricted to PLCs for which every vertex is an endpoint of an edge, and every edge belongs to the boundary of at least one facet of the PLC.

4.3.1 Augmented Complex Q

Let C be a PLC such that every edge of C belongs to the boundary of at least one facet of C , and let \mathcal{B} be a set of closed balls, called *protecting balls*, satisfying the same conditions as in Subsection 3.4.2 plus the condition that if two balls in \mathcal{B} intersect then they are orthogonal to each other. We refer the reader to [8] for an algorithm for constructing such a set of protecting balls. We call two protecting balls *consecutive* if their centers are neighbors on some edge of C . Clearly, consecutive balls intersect each other and therefore they are orthogonal.

Let $Z(\mathcal{B}) = bd(\bigcup_{B \in \mathcal{B}} B)$ be the boundary of the union of all protecting balls in \mathcal{B} . We call *buffer zone* the space inside $Z(\mathcal{B})$. For each edge $[u, v]$ of C , let $B_{[u, v]}$ be the sequence of protecting balls in \mathcal{B} whose centers lie on $[u, v]$. Then, $Z(\mathcal{B}) \cap \left(\bigcup_{B \in B_{[u, v]}} B \right)$ consists of a sequence of *rings* delimited by two spheres with holes. For each sphere $B' \in B_{[u, v]}$ other than the ones centered at u and v , the sphere B' contributes to exactly one ring of $Z(\mathcal{B}) \cap \left(\bigcup_{B \in B_{[u, v]}} B \right)$.

We merge $Z(\mathcal{B})$ with C to create a new complex Q . Note that $Z(\mathcal{B})$ splits each facet of C into two smaller facets, one inside $Z(\mathcal{B})$ and one outside $Z(\mathcal{B})$. These facets are called the *flat facets* of Q . For each edge $[u, v]$ of C , each ring $Z(\mathcal{B}) \cap B(x)$, where $B(x)$ is a protecting ball with center at x inside $[u, v]$, is divided by the facets of C incident to $[u, v]$ into curved rectangular patches. For each vertex v of C , we have that $Z(\mathcal{B}) \cap B(v)$, where $B(v)$ is the protecting ball with center at v , is divided by facets of C incident to v into spherical patches. Curved rectangular patches and spherical patches are the *curved facets* of Q . The centers of protecting balls split the edges of C into the *linear edges* of Q . The circular arcs on the boundary of curved facets are the *curved edges* of Q . The set of vertices of Q consists of the endpoints of linear and curved edges. Figure 4.4 illustrates an augmented complex Q in the neighborhood of two adjacent facets of C .

Note that, for any protecting ball B and any curved facet f in $Z(\mathcal{B}) \cap B$, the boundary $bd(f)$ of f consists of curved edges that lie at the intersections between B and either facets of C or protecting balls of \mathcal{B} consecutive to B . Furthermore, the curved edges of $bd(f)$ resulting from these intersections alternate in $bd(f)$. If the center v of B belongs to the boundary $bd(f')$ of a facet f' of C and v is not a vertex of C , then f is a rectangular patch and f' can contribute to at most one edge of $bd(f)$. If v is a vertex of C then f is a spherical patch and v appears in exactly one simple cycle of $bd(f')$. Since B cannot intersect more than one cycle or intersect one cycle in $bd(f')$ more than once, we deduce that $B \cap f'$ is connected. It follows that f' also contributes at most one edge to $bd(f)$. However, a hole on $Z(\mathcal{B}) \cap B$ can contribute several edges to $bd(f)$ when v is a vertex of C .

By construction, the angle between adjacent faces of Q in the space outside \mathcal{B} is equal to $\frac{\pi}{2}$. The following Lemma precisely states this important property:

Lemma 4.3.1. [8] *Let Q be the augmented complex of an input PLC C as described above. Then, the following hold:*

1. *Let f be a curved facet of Q , and let f' be a flat or curved facet of Q adjacent to f . If f and f' do not lie on the same sphere, the normal to f' at any point in $f \cap f'$ is tangent to f .*
2. *Let e and e' be two adjacent curved edges that do not lie on the same circle. Let l (resp. l') be the line through $e \cap e'$ that is tangent to and coplanar with e (resp. e'). Then, the line l is perpendicular to the line l' .*
3. *Let f be a flat or curved facet, and let e be a curved edge adjacent to f . If f and e do not lie on the same plane or sphere, then the normal to f at $e \cap f$ is tangent to and coplanar with e .*

Lemma 4.3.1 motivates the use of Delaunay refinement in the space outside $BZ(\mathcal{B})$. The idea is to compute a Delaunay tetrahedrization that approximates Q and conforms to the edges and facets of C . Due to the fact that Q has curved edges and facets, the Delaunay refinement must be modified to take these edges and facets into account. As we shall see next, curved edges and facets are approximated by straight line segments and triangles, respectively.

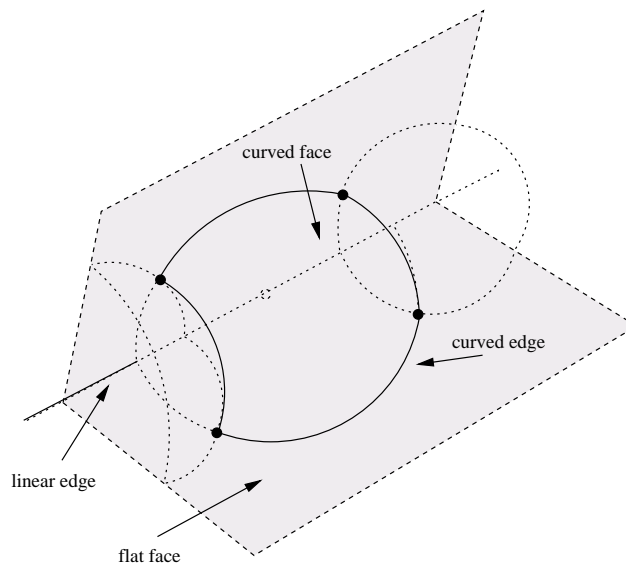


Figure 4.4: Flat and curved facets, and linear and curved edges of the augmented complex.

4.3.2 Delaunay Refinement

Cheng and Poon's algorithm takes as input a PLC C such that every vertex of C is an endpoint of some edge of C and every edge of C is incident to the boundary of at least one facet of C . The input PLC C does not have to satisfy the angle constraint. The algorithm starts by creating the augmented complex Q by defining an appropriate set \mathcal{B} of protecting balls, and then generates a Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of the set of vertices P of Q . Next, the algorithm refines $\mathcal{DT}_3(P)$ by keeping inserting vertices into P until $\mathcal{DT}_3(P)$ conforms to C and every tetrahedron σ of $\mathcal{DT}_3(P)$ has bounded radius-edge ratio $\rho(\sigma)$ no greater than a constant $\rho_0 > 16$.

During the process of generating the conforming tetrahedrization, the algorithm modifies Q by creating several types of geometric objects. We now define these objects and some notation needed by the description of the algorithm. Given a circle C on a sphere S , the *orthogonal sphere of S at C* is the sphere orthogonal to S that passes through C . Each curved edge e of Q is split by vertices of P into *helper arcs*. Let S be the equatorial sphere of e , i.e., the curved edge e lies on the equator of S . Let \widehat{pq} be a helper arc on e . The *circumcap K of \widehat{pq}* is the smallest (spherical) cap on S that contains \widehat{pq} . Figure 4.5 shows the circumcap of a helper arc \widehat{pq} . Note that \widehat{pq} is on the equator of S . If the angular width of \widehat{pq} is less than π , the *normal sphere of \widehat{pq}* is the orthogonal sphere of S at $bd(K)$, where $bd(K)$ is the boundary of K . We say that \widehat{pq} is *encroached upon by a point v* if v lies inside its normal sphere. We say that \widehat{pq} is *wide* if its angular width is larger than $\frac{\pi}{3}$.

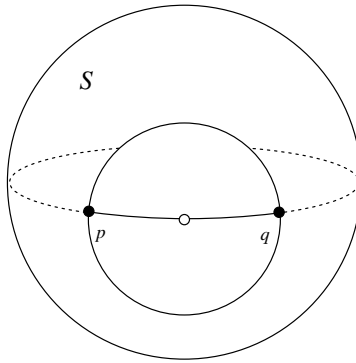


Figure 4.5: The circumcap of a helper arc.

Helper triangles are defined when no helper arc is wide or encroached by a vertex in P . Let $conv_B$ be the convex hull of $P \cap Z(\mathcal{B}) \cap B$ for a protecting ball B . If a convex polygon X with more than three vertices appears as a boundary facet of $conv_B$, then we triangulate X as follows. Let H be the supporting plane of X . The *circumcap of X* is the cap on B that is bounded by $H \cap B$ and separated from $conv_B$ by H . First, for each helper arc \widehat{pq} such that $p, q \in bd(X)$, where $bd(X)$ is the boundary of X , and \widehat{pq} lies on the circumcap of X , we insert the line segment \overline{pq} as a diagonal in

X . Then, we arbitrarily complete the triangulation of X .

After triangulating every polygon X on the boundary of conv_B and with more than three vertices, a boundary triangle t of conv_B is said to be a *helper triangle* if no hole on $Z(\mathcal{B}) \cap B$ contains all vertices of t on its boundary. Let H be the plane containing a helper triangle t . The *circumcap* of t is the cap K on B that is bounded by $H \cap B$ and separated from conv_B by H . If the angular diameter of K is less than π , the *normal sphere* of t is the orthogonal sphere of B at $bd(K)$, where $bd(K)$ is the boundary of K , and t is said to be *encroached upon by a point* v if v lies inside its normal sphere. We say that t is *wide* if the angular diameter of K is larger than $\frac{\pi}{3}$. Figure 4.6 illustrates helper triangles and non-helper triangles on the boundary of conv_B for a ball B of \mathcal{B} .

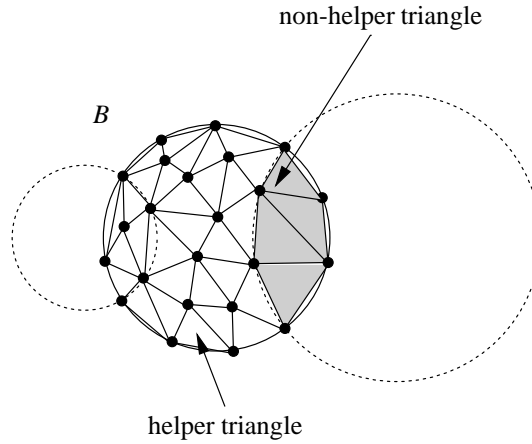


Figure 4.6: Helper triangles and non-helper triangles.

Sub-Facets are defined when no helper arc is wide or encroached by a vertex in P . For every facet f of \mathcal{C} , a *sub-facet* is a triangle on f in the Delaunay triangulation $\mathcal{DT}_2(P \cap f)$. Note that we define sub-facet in the same way as in Shewchuk's algorithm and Cohen-Steiner, Verdière, and Yvinec's algorithm. That is, we use facets of \mathcal{C} to define sub-facets instead of flat facets of Q . The reason for that is that Cheng and Poon's algorithm only approximates Q and it does not conform to the curved boundary edges of flat faces. The *circumcap* of a sub-facet f is the disk bounded by the circumcircle of f . The *normal sphere* of f is the equator sphere of f . We say that f is *encroached upon by a point* v if v lies inside the normal sphere of f .

We are now ready to describe the algorithm. Let P be the set of vertices of Q , then the algorithm repeatedly invokes the applicable rule of least index in the following list. When no rule is applicable, the Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of the current set of points P is the output. Recall that $\rho_0 > 16$ is a constant defined a priori.

1. If there exists any helper arc \widehat{pq} such that \widehat{pq} is wide or encroached upon by a vertex in P , add the midpoint v

of the helper arc to P and Q . The insertion of the midpoint of \widehat{pq} into Q eliminates \widehat{pq} from Q and adds the arcs \widehat{pv} and \widehat{vq} to Q .

2. If there exists any helper triangle t such that t is wide or encroached upon by a vertex in P , add the center v of the circumcap of t to P if v does not encroach upon any helper arc of Q . If it does, do not add v to P ; instead apply rule 1 to split all helper arcs encroached upon by v .
3. If there exists any sub-facet f such that f is encroached upon by a vertex in P , add the center v of the circumcap of f to P if v does not encroach upon any helper arc of Q . If it does, do not add v to P ; instead apply rule 1 to split all helper arcs encroached upon by v .
4. If there exists any tetrahedron σ such that $\rho(\sigma) > \rho_0$ and no vertex of σ lies inside $Z(\mathcal{B})$, then add the circumcenter v of σ to P if v does not encroach upon any helper arc or helper triangle of Q , or sub-facet of $\mathcal{DT}_2(P \cap f)$ for every facet f of \mathcal{C} . If it does, do not add v to P ; instead apply rule 1 to split all helper arcs encroached upon by v , and then rules 2 and 3 to split the helper triangles and sub-facets encroached upon by v .

4.3.3 Boundary Conformity

We now describe the proof of boundary conformity of Cheng and Poon's algorithm. The idea behind the proof is to show that the algorithm never inserts a vertex inside $Z(\mathcal{B})$, and then prove that whenever no helper arc is wide or encroached and no sub-facet is encroached, the tetrahedrization is conforming. We start with some definitions, and then we state the relevant lemmas.

Given a sphere S and a point p outside S , we denote by $K(p, S)$ the (spherical) cap on S visible from p . Given a cap K on a sphere S , if the angular diameter of K is less than π , we use K^\perp to denote the orthogonal sphere of S at $bd(K)$, where $bd(K)$ is the boundary of K . If S is a plane (infinite sphere), then K^\perp is the equator sphere of K . For any point $q \in bd(K(p, S))$, the line segment \overline{pq} is tangent to S , so p is the center of $K(p, S)^\perp$.

Claim 4.3.1. [8] *Let S be a sphere, and let S' be a sphere such that $S \cap S'$ is an equator of S' . Then, for any point p on the plane containing $S \cap S'$ and outside S' , the orthogonal sphere $K(p, S')^\perp$ of S' at $bd(K(p, S'))$ is also orthogonal to S .*

Claim 4.3.2. [8] *Let p and q be two non-diameter points on a sphere S centered at x . Let \mathcal{N} be the set of spheres orthogonal to S that pass through p and q . There exists a unique circle C such that*

1. The circle C is coplanar with the triangle $[p, q, x]$, it passes through p and q , and \mathcal{N} is the set of spheres that pass through C .
2. The locus of centers of spheres in \mathcal{N} is the line l through the center of C perpendicular to the plane containing C .

Claim 4.3.3. [8] *Let S be a sphere, and let K_1 and K_2 be caps on S with angular diameter less than π . If $K_2 \subseteq K_1$, then K_1^\perp encloses K_2^\perp .*

For the following lemmas, we need some definitions and notations. We use K_τ to denote the circumcap of a helper arc, helper triangle, or sub-facet τ . If the angular diameter of K_τ is less than π , we use K_τ^\perp to denote the normal sphere of τ . We also extend the definition of circumcap and normal sphere to any arc β . The circumcap K_β is the smallest cap on the equator sphere of β that contains β , and the normal sphere K_β^\perp of β is defined as before if the angular diameter of K_β is less than π . We say that a helper arc *belongs to* the curved edge that contains it. A sub-facet τ *belongs to* the flat facet f if f lies outside $Z(\mathcal{B})$ and f contains the vertices of τ . Note that it is possible that $\tau \not\subseteq f$ (if τ is not in the Delaunay tetrahedrization). Clearly, a sub-facet belongs to at most one flat facet. A helper triangle t *belongs to* a curved facet f if there exists a connected subset $\gamma \subseteq \text{int}(K_t) \cap f$ such that the closure $cl(\gamma)$ of γ contains the vertices of t .

Lemma 4.3.2. [8] *Assume that the algorithm has not inserted any vertex inside $Z(\mathcal{B})$. If there is no wide or encroached helper arc, then each helper triangle belongs to exactly one curved facet of Q and, if t is a helper triangle belonging to a curved facet f of Q , the center of K_t lies on f .*

Lemma 4.3.3. [8] *Assume that the algorithm has not inserted any vertex inside $Z(\mathcal{B})$. If there is no wide or encroached helper arc, then if τ is a sub-facet belonging to a flat facet f of Q , the center of K_τ lies on f .*

Lemma 4.3.2 ensures that the new vertices added to the tetrahedrization by rule 2 will not be outside the curved facets that contain the encroached helper triangles, as rule 2 is applied after the elimination of wide or encroached helper arcs by the application of rule 1. Similarly, Lemma 4.3.3 ensures that the new vertices added to the tetrahedrization by rule 3 are on the flat faces of Q .

Lemma 4.3.4. [8] *Assume that the algorithm has not inserted any vertex inside $Z(\mathcal{B})$. Let B_1 and B_2 be two consecutive protecting balls. When there is no wide or encroached helper arc, the following conditions (1)–(3) hold. When there is no wide or encroached helper arc and triangle, the following conditions (1)–(4) hold.*

1. For any flat facet f incident to the center b_1 of B_1 and helper arc $\widehat{pq} \subseteq Z(\mathcal{B}) \cap B_1 \cap f$, the equator sphere of the triangle $[p, q, b_1]$ is empty.

2. For any helper arc endpoint $p \in (B_1 \cap B_2)$, the equator sphere of the triangle $[p, b_1, b_2]$ is empty.
3. For any helper arc $\widehat{pq} \subseteq (B_1 \cap B_2)$, the circumsphere of the tetrahedron $[p, q, b_1, b_2]$ is empty.
4. For any helper triangle $[p, q, r]$ on $\text{conv}_{B_1} = \text{conv}(P \cap Z(\mathcal{B}) \cap B_1)$, where P is the current set of vertices of the tetrahedrization, the circumsphere of the tetrahedron $[p, q, r, b_1]$ is empty.

Proof. (i) Let $\alpha = \widehat{pq}$, and let S denote the equator sphere of the triangle $[p, q, b_1]$. Note that the centers of B_1 , S , and K_α^\perp lie on a straight line. Since b_1 lies on S but outside K_α^\perp , the center of S lies between b_1 and the center of K_α^\perp . Thus, the sphere S is enclosed by $bd(B_1 \cup K_\alpha^\perp)$. By hypothesis, the vertex b_1 is the only vertex inside B_1 and the interior of K_α^\perp is also empty. So, the sphere S is empty. (ii) Let S be the equator sphere of the triangle $[p, b_1, b_2]$. Since B_1 and B_2 are orthogonal, they intersect at a right angle, and therefore the angle $\angle b_1 p b_2$ in the triangle $[p, b_1, b_2]$ is equal to $\frac{\pi}{2}$. Thus, $\overline{b_1 b_2}$ is the diameter of S , which implies that $bd(B_1 \cup B_2)$ encloses S . By hypothesis, the only vertices inside $bd(B_1 \cup B_2)$ are b_1 and b_2 . So, the sphere S is empty. (iii) Note that the circumsphere S of the tetrahedron $[p, q, b_1, b_2]$ is exactly the same as the equator sphere of the triangle $[p, b_1, b_2]$. So, from (ii), the sphere S is empty. (iv) Let $t = [p, q, r]$, and let S denote the circumsphere sphere of the triangle $[p, q, r, b_1]$. By hypothesis, we have K_t^\perp is empty. Since the centers of S , B_1 and K_t^\perp lie on a straight line and b_1 lies on S but outside K_t^\perp , the center of S lies between b_1 and the center of K_t^\perp , and therefore S is enclosed by $bd(B_1 \cup K_t^\perp)$. So, the result follows as in (i). \square

Lemma 4.3.5. [8] *The algorithm never inserts any vertex inside $Z(\mathcal{B})$.*

Proof. We just sketch the proof here. See [8] for details. Suppose that the algorithm inserts vertices inside $Z(\mathcal{B})$. Let v be the first vertex inserted inside $Z(\mathcal{B})$. Certainly, the algorithm is not applying rule 1. So, there cannot be any wide or encroached helper arc by the time v is inserted into $Z(\mathcal{B})$. According to Lemma 4.3.2, the vertex v cannot be inserted by rule 2 as well. So, there cannot be any wide or encroached helper arc or triangle by the time v is inserted into $Z(\mathcal{B})$. If v is inserted by rule 3 to split an encroached sub-facet τ , then, from Lemma 4.3.4, the sub-facet τ does lie inside $Z(\mathcal{B})$. Lemma 4.3.4 further implies that τ belongs to a flat facet outside $Z(\mathcal{B})$. But, from Lemma 4.3.3, we have that v does not lie inside $Z(\mathcal{B})$, which is a contradiction. So, the vertex v must have been inserted by rule 4 into some protecting ball B , and thus v is the circumcenter of a tetrahedron σ . By rule 4, the tetrahedron σ has no vertex inside $Z(\mathcal{B})$. Furthermore, at least one vertex of σ must lie outside B , as B cannot be the empty circumsphere of σ . Let S be the circumsphere of σ , and let K be the cap on S that is bounded by $B \cap S$ and lies inside S . By the emptiness property of Delaunay, we know that the center of B is not inside S . Thus, the angular diameter of K is less than π and therefore K^\perp is well-defined. If $K \cap K_t = \emptyset$ for all helper triangles t on $\text{conv}_B = \text{conv}(P \cap Z(\mathcal{B}) \cap B)$, then $K \subseteq K(p, bd(B))$, where p is the center of some protecting ball B' consecutive to B . It follows that $bd(B \cup B')$

encloses S and hence σ , contradicting the fact that the algorithm had not inserted any vertex inside $Z(\mathcal{B})$ before v . So, there must be some helper triangle t on conv_B such that $K \cap K_t \neq \emptyset$. If $K \subseteq K_t$ for some triangle t on conv_B then S lies inside $bd(B \cup K^\perp)$ as v is inside B by assumption. By Claim 4.3.3, we have $K^\perp \subseteq K_t^\perp$. So, the sphere S lies inside $bd(B \cup K_t^\perp)$, and hence K_t^\perp contains a vertex of σ outside B , which is a contradiction as t is assumed to not be encroached upon by any vertex. If $K \not\subseteq K_t$ for every triangle t on conv_B , by the emptiness property of the Delaunay tetrahedrization, the sphere S does not enclose any vertex of a triangle on conv_B . So, we have $bd(K) \cap bd(K_t)$ is either empty or lies on an arc \widehat{uv} between vertices u and v of t for every $t \in \text{conv}_B$. As a result we have two cases: (1) there exists a triangle t that is the only helper triangle on conv_B that shares u and v with K , and u and v are the endpoints of a helper arc $\alpha \subseteq B \cap B'$ for some protecting B' consecutive to B , or (2) there exist two triangles t_1 and t_2 that share u and v with K and $K \subseteq K_{t_1} \cup K_{t_2}$. In case (1), we use Claim 4.3.1 to show that S is enclosed by either $bd(B \cup K_t^\perp \cup K_\alpha^\perp)$ or $bd(B \cup K_\alpha^\perp \cup B')$, and hence some vertex of σ encroaches upon t or α , which is a contradiction. In case (2), we use Claim 4.3.2 to show that $bd(B \cup K_{t_1}^\perp \cup K_{t_2}^\perp)$ encloses S , and hence some vertex of σ encroaches upon t_1 or t_2 , which is also a contradiction. So, the result follows. \square

Lemma 4.3.4 and Lemma 4.3.5 together imply that the algorithm never inserts a vertex inside $Z(\mathcal{B})$ and, whenever no helper arc or helper triangle is wide or encroached, we have that the Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of P conforms to the sub-facets inside $Z(\mathcal{B})$, the tetrahedron $[p, q, b_1, b_2]$ belongs to $\mathcal{DT}_3(P)$ for any helper arc $\widehat{pq} \subseteq (B_1 \cup B_2)$ and protecting balls B_1 and B_2 centered at b_1 and b_2 , respectively, and the tetrahedron $[p, q, r, b_1]$ belongs to $\mathcal{DT}_3(P)$ for any protecting ball B_1 and helper triangle $[p, q, r]$ on $\text{conv}_{B_1} = \text{conv}(P \cup Z(\mathcal{B}) \cup B_1)$. Furthermore, whenever no helper arc or helper triangle is wide or encroached and no sub-facet is encroached, the Delaunay tetrahedrization $\mathcal{DT}_3(P)$ of P conforms to the input PLC \mathcal{C} .

4.3.4 Termination

The termination of the algorithm relies on the fact that the algorithm will not keep generating encroached helper arcs, helper triangles or sub-facets indefinitely. So, we must show that the algorithm executes only a finite number of steps to reach the state in which no helper arc or helper triangle is wide or encroached, no sub-facet is encroached, and no tetrahedron has large radius-edge ratio. This fact is shown by proving that the length of the shortest edge of the tetrahedrization cannot decrease after the execution of some finite number of steps.

Before stating the relevant lemmas for the proof of termination, we need introduce some notations. Consider the time the algorithm inserts or rejects a vertex v using rule i with $i \in \{1, 2, 3, 4\}$. We say that v has type i and we define

the *parent* of v as follows. If v is the center of K_τ for a wide helper arc or triangle τ , the parent of τ is undefined. Suppose that v is the center of K_τ where τ is a non-wide encroached helper arc or helper triangle or encroached sub-facet, then the parent of v is the nearest encroaching vertex $p \in P$. Note that the vertex p may very well be a rejected vertex. If v is the circumcenter of a tetrahedron σ , then the parent of v is the endpoint of the shortest edge of σ that appeared in P the latest. If v is a vertex of Q then the parent of v is undefined. For each vertex $v \in P$, we define the *insertion radius* r_v of v in a way much like before. If v is a vertex of Q then r_v is the minimum distance from v to another vertex of Q . If v is inserted or rejected by the algorithm then r_v is the minimum distance to a vertex in P at the time when v was inserted or rejected.

Claim 4.3.4. [8] *Let K be a cap with angular diameter at most $\frac{\pi}{3}$. Let v be the center of K . For any point p inside K^\perp and any point q on or outside K^\perp , we have $d(q, v) > \frac{1}{4} \cdot \max\{d(p, v), d(p, q)\}$.*

Proof. Let z be the center of K^\perp . Since the angular diameter of K is at most $\frac{\pi}{3}$, we have $d(v, z) < R \cdot \cos(\frac{\pi}{3})$, where R is the radius of K^\perp . Since $R \leq d(q, z)$, we have $d(v, z) \leq d(q, z)/2$. By the triangle inequality, we get $d(q, v) \geq d(q, z) - d(v, z)$. It follows that $d(q, v) > d(q, z)/2$. Since p and v lie inside K^\perp , we get $d(q, z) \geq d(p, z)/2$. Thus, $d(q, v) > d(p, v)/4$. By the triangle inequality, $d(p, q) \leq d(p, z) + d(q, z) \leq 2 \cdot d(q, z)$. Thus, $d(q, v) > d(p, q)/4$. \square

Lemma 4.3.6. [8] *Let v be a vertex of Q or a vertex inserted or rejected by the algorithm, let p be the parent of v , and let $lfs : \mathbb{R}^3 \mapsto \mathbb{R}$ be the local feature size function defined with respect to Q (not C). Then, the following hold:*

1. *If p is undefined then $r_v \geq lfs(v)/2$, where r_v is the insertion radius of v .*
2. *Otherwise, $r_v > d(p, v)/4$ and if $r_v \leq lfs(v)/4$, then $r_v > r_p/4$ if v has type 1, 2, or 3, and $r_v > \rho_0 \cdot r_p$ if v has type 4, where r_v and r_p are the insertion radius of v and p , respectively.*

Proof. We just sketch the proof here. See [8] for details. If v is a vertex of Q then the parent p of v is undefined and $r_v \geq lfs(v)$ by the definition of insertion radius and lfs . If v is not a vertex of Q , we have four cases: (1) the vertex v is the center of K_τ for a wide helper arc or triangle τ ; (2) the vertex v is the midpoint of a non-wide encroached helper arc α ; (3) the vertex v is the center of K_τ where τ is a non-wide encroached helper triangle or an encroached sub-facet; and (4) the vertex v is the circumcenter of a tetrahedron σ . In case (1), the parent p of v is undefined, and we can show that $r_v \geq lfs(v)/2$. In case (2), the vertex v has type 1. Let q be the vertex of P such that $r_v = d(q, v)$. If q lies inside K_τ^\perp then q is the parent p of v . Otherwise, from Claim 4.3.4, $d(q, v) > d(p, v)/4$. Hence, in either case, $r_v = d(q, v) > d(p, v)/4$. Next, we can show that if $r_v \leq lfs(v)/4$ then $r_v > r_p/4$. In case (3), we can show that $r_v = d(q, v) > d(p, v)/4$, for a vertex q inside K_τ^\perp such that $r_v = d(q, v)$, using a similar argument

to the one in Case (2). The parent p of v can be a vertex of Q or a vertex of type 1, 2, or 3. If p is a vertex of Q or a vertex of type 1, 2, or 3, we can show that $r_v > lfs(v)/4$. Otherwise, the vertex p has type 4, and we can show that $r_v \leq lfs(v)/4$ and $r_v > r_p/4$. In case (4), by definition, the vertex p is an endpoint of the shortest edge of σ . Let q be the other endpoint of this edge. If p is a vertex of Q , by the definition of parent, the vertex q is also a vertex of Q . So, $r_v = d(p, v) = d(q, v) = lfs(v)$. If p is not a vertex of Q , since $\rho(\sigma) > \rho_0$, we have $r_v = d(p, v) > \rho_0 \cdot d(p, q) \geq \rho_0 r_p$. \square

Lemma 4.3.6 limits how quickly the insertion radius of a vertex v of $\mathcal{DT}_3(P)$ can decrease, and it is the analogous of Lemma 3.3.2 used by the proof of termination of Shewchuk's algorithm in Subsection 3.3.3. We now show that, by setting to the constant ρ_0 an appropriate value, the insertion radius of v cannot decrease indefinitely. Because the new vertices are always inserted into some bounded, convex region enclosing the complex Q , the algorithm can only insert a finite amount of vertices into P if the distance between any two vertices of P is bounded from below by some constant $d > 0$. So, the algorithm must terminate after a finite number of steps.

Claim 4.3.5. *Let v be a vertex inserted or rejected by the algorithm, and let p be the parent of v . If $r_v > c \cdot r_p$ for some constant $c \in \mathbb{R}$ then $lfs(v) < (lfs(p) \cdot r_v / (c \cdot r_p)) + 4 \cdot r_v$, where $lfs : \mathbb{R}^3 \mapsto \mathbb{R}$ is the local feature size function with respect to Q .*

Proof. Since p is undefined, from Lemma 4.3.6, we have $r_v > d(p, v)/4$. By the Lipschitz property of lfs , we have $lfs(v) \leq lfs(p) + d(p, v)$. Thus, $lfs(v) \leq lfs(p) + d(p, v) < lfs(v) + 4 \cdot r_v < (lfs(p) \cdot r_v / (c \cdot r_p)) + 4 \cdot r_v$. \square

Lemma 4.3.7. [8] *Let v be a vertex of Q or a vertex inserted or rejected by the algorithm. If v is a vertex of Q then $r_v \geq lfs(v)$. Otherwise, there are four constants C_1, C_2, C_3 , and C_4 depending on ρ_0 such that $C_1 > C_2 = C_3 > C_4 > 4$ and $r_v > lfs(v)/C_i$, where $i \in \{1, 2, 3, 4\}$ is the type of v .*

Proof. Let $C_1 = 84/(\rho_0 - 16)$, $C_2 = C_3 = (20\rho_0 + 16)/(\rho_0 - 16)$ and $C_4 = (4\rho_0 + 20)/(\rho_0 - 16)$. Before the algorithm inserts the first vertex into P , the set P contains the vertices of Q only. By definition of r_v and lfs , we have $r_v \geq lfs(v)$ for every vertex v of Q . We now use induction on the number n of vertices inserted into P to show that our claim is true after the algorithm starts inserting vertices in P . For $n = 1$, the vertex v has type 1, 2, 3 or 4, and the parent p of v is a vertex of Q . So, from Lemma 4.3.6, if v has type 1, 2, or 3, we have that $r_v > lfs(v)/4$, and our claim is true as $C_4 = 4$. If v has type 4, by Lemma 4.3.6, then $r_v > \rho_0 r_p$. Since p is a vertex of Q , we have $r_p \geq lfs(p)$. From Claim 4.3.5, we have $lfs(v) < r_v/\rho_0 + 4 \cdot r_v < (C_1 \cdot r_v)/\rho_0 + 4 \cdot r_v = C_4 r_v$. So, our claim is true for $n = 1$. Assume that our claim also holds for $n = k \geq 1$ and consider the case $n = k + 1$. If $r_v > lfs(v)/4$,

we are done as $C_4 > 4$. Otherwise, Lemma 4.3.6 implies that the parent p of v is defined. From 4.3.6 again, if v has type 1 then p has type 1, 2, 3, or 4 and $r_v > r_p/4$. By the induction hypothesis, we have $r_p > lfs(p)/C_2$. From Claim 4.3.5, we have $lfs(v) < (lfs(p) \cdot r_v / (c \cdot r_p)) + 4 \cdot r_v < 4C_2r_v = C_1r_v$. If v has type 2 or 3, by Lemma 4.3.6, the vertex p has type 4 and $r_v > r_p/4$. By the induction hypothesis, $lfs(p) < C_4r_p$. From Claim 4.3.5, we have $lfs(v) < 4C_4r_v + 4r_v = C_2r_v$. If v has type 4, by Lemma 4.3.6, then $r_v > \rho_0r_p$. By the induction hypothesis, $lfs(p) < C_1r_p$ regardless of whether p is a vertex of Q or p was rejected or inserted by the algorithm. From Claim 4.3.5, we get $lfs(v) < C_1r_v/\rho_0 + 4r_v = C_4r_v$. So, our claim holds. \square

Theorem 4.3.1. [8] *The algorithm terminates and, for each output vertex v , its shortest incident edge has length at least $lfs(v)/(1 + C_1)$.*

Proof. Let $[u, v]$ be the shortest edge incident to v . If u appeared in P no later than v then $d(v, u) \geq r_v \geq lfs(v)/C_1$, by Lemma 4.3.7. Likewise, if v appeared in P before u then $d(u, v) \geq r_u \geq lfs(u)/C_1$. By the Lipschitz property of lfs , we have $lfs(v) \leq lfs(u) + d(u, v) \leq (1 + C_1) \cdot d(u, v)$. So, $d(u, v) \geq lfs(v)/(1 + C_1)$. The edge length bound implies that we can center disjoint balls at the output vertices with radii $lfs_{\min}/(2 + 2C_1)$, where lfs_{\min} is the minimum local feature size with respect to Q . Since $lfs_{\min} > 0$ and the input domain has bounded area, there can be only finitely many output vertices. It follows that the algorithm terminates. \square

4.4 Concluding Remarks

The algorithm due to Xiang-Yang Li [6] solved a long standing open problem: Provably guaranteed elimination of slivers. However, his algorithm can only deal with input PLCs satisfying the angle constraint. In addition, the parameter φ_0 that defines what a sliver is may be too small for any practical use. Some experiments could help find a larger value of φ_0 for which no small sliver is created, but Li has not reported on a possible implementation of his algorithm so far.

The algorithm due to Siu-Wing Cheng and Sheung-Hung Poon [8] is an extension of Shewchuk's algorithm that can deal with input PLCs that may not satisfy the angle constraint. Their algorithm is provably guaranteed to generate an output tetrahedrization whose tetrahedra have bounded radius-edge ratio, and that is the main advantage of this algorithm over the algorithm due to Cohen-Steiner, Verdière, and Yvinec [7]. However, the fact that ρ_0 is greater than 16 may prevent their algorithm from being used in practice, as the radius-edge ratio may still be too large for $\rho_0 > 16$. Shewchuk's algorithm uses $\rho_0 > 2$. Because Siu-Wing Cheng and Sheung-Hung Poon have not

reported on an implementation of their algorithm, it is not possible to evaluate the impact of a value $\rho_0 > 16$ over the tetrahedrization quality.

The complexity of Li's algorithm and Cheng and Poon's algorithm was not analyzed in their papers. As we mentioned before, it is an open problem to find a polynomial upper bound, if any, for the number of insertion points needed to guarantee boundary conformity of input PLCs. It also remains an open problem to develop an algorithm for constructing conforming Delaunay tetrahedrizations free of slivers and with provably guaranteed bounded radius-edge ratio that can handle PLCs that may not satisfy the angle constraint.

Chapter 5

Conclusion

We reviewed four algorithms for constructing conforming Delaunay tetrahedrizations. All algorithms are provably guaranteed to terminate and to generate a correct output if the input constraints are respected. The first algorithm is due to Jonathan Shewchuk, [5]. His algorithm is the first one *based on Delaunay refinement* to generate conforming Delaunay tetrahedrization with theoretical bounds on the radius-edge ratio of the output tetrahedra and guarantee of good grading. Shewchuk also implemented his algorithm and provided several output examples that showed its value in practice. The main weakness of his algorithm is its requirement that the input PLC satisfy the angle constraint.

The second algorithm we studied is due to Cohen-Steiner, Verdière, and Yvinec, [7]. The main contribution of their algorithm is the ability to handle input PLCs that may not satisfy the angle constraint. For this purpose, they devised a scheme of edge protection similar to the one created by Murphy, Mount, and Gable, [16]. The edge protection scheme used by Cohen-Steiner, Verdière, and Yvinec enabled them to develop a Delaunay refinement algorithm that used fewer extra vertices than the one by Murphy, Mount, and Gable. Cohen-Steiner, Verdière, and Yvinec also implemented their algorithm and showed its application to some complex examples. The main weakness of their algorithm is the fact that it does not provide any quality guarantee for the output tetrahedrization. They did not report on the quality of the output tetrahedra in the examples provided in [7].

The third algorithm we examined is due to Xiang-Yang Li, [6]. His algorithm is an extension of Shewchuk's algorithm to generate conforming Delaunay tetrahedrizations that also satisfy the aspect ratio quality metric. Tetrahedrizations with bounded radius-edge ratio and bounded aspect ratio have no tetrahedra with poor angles, and thus they are very desirable in practical applications. Like Shewchuk's algorithm, Li's algorithm also produces good graded tetrahedrizations, but it is also limited to input PLCs that satisfy the angle constraint. Besides, the theoretical

bounds for the aspect ratio given by Li do not seem to be very attractive in practice. An experimental study of his algorithm could certainly provide a good idea of the practical value of his algorithm. Unfortunately, such an implementation has not been reported so far.

The fourth and last algorithm we described is due to Siu-Wing Cheng and Sheung-Hung Poon, [8]. Their algorithm is also an extension of Shewchuk's algorithm, but at this time the extension is to handle input PLCs that may not satisfy the angle constraint. Like Cohen-Steiner, Verdière, and Yvinec's algorithm, Cheng and Poon's algorithm also uses an edge protection scheme to guarantee termination of the algorithm when dealing with PLCs that do not satisfy the angle constraint. The main difference between Cheng and Poon's algorithm and the one by Cohen-Steiner, Verdière, and Yvinec is that the former also gives a theoretical bound on the radius-edge ratio of output tetrahedra. However, this bound is looser than the one by Shewchuk's algorithm, and it also does not seem to be very attractive in practice. Again, an experimental study of their algorithm would be very useful to evaluate its practical value, but such an implementation has not been reported so far either. Like Shewchuk's algorithm, both Li's algorithm and Cheng and Poon's algorithm produce good graded tetrahedrizations.

Several problems related to the algorithms studied here remain open. From the practical point of view, the most important one is likely to be the development of an algorithm for constructing conforming Delaunay tetrahedrizations of any input PLC with bounded radius-edge ratio and bounded aspect ratio. One such an algorithm could be the result of the creation of an edge protection scheme that allowed for the use of ideas similar to the ones behind Li's algorithm. From the theoretical point of view, the most important problem is to carefully analyze the time and size complexities of the four algorithms aforementioned. Shewchuk, [5], was the only author to mention algorithm complexity in the papers we reviewed. He said that his algorithm has a worst-case time complexity of $O(n^2)$ per vertex insertion, where n is the number of vertices in the current tetrahedrization. The difficulty is that we do not know how many vertices we will insert to enforce boundary conformity and to satisfy quality constraints. We do not even know if there is a polynomial bound for the total number of inserted vertices. All four algorithms we reviewed can indeed produce a tetrahedrization whose size is exponential with respect to the number of vertices of the input PLC.

As one might expect, conforming Delaunay tetrahedrizations are not the only approach for reconciling boundary conformity and quality constraints. The most common approach is actually the *almost Delaunay tetrahedrization*, [18]. In this approach, boundary conformity is also enforced by inserting new vertices in an initial Delaunay tetrahedrization, but the final tetrahedrization does not have to be a Delaunay one. Almost Delaunay tetrahedrization algorithms are in general based on heuristics, and these heuristics are not guaranteed to work well in the presence of complex input PLCs. Mitchell and Vavasis, [19], gave an algorithm based on octrees that is provably guaranteed to

generate conforming tetrahedrizations with bounded dihedral angles and good graded, but its bounds are too weak for practical use.

More recently, Shewchuk also proposed another approach, called *constrained conforming Delaunay tetrahedrization*, [4]. Like the almost Delaunay approach, this one also starts with a Delaunay tetrahedrization of the vertices of the input domain and enforces boundary conformity by inserting vertices into it. The final tetrahedrization is not necessarily a Delaunay tetrahedrization either. His approach, however, gives provable bounds on quality constraints, and it seems to need fewer vertices than the conforming Delaunay tetrahedrization approach in order to enforce boundary conformity and meet quality constraints. But, like the algorithms we studied here, there is no known polynomial upper bound on the number of extra vertices inserted into the initial tetrahedrization.

Bibliography

- [1] P.C.P. Carvalho, J.M. Gomes, and L. Velho. Space Decompositions: Theory and Practice. Technical report, Instituto de Matemática Pura e Aplicada, Rio de Janeiro (RJ), Brazil, 1992.
- [2] T. Dey, C. Bajaj, and K. Sugihara. On Good Triangulations in Three Dimensions. *International Journal of Computational Geometry & Applications*, 2(1):75–95, 1992.
- [3] P. Chew. Guaranteed-Quality Delaunay Meshing in 3D. In *Proceedings of the 13th Annual Symposium on Computational Geometry*, pages 391–393, 1997.
- [4] J. Shewchuk. Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery. In *Proceedings of the 11th International Meshing Roundtable*, pages 193–204, 2002.
- [5] J. Shewchuk. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proceedings of the 14th Annual Symposium on Computational Geometry*, pages 86–95, 1998.
- [6] X-Y Li. Spacing Control and Sliver-Free Delaunay Mesh. In *Proceedings of the 9th International Meshing Roundtable*, pages 295–306, 2000.
- [7] D. Cohen-Steiner, E. Verdière, and M. Yvinec. Conforming Delaunay Triangulations in 3D. In *Proceedings of the 18th Annual Symposium on Computational Geometry*, pages 199–208, 2002.
- [8] S-W. Cheng and S-H. Poon. Graded Conforming Delaunay Tetrahedralization with Bounded Radius-Edge Ratio. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 295–304, 2003.
- [9] J-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, Cambridge, UK, 1998.
- [10] H. Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, Cambridge, UK, 2001.

-
- [11] J. Gallier. *Geometric Methods and Applications for Computer Science and Engineering*. Springer-Verlag, New York, NY, 2000.
- [12] A. Okabe, B. Boots, K. Sugihara, and S. Chiu. *Spatial Tesselations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons Ltd, 2nd edition, 2000.
- [13] J. Ruppert and R. Seidel. On the Difficulty of Triangulating Three-Dimensional Nonconvex Polyhedra. *Discrete & Computational Geometry*, 7(3):227–254, 1992.
- [14] M. Bern and D. Eppstein. Mesh Generation and Optimal Triangulation. In F.K. Hwang and D.-Z. Du, editors, *Computing in Euclidean Geometry*. World Scientific, 1992.
- [15] B. Chazelle. Convex Partitions of Polyhedra: a Lower Bound and Worst-Case Optimal Algorithm. *SIAM Journal of Computing*, 13:488–507, 1984.
- [16] M. Murphy, D. Mount, and C. Gable. A Point-Placement Strategy for Conforming Delaunay Tetrahedralization. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 67–74, 2000.
- [17] J. Shewchuk. What is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *Proceeding of the 11th International Meshing Roundtable*, pages 115–126, 2002.
- [18] P-L. George and H. Borouchaki. *Delaunay Triangulation and Meshing: Application to Finite Elements*. Hermès, Paris, France, 1998.
- [19] S. Mitchell and S. Vavasis. Quality Mesh Generation in Three Dimensions. In *Proceedings of the 8th Annual Symposium on Computational Geometry*, pages 212–221, 1992.