Weierstraß-Institut
für Angewandte Analysis und Stochastik
Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

Higher-quality tetrahedral mesh generation for domains with small angles by constrained Delaunay refinement

Jonathan Richard Shewchuk[1], Hang Si[2]

submitted: June 26, 2014

[1] University of California at Berkeley
Department of Electrical Engineering and Computer Sciences
Berkeley
California 94720
email: jrs@cs.berkeley.edu

[2] Weierstrass Institute
Mohrenstr. 39
10117 Berlin
Germany
email: hang.si@wias-berlin.de

No. 1976
Berlin 2014

2010 *Mathematics Subject Classification.* 65M50, 65N50, 65D18.

Key words and phrases. constrained Delaunay triangulation, Delaunay refinement algorithm, tetrahedral mesh generation, computational geometry.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Abstract

Algorithms for generating Delaunay tetrahedral meshes have difficulty with domains whose boundary polygons meet at small angles. The requirement that all tetrahedra be Delaunay often forces mesh generators to overrefine near small domain angles—that is, to produce too many tetrahedra, making them too small. We describe a provably good algorithm that generates meshes that are constrained Delaunay triangulations, rather than purely Delaunay. Given a piecewise linear domain free of small angles, our algorithm is guaranteed to construct a mesh in which every tetrahedron has a radius-edge ratio of $2\sqrt{2/3} \doteq 1.63$ or better. This is a substantial improvement over the usual bound of 2; it is obtained by relaxing the conditions in which boundary triangles are subdivided. Given a domain with small angles, our algorithm produces a mesh in which the quality guarantee is compromised only in specific places near small domain angles. We prove that most mesh edges have lengths proportional to the domain's minimum local feature size; the exceptions span small domain angles. Our algorithm tends to generate meshes with fewer tetrahedra than purely Delaunay methods because it uses the constrained Delaunay property, rather than vertex insertions, to enforce the conformity of the mesh to the domain boundaries. An implementation demonstrates that our algorithm does not overrefine near small domain angles.

1 Introduction

Delaunay refinement algorithms for tetrahedral mesh generation [5, 12, 20, 24, 25] offer mathematical guarantees on the quality of the tetrahedra they produce, and they have proven to be popular and effective in practice for generating meshes suitable for finite element and finite volume methods. However, these methods have difficulty in theory and practice when meshing domains whose facets or edges meet at small angles. A primary difficulty is that Delaunay triangulations do not naturally respect the boundaries of a nonconvex domain, especially if the domain has *internal boundaries* (e.g. separating different materials in a heat conduction simulation) to which the mesh must conform.

Delaunay mesh generators insert additional vertices that force a Delaunay triangulation to conform to the input domain. Even if we disregard the quality of the tetrahedra, constructing what is known as a *conforming Delaunay triangulation* of a polyhedral domain with internal boundaries is challenging. The problem has received some attention [9, 14], but no solution is known for which the number of added vertices is polynomial in the size of the domain description. The mesh generation problem, in which tetrahedron quality is not disregarded, has received yet more attention [6, 4, 15, 16]. Some authors replace Delaunay triangulations with weighted Delaunay triangulations [2, 3, 5], which help to reduce the number of added vertices. In quality mesh generation, the number of vertices is not expected to be polynomial in the size of the input description, but the edges of the mesh should not be much shorter than they must be to accommodate the domain geometry and the user's wishes. Especially for domains with small angles, it is crucial to prevent *overrefinement*, wherein the edges are shorter and the tetrahedra more numerous than desired.

In this paper, we advocate using constrained Delaunay triangulations (CDTs, defined in Section 2) to enforce domain conformity in guaranteed-quality tetrahedral meshing. A CDT has the advantage that once enough vertices have been inserted to recover the edges of the domain, no additional vertices are needed to recover its polygonal facets (although some are usually needed to improve the quality of the tetrahedra).

Two-dimensional CDTs are widely and successfully used in algorithms for triangular mesh generation [1, 5, 7, 8, 22] because they enforce domain conformity with no need to add any new vertices. In three dimensions, the advantages of CDTs are less clear-cut. A difficulty of working with CDTs is that not every polyhedron has one—there exist simple polyhedra with no tetrahedralization at all [18]. Overcoming this difficulty requires added vertices. However, for domains with internal boundaries meeting at small angles, the number of added vertices can be far fewer than a conforming Delaunay triangulation would require.

A mathematical difficulty that all mesh generators face, Delaunay or not, is that for some domains with small angles, no algorithm can guarantee that every tetrahedron will have high quality. It is not possible to place good tetrahedra at points where boundary polygons meet at a tiny angle, of course; worse yet, for some domains, there exists no mesh in which even the tetrahedra not adjoining those points are all good [21]. Inherently, part of the problem of meshing domains with small angles is to decide where to let skinny tetrahedra survive in the output mesh. For example, Cheng et al. [5] allow poor-quality tetrahedra to adjoin protective weighted vertices placed at small domain angles, but all the other tetrahedra are good.

This paper makes three main contributions. First, we devise a mesh generation algorithm that, given a piecewise linear domain free of small angles, constructs a mesh in which every tetrahedron has a radius-edge ratio of $2\sqrt{2/3} \doteq 1.63$ or better. (The *radius-edge ratio* of a tetrahedron is its circumradius divided by the length of its shortest edge. Its *circumradius* is the radius of its circumscribing sphere. The radius-edge ratio has become a standard measure of tetrahedron quality in Delaunay refinement algorithms. It is a flawed measure of quality, but the bad tetrahedra that escape it, called *slivers*, are relatively easy to eliminate in practice [20]—except near small domain angles.) This is a substantial improvement over the usual radius-edge ratio bound of 2 and the strongest bound on radius-edge ratios we know of for any tetrahedral mesh generation algorithm for polyhedra. The main insight is that the constrained Delaunay property allows us to relax the conditions in which triangles on the domain boundary are considered to be "encroached."

Second, given a domain with small angles, our algorithm produces meshes in which the quality guarantee is compromised only in specific places: where a tetrahedron intersects a boundary polygon or edge that meets another polygon or edge at an acute angle. Our implementation, discussed in Section 5, shows that for difficult domains, the use of a CDT helps us control the number of added vertices quite well.

Third, we provide theorems useful for understanding the refinement of constrained Delaunay meshes. Three-dimensional constrained Delaunay refinement is substantially harder to analyze than ordinary Delaunay refinement because the tetrahedra in a CDT are not guaranteed to have empty circumspheres. We overlooked this problem in our prior work on meshing with CDTs [21, 24]. Here, we develop a theory and an algorithm for attacking a skinny tetrahedron (specifically, by a careful treatment of the order in which encroached subsegments and subpolygons are attacked) that offers correctly proven guarantees on tetrahedron quality and thereby repairs the prior work. (The present algorithm is also more strict about where it permits poor tetrahedra to survive than many prior algorithms.)

2 Piecewise Linear Complexes and Constrained Delaunay Triangulations

The input to our meshing algorithm is a *piecewise linear complex* (PLC), introduced by Miller et al. [13]. PLCs generalize polyhedra to permit internal boundaries and other constraints. A three-dimensional PLC $\mathcal{X}$ is a set of vertices, edges, polygons (not necessarily convex), and polyhedra, collectively called *cells*, that satisfies two properties. (1) The boundary of each cell in $\mathcal{X}$ is a union of cells in $\mathcal{X}$. (2) If two distinct cells $F, G \in \mathcal{X}$ intersect, their intersection is a union of cells in $\mathcal{X}$, all having lower dimension than at least one of F or G. The *underlying space* of $\mathcal{X}$, denoted $|\mathcal{X}|$, is $\bigcup_{F \in \mathcal{X}} F$, which is usually the domain to be triangulated. PLCs permit vertices and segments to float in the relative interior of a polygon or polyhedron to ensure that a triangulation of the PLC will support boundary conditions applied there. See elsewhere [5, 23] for details.

The segments and polygons in $\mathcal{X}$ constrain how $\mathcal{X}$ can be triangulated. A *triangulation* of $\mathcal{X}$ is a simplicial complex $\mathcal{T}$ such that (1) $\mathcal{X}$ and $\mathcal{T}$ have the same vertices, (2) every cell in $\mathcal{X}$ is a union of simplices in $\mathcal{T}$, and (3) $|\mathcal{T}| = |\mathcal{X}|$. A *mesh of* $\mathcal{X}$ is a triangulation of $\mathcal{X} \cup S$, where $S \subset |\mathcal{X}|$ is a set of *Steiner points* disjoint from $\mathcal{X}$'s vertices. A triangulation of $\mathcal{X}$ does not permit added vertices, whereas a mesh of $\mathcal{X}$ does.

A mesh $\mathcal{T}$ of $\mathcal{X}$ subdivides each polygon in $\mathcal{X}$ into triangles in $\mathcal{T}$, and each edge in $\mathcal{X}$ into edges in $\mathcal{T}$. We call the edges in a PLC *segments* to distinguish them from the edges in the mesh. An edge in $\mathcal{T}$ is a *subsegment* if it is included in a segment. A triangle in $\mathcal{T}$ is a *subpolygon* if it is included in a polygon.

Two points x and y are *visible* to each other if the open line segment xy does not intersect a polygon in $\mathcal{X}$, excepting polygons that x or y is coplanar with. A polygon in $\mathcal{X}$ that xy crosses (i.e. intersects though neither x nor y lie on the polygon's affine hull) is said to *occlude* the visibility between x and y. A tetrahedron $t \in \mathcal{T}$ is *constrained Delaunay* if the circumsphere (circumscribing sphere) of t encloses no vertex in $\mathcal{X}$ that is visible from a point in the interior of t. A *constrained Delaunay triangulation* (CDT) of $\mathcal{X}$ is a triangulation of $\mathcal{X}$ in which every tetrahedron is constrained Delaunay. A CDT of $\mathcal{X}$ does not permit added vertices. A *Steiner CDT* of $\mathcal{X}$ is a CDT of $\mathcal{X} \cup S$, where $S \subset |\mathcal{X}|$ is a set of Steiner points. Our mesh generation algorithm constructs a Steiner CDT of the input PLC.

Our algorithm relies on the *CDT Theorem* [19, 23], which provides a useful sufficient condition for a PLC (or a polyhedron) to have a CDT. A segment $e \in \mathcal{X}$ is *strongly Delaunay* if there exists a closed ball whose boundary passes through e's two vertices, but the ball contains no other vertex in $\mathcal{X}$. (This is a slightly stronger condition than e being Delaunay, which requires only that no vertex lie in the ball's interior.) A PLC is *edge-protected* if all its segments are strongly Delaunay. The CDT Theorem states that every edge-protected PLC has a CDT.

Let F be a segment, polygon, subsegment, or subpolygon. For $p \in \mathbb{R}^3$, $\mathrm{proj}_F(p)$ denotes the orthogonal projection of p onto the affine hull of F—that is, the point nearest p on the affine hull. Two adjoining cells F and G in a PLC $\mathcal{X}$ are said to satisfy the *projection condition* if $\mathrm{proj}_F(G) = \{\mathrm{proj}_F(p) : p \in G\}$ does not intersect $F \setminus G$, and $\mathrm{proj}_G(F)$ does not intersect $G \setminus F$. (It is trivially satisfied for the vertices in $\mathcal{X}$.) $\mathcal{X}$ satisfies the projection condition if every pair of adjoining cells in $\mathcal{X}$ does. Roughly speaking, this rules out cells meeting at acute angles. For such a PLC, the standard Delaunay refinement algorithm [20] is guaranteed to produce a mesh of tetrahedra whose radius-edge ratios do not exceed 2. Our goal is to mesh PLCs that fail the projection condition, for which the standard algorithm often fails to terminate at all.

The lengths of the edges in a high-quality mesh are largely determined by user-specified upper bounds and the geometry of the domain; small gaps between PLC cells necessitate short mesh edges nearby. The effect of geometry on the edge lengths is roughly captured by the well-known *local feature size* of a PLC $\mathcal{X}$, a function $\mathrm{lfs} : \mathbb{R}^3 \to \mathbb{R}$ such that $\mathrm{lfs}(x)$ is the radius of the smallest ball centered at x that intersects two disjoint cells in $\mathcal{X}$. Let $\mathrm{lfs}_{\min} = \min_{p \in |\mathcal{X}|} \mathrm{lfs}(p)$.

3 A Constrained Delaunay Refinement Algorithm

Here, we describe an algorithm that generates a tetrahedral mesh by refining a CDT. The input is a PLC $\mathcal{X}$ and a positive constant B that specifies the maximum permitted radius-edge ratio for tetrahedra in the output mesh. We call a tetrahedron *skinny* if its radius-edge ratio exceeds B. The algorithm is guaranteed to terminate and produce a mesh if $B \geq 2\sqrt{2/3} \doteq 1.63$. If $\mathcal{X}$ satisfies the projection condition, the mesh has no skinny tetrahedra. If $\mathcal{X}$ fails, the mesh may have some, but a skinny tetrahedron can exist only if it adjoins (has at least one vertex lying on) the relative interior of a segment or polygon that fails the projection condition. Every other tetrahedron is guaranteed not to be skinny.

The refinement algorithm maintains the CDT of an augmented PLC $\mathcal{Y}$ as it adds new vertices to $\mathcal{Y}$. The PLC $\mathcal{Y}$ is $\mathcal{X}$ with additional vertices, so the CDT of $\mathcal{Y}$ is a Steiner CDT of $\mathcal{X}$. At times during refinement, $\mathcal{Y}$ might not have a CDT (even the initial PLC $\mathcal{X}$ might not have a CDT), but repeated applications of the forthcoming Rule 1 restore the edge-protected property to $\mathcal{Y}$. The algorithm updates the CDT (or computes it for the first time) whenever $\mathcal{Y}$ is edge-protected.

We employ refinement rules typical of tetrahedral Delaunay refinement: skinny tetrahedra are "split" by new vertices inserted at their circumcenters, and "encroached" subsegments and subpolygons are split likewise. The standard methods used to prove the correctness of Delaunay refinement algorithms serve as our inspiration for modifying the refinement rules so that small domain angles do not cause havoc.

For each vertex v in the mesh, its *insertion radius* r_v is the distance to the closest distinct vertex visible from v at the moment when v is first inserted into the PLC $\mathcal{Y}$. If $\mathcal{Y}$ has a CDT, r_v is, equivalently, the length of the shortest edge that initially adjoins v. This definition differs from that of most Delaunay refinement algorithms by considering visibility; CDTs do not connect vertices that cannot see each other.

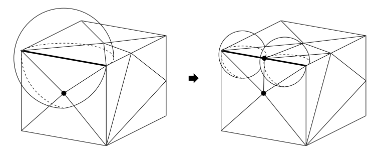
Standard analyses of Delaunay refinement algorithms rely on the relationships between the insertion radius of a newly inserted vertex and the insertion radius of some prior vertex to guarantee a lower bound on the lengths of all the edges created during refinement—specifically, a provably good refinement algorithm never creates an edge much shorter than the shortest edge in the initial triangulation. It eventually runs out of space to place new vertices, so it must terminate. But it does not terminate while skinny tetrahedra survive in the mesh—thus one can prove that it produces meshes free of skinny tetrahedra.

Unfortunately, the usual relationships between insertion radii do not hold where PLC polygons or segments meet at small angles. Sometimes it is necessary to insert a new vertex that creates a CDT edge that is much shorter than any prior edge. The central idea of our algorithm is to deprive those unreasonably short edges of the power to cause further refinement. Specifically, if a tetrahedron is skinny because it has an unreasonably short edge, we may decline to try to split the tetrahedron. This breaks an endless cycle wherein ever-shorter edges drive the creation of yet shorter edges. The cost is that some skinny tetrahedra survive in the final mesh, but only near small domain angles.

We implement this policy by storing for each vertex v a *relaxed insertion radius* rr_v, which always satisfies the constraint $rr_v \geq r_v$. For most vertices, including all input vertices, $rr_v = r_v$. However, when the algorithm is forced to create a new edge that it considers to be unreasonably short, the newly inserted vertex

v has rr_v greater than the length of that edge. This communicates to the algorithm that skinny tetrahedra having that edge should not be split if the splitting would create edges shorter than rr_v.

Specifically, when we compute the radius-edge ratio of a tetrahedron t, we pretend that t's shortest edge is not shorter than rr_v for the minimizing vertex v of t. The *relaxed shortest edge length* ℓ_t of t is either the length of t's shortest edge or the smallest relaxed insertion radius among t's vertices—whichever is *greater*. The *relaxed radius-edge ratio* of t is t's circumradius divided by ℓ_t. We say that t is *splittable* if its relaxed radius-edge ratio exceeds B. Every splittable tetrahedron is skinny (has a radius-edge ratio greater than B), but not every skinny tetrahedron is splittable. Our algorithm eliminates all tetrahedra that are splittable but not *fenced in* (a term we will define shortly).

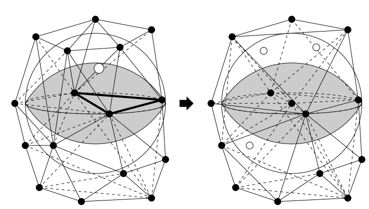


Whereas most Delaunay refinement algorithms define the insertion radius r_v as an analysis tool but do not compute it, our algorithm explicitly computes the relaxed insertion radius rr_v for each vertex v and stores it with v for future reference. Vertex insertion is governed by three rules.

Rule 1: Splitting encroached subsegments. The *diametric ball* of a subsegment is the unique smallest closed ball that includes the subsegment. We say that a subsegment is *encroached* if a vertex other than its endpoints lies in its diametric ball—even if the encroaching vertex is not visible from the subsegment. The algorithm splits any encroached subsegment that arises into two subsegments by inserting a new vertex, usually at its midpoint (but not always), as the figure shows.

Rule 2: Splitting encroached subpolygons. The *diametric ball* of a triangular subpolygon is the unique smallest closed ball whose boundary passes through the subpolygon's three vertices. The *diametric ellipsoid* is the diametric ball scaled by a factor of $1/\sqrt{3}$ in the direction orthogonal to the polygon. The subpolygon's circumcircle is the equator of the diametric ellipsoid. We depart from standard usage by declaring that the ellipsoid, like the ball, is a point set that includes all the points inside the ellipsoid too. The shape is chosen so that if a tetrahedron on one side of the equator has its circumcenter on the other side, either the circumcenter is in the ellipsoid or a vertex of the triangle is.

Usually, we say a subpolygon is *encroached* if a vertex other than its vertices lies in its diametric ellipsoid, but later we describe three circumstances in which a vertex is not eligible to encroach upon the subpolygon. Each subpolygon is a face of one or two tetrahedra in the CDT. A subpolygon is *immediately encroached* if the apex of one of those tetrahedra encroaches upon it. With some exceptions, discussed later, our algorithm usually ignores encroached subpolygons unless they are immediately encroached.



When no subsegment is encroached, the algorithm responds to an immediately encroached subpolygon by trying to split an encroached subpolygon—but not necessarily the same one. If a vertex p encroaches upon a subpolygon f of a polygon F, but the projected point $\mathrm{proj}_F(p)$ does not lie on f, then splitting f does not obtain the best guarantee of quality. Given that no subsegment of F is encroached, one can prove that $\mathrm{proj}_F(p)$ lies on F and that p also encroaches upon the subpolygon g of F that contains $\mathrm{proj}_F(p)$ [5, 20]. We usually split g in preference to f. The attempt to split g often eliminates f, but if it doesn't, the algorithm may try again with another subpolygon split.

We split a subpolygon by inserting a new vertex at its circumcenter and deleting all the vertices in its diametric ball that were inserted by Rule 3 and are visible from the new vertex. as illustrated. These vertices are deleted so that the new vertex will not adjoin unnecessarily short edges. (This idea was introduced by Chew [8] for triangular meshing.) However, if the new vertex would encroach upon a subsegment (visible or not), it is not inserted (and no vertex is deleted). Instead, a subsegment it would encroach upon is split by Rule 1. We say that the new vertex (circumcenter) has been *rejected*.

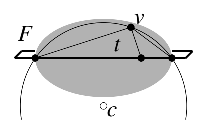
Rule 3: Splitting splittable tetrahedra. When no subsegment is encroached and no subpolygon is immediately encroached, the algorithm tries to split a splittable tetrahedron by inserting a new vertex at its

parent p new vertex v	type 3: tetra. circumcenter	type 2: subpolygon circumcenter	type 1: subsegment vertex or type 0: input vertex
3: tetrahedron circumcenter	$rr_v \leftarrow \lvert vp \rvert$	$rr_v \leftarrow \lvert vp \rvert$	$rr_v \leftarrow \lvert vp \rvert$
2: subpolygon circumcenter	$rr_v \leftarrow \min_w \lvert vw \rvert$ with w ranging over a and eligible neighbors in diametric ball		
1: subsegment vertex v not entwined with p v entwined with p	$rr_v \leftarrow \lvert va \rvert$	 $rr_v \leftarrow \lvert va \rvert$ $rr_v \leftarrow \max\{\lvert va \rvert, \min\{\frac{rr_p}{\sqrt{2}}, \min_w d_{vw}\}\}$	$rr_v \leftarrow \min_w d_{vw}$ with w ranging over all neighbors of v

If v is a subpolygon circumcenter, a is the subpolygon's nearest vertex and $|va|$ is the subpolygon's circumradius.
If v is a subsegment vertex, a and b are the subsegment endpoints with $|va| \leq |vb|$.
$d_{vw} = \max\{|vw|, rr_w\}$ if $w = a$ or $w = b$ or v and w are entwined; $d_{vw} = |vw|$ otherwise.

Table 1: How the constrained Delaunay refinement algorithm assigns a relaxed insertion radius rr_v to a new vertex v with parent p. Note that for the entries below the diagonal, the parent vertex p is rejected from the mesh.

circumcenter. Tetrahedra larger than the user desires are also split this way. However, if the new vertex would encroach upon a subsegment or subpolygon, then it is not inserted; instead, a subsegment or subpolygon it would encroach upon is split by Rule 1 or 2. If several subsegments and subpolygons would be encroached, the choice of which one to split is crucial; we discuss it in detail later. If the splittable tetrahedron is not eliminated as a side effect, the algorithm may try again to split it later.



There is one circumstance in which our algorithm declines to try to split a splittable tetrahedron. Let t be a tetrahedron with circumcenter c, and let q be an arbitrary point in the interior of t. Suppose c is not visible from q; one or more polygons occlude the visibility. Let F be the occluding polygon that intersects qc nearest q. Usually c is in the diametric ellipsoid of the subpolygon g of F containing $\mathrm{proj}_F(c)$; we split g and reject c. However, if c is so far to the other side of F that it does not encroach upon any subpolygon of F, then at least one vertex of t not on F must lie in g's diametric ellipsoid. However, some or all of t's vertices might be ineligible to encroach upon F because they lie on F or on a polygon or segment that meets F at a small angle. If c does not lie in any subpolygon's diametric ellipsoid and no vertex of t encroaches upon any subpolygon of F either, we say that t is *fenced in* and we do not attempt to split it. Our algorithm tries to split every splittable tetrahedron that is not fenced in, and terminates only when none remain.

Encroached subsegments (Rule 1) have priority over immediately encroached subpolygons (Rule 2), which have priority over splittable tetrahedra (Rule 3). A CDT might not exist when there are encroached subsegments; observe that Rule 1 makes no reference to the constrained Delaunay tetrahedra, and Rules 2 and 3 are executed only when a CDT exists. We update the CDT whenever no subsegment is encroached.

We say that a vertex is *of type i* if it is inserted by Rule i. Input vertices in $\mathcal{X}$ are of type 0.

Our algorithm must be careful in handling encroachment between segments or polygons that meet at small angles. Let F and G be two segments or polygons (possibly one of each, but the segment is not a subset of the polygon) that adjoin each other and do not respect the projection condition. If a vertex v lies on $F \setminus G$ and a vertex w lies on $G \setminus F$, with v and w each having type 1 or 2, we say that v and w are *entwined*. We also say that v and G are entwined; likewise that w and F are entwined; likewise that F and G are entwined.

There is a danger that these vertices might form an edge vw shorter than the shortest prior edge in the mesh. The hazard is great when one vertex is inserted because the other one is encroaching, then the new vertex encroaches back. The standard analysis of Delaunay refinement shows that this cycle of mutual encroachment cannot continue forever if no vertices are entwined (e.g. when $\mathcal{X}$ satisfies the projection condition). To break the cycle, we sometimes choose $rr_v > r_v$ for a new entwined vertex v.

We now consider Rules 1–3 in more detail and discuss how the algorithm chooses the relaxed insertion radius rr_v; these choices are summarized in Table 1.

Rule 1: Let e be an encroached subsegment. Our algorithm usually inserts a new vertex v at the midpoint of e, but it occasionally inserts v off-center. We use "modified segment splitting using concentric spherical shells," introduced by Ruppert [17], to prevent segments that meet at small angles from engaging in cycles of mutual encroachment that produce ever-tinier subsegments. Imagine that each input vertex is enclosed by concentric spheres whose radii are 2^i for all integers i. If e adjoins another segment at an acute angle, we split e not at its midpoint, but on one of the circular shells centered at the shared vertex, so that one of new subsegments has a power-of-two length. We choose the shell that gives the best-balanced split, so the two new subsegments are between one-third and two-thirds the length of the split subsegment. Each segment in $\mathcal{X}$ undergoes at most two unbalanced splits—one for each end—in which case all three subsegments are at least one-fifth the length of the original segment. All other subsegment splits are bisections.

If the encroaching vertex is not a circumcenter, e's diametric ball may contain multiple vertices, all of type 0 or 1. (Type 2 and 3 vertices would be rejected.) Let v be the vertex inserted on e, let a be the endpoint of e closer to v, and let b be the farther endpoint. Let $|vw|$ denote the distance from v to w. We define a *relaxed distance* d_{vw} between v and any other vertex w. For most vertices, $d_{vw} = |vw|$, but if v and w are entwined (implying that w is of type 1) or w is an endpoint of e, let $d_{vw} = \max\{|vw|, rr_w\}$. The algorithm sets $rr_v \leftarrow \min_w d_{vw}$, where w ranges over the vertices that are connected to v by edges of the CDT, including a and b. (If no CDT exists, w ranges over all the vertices in e's diametric ball.) The vertex p that provides the minimum value of d_{vp} is called the *parent of* v—the vertex held responsible for v's insertion. Observe that rr_v is the distance r_v from v to its nearest visible neighbor if that neighbor is not a, b, or entwined with v.

If a circumcenter p of type 2 or 3 encroaches upon e, p is rejected, but p is the parent of v. The diametric ball of e contains no vertex not on e—otherwise the algorithm would have split e before attempting to insert p. If p is not entwined with v, the algorithm sets $rr_v \leftarrow r_v = |va|$. If p is entwined with v (implying that p is of type 2), the algorithm sets $rr_v \leftarrow \max\{|va|, \min\{rr_p/\sqrt{2}, \min_w d_{vw}\}\}$ where w ranges over v's neighbors in the CDT. If v's insertion yields an updated PLC $\mathcal{Y}$ with no CDT, during the failed attempt to insert v we can nonetheless easily identify the vertices of the tetrahedra that are no longer constrained Delaunay.

Rule 2: Let f be a subpolygon of a polygon F. A vertex w in f's diametric ellipsoid encroaches upon f only if it is *eligible* to encroach upon F. Type 3 vertices (tetrahedron circumcenters) are always eligible. Type 0–2 vertices are *ineligible* in the following three circumstances:

- $w \in F$. (Note that a vertex $w \in F$ influences how F is subdivided into constrained Delaunay subpolygons, but w is not permitted to encroach upon those subpolygons.)
- w is of type 1, w is entwined with F, and the distance from w to F is less than $\sqrt{2}rr_w$.
- w is of type 2, w is entwined with F, and the distance from w to F is less than rr_w.

The last two disqualifiers prevent entwinement from leading to sequences of ever-shorter edges.

Suppose f is encroached and we insert a vertex v at the circumcenter of the subpolygon g that contains $\mathrm{proj}_F(w)$. Rule 2 is executed only when no subsegment is encroached, which implies that $v \in F$ [5, Lemma 6.2]. Thus a vertex inserted by Rule 2 always lies on the same polygon as the encroached subpolygon.

We delete from v's diametric ball any type 3 vertices visible from v. (It suffices to delete only those that would otherwise be connected to v by an edge of the CDT.) *After* deleting the type 3 vertices, we set $rr_v \leftarrow \min_w |vw|$, where w is chosen from among g's vertices and all the vertices in g's diametric ball that are connected to v by edges of the CDT and eligible to encroach upon F (even if they're not in the diametric ellipsoid and don't actually encroach). Observe that rr_v is the distance r_v from v to its nearest visible neighbor if that neighbor is not entwined with F.

The minimizing vertex w is the parent of v unless it is a vertex of g. In the latter case, rr_v is the radius of the diametric ball and the parent of v is the encroaching vertex p that triggered the subpolygon split. (Usually p is a rejected type 3 circumcenter, but occasionally p is a mesh vertex that is not connected to v because ineligible vertices block the way).

Observe an important difference between the treatment of subsegments and the treatment of subpolygons: a subsegment is always split if there is a vertex in its diametric ball, whereas we often decline to split a subpolygon if the vertices in its diametric ellipsoid are not visible or are too close to the subpolygon.

Rule 3: Let t be a splittable tetrahedron with circumsphere S and circumcenter c. We always set rr_c to be the radius of S. In the standard Delaunay refinement algorithm, r_c also is the radius of S, as S is empty. In a CDT, however, there might be vertices inside S, and these might be visible from c (albeit not from the interior of t), in which case $r_c < rr_c$. We will see that in that circumstance, c is always rejected on account of encroachment. However, we are in danger of creating indefensibly short edges by repeatedly splitting a small subsegment or subpolygon near c. One of our main contributions is a proof that we can always avoid that fate by splitting subsegments and subpolygons in the right order.

Our procedure for splitting a tetrahedron appears in Figure 1. If c encroaches upon multiple subsegments or subpolygons, the main goal of SPLITTETRAHEDRON is to find a subsegment whose diametric ball's radius is at least $rr_c/\sqrt{2}$ or a subpolygon whose diametric ball's radius is at least $\sqrt{3}rr_c/2$ to help guarantee that the final mesh has no skinny tetrahedra. Usually, SPLITTETRAHEDRON achieves this goal by identifying a subsegment or subpolygon that has no vertex inside S. Occasionally, SPLITTETRAHEDRON meets the goal by identifying a subpolygon that is sufficiently far from c.

SPLITTETRAHEDRON has several subsidiary goals. We prefer to split encroached subsegments over encroached subpolygons, except that we prefer to split subpolygons that are partly or fully visible from c over fully occluded subsegments. If polygons subdivide the domain into multiple chambers, we try to split a subsegment or subpolygon in the same chamber as t, even if c is in a different chamber.

We have embedded a proof of the procedure's correctness as comments in the pseudocode that explain the theoretical justification for each step. The theory it relies on (the lemmas in the appendix) form a foundation for constrained Delaunay refinement that we hope will enable further developments.

4 Correctness and Guarantees of the Refinement Algorithm

Every vertex v of type 1–3, inserted or rejected, has a *parent* vertex $p(v)$. Parents for type 1 and 2 vertices are defined in Section 3. For a type 3 circumcenter v of a splittable tetrahedron t, $p(v)$ is the vertex of t with the smallest relaxed insertion radius rr_p. Every vertex v has an insertion radius r_v, even a rejected vertex, for which r_v is the distance to the nearest distinct vertex visible from v at the moment when v was rejected. Our algorithm assigns a relaxed insertion radius rr_v to every mesh vertex and rejected circumcenter.

The success of our algorithm follows from the fact that the relaxed insertion radii obey the same inequalities that the insertion radii obey for domains that satisfy the projection condition.

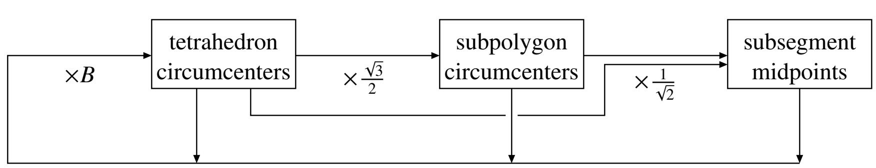
Lemma 1. *Let v be a vertex (inserted or rejected), and let $p = p(v)$ be its parent.*

i. If v is of type 3, then $rr_v > Brr_p \geq Br_p$ (by the definition of splittable).

ii. If v is of type 2 and p is of type 3, then $rr_v \geq \sqrt{3}rr_p/2$.

iii. If v is a type 1 midpoint (not inserted off-center) and p is of type 3, then $r_v \geq rr_p/\sqrt{2}$.

iv. If v is a type 1 midpoint of a segment s and p is of type 2 on a polygon $F \supset s$, then $r_v \geq rr_p/\sqrt{2}$.



The flow graph at right represents Lemma 1. Type 3 circumcenters can father type 2 circumcenters whose relaxed insertion radii are smaller by a factor of $\sqrt{3}/2$, which in turn can father type 1 midpoints whose insertion radii are smaller by another factor of $1/\sqrt{2}$. To avoid spiralling into the abyss, we insist that no cycle in the graph have a product less than one. This constraint fixes the best guarantee on the relaxed radius-edge ratios at $B = 2\sqrt{2/3}$.

For an input PLC $\mathcal{X}$ that satisfies the projection condition, the inequalities in Lemma 1 make it possible to put a lower bound on the insertion radius of every vertex. Without the projection condition, vertices with

SplitTetrahedron($t, \mathcal{T}, \mathcal{X}$) { Split a tetrahedron t in a Steiner CDT $\mathcal{T}$ of $\mathcal{X}$. }
1 Let S and c be the circumsphere and circumcenter of t. Let q be any point in the interior of t.
2 Locate c in $\mathcal{T}$ by a straight-line walk from q to c. (Stop the walk if it strikes a subpolygon.)
3 **if** c is visible from q
{ As q sees c, by Lemma 5, no mesh vertex inside S is visible from c. }
{ Therefore, a subsegment or subpolygon that is fully visible from c has no vertex inside S. }
4 **if** c encroaches upon a subsegment $e \in \mathcal{T}$ that is fully visible from c
5 SplitSubsegment(e). **return**.
{ At this point, by the contrapositive of Lemma 4, a subsegment can be encroached upon by c only if }
{ it is fully occluded from c. We prefer to split visible subpolygons over occluded subsegments. }
6 **if** c encroaches upon a subpolygon $f \in \mathcal{T}$ that is fully visible from c and contains $\text{proj}_f(c)$
7 SplitSubpolygon(f). **return**.
{ At this point, by Lemma 8, no subsegment or subpolygon intersects the inner half of S's radius. }
{ Therefore, if c encroaches upon a subpolygon g that contains $\text{proj}_g(c)$, g must have a circumradius }
{ at least $\sqrt{3}/2$ times the radius of S and be safe to split. }
8 **if** c encroaches upon a subpolygon $f \in \mathcal{T}$ that is partly or fully visible from c
9 Let $F \in \mathcal{X}$ be the polygon that includes f.
10 Locate the subpolygon g of F that contains $\text{proj}_F(c)$ by a straight-line walk on F from f.
{ As c encroaches upon f, c also encroaches upon every subpolygon and subsegment that inter- }
{ sects a straight-line walk from f to $\text{proj}_F(c)$ by the Monotone Power Lemma [5, Lemma 7.5]. }
11 **if** the walk strikes a subsegment $e \subset F$ before reaching $\text{proj}_F(c)$
12 SplitSubsegment(e). **return**. { Note: e might not be visible from c. That's okay. }
13 **else** SplitSubpolygon(g). **return**. { Note: g might not be visible from c. That's okay. }
{ Unfortunately, an encroached subsegment can be entirely occluded by an unencroached subpolygon. }
14 **if** c encroaches upon a subsegment $e \in \mathcal{T}$ that is occluded from c
15 SplitSubsegment(e), but if the occluding polygon is entwined with e, pretend that c is also entwined for the purpose of computing rr_v for the new vertex v. **return**.
{ c might encroach upon a subpolygon that is fully occluded from c, but we don't care. }
16 Insert c into $\mathcal{T}$. Set $rr_c \leftarrow$ the radius of S. **return**.
17 **else** { c is not visible from q; c will not be inserted. }
18 Let $f \in \mathcal{T}$ be the subpolygon intersecting qc nearest q (blocking c's visibility from q).
19 Let $F \in \mathcal{X}$ be the polygon that includes f. Let c' be the point where qc intersects F.
{ Let c'' be a point on qc' infinitesimally close to c'. At this point, by Lemma 5, no mesh vertex }
{ inside S is visible from c''. Therefore, no vertex of f nor g (Line 22) is inside S. }
20 **if** some vertex w of t encroaches upon f
{ Sometimes we discover late that an existing vertex encroaches upon f. Time to split it. }
21 Locate the subpolygon g of F that contains $\text{proj}_F(w)$. SplitSubpolygon(g). **return**.
22 Locate the subpolygon g of F that contains $\text{proj}_F(c)$ by a straight-line walk in F from c'.
23 **if** the walk strikes a subsegment e of F before reaching $\text{proj}_F(c)$
24 SplitSubsegment(e). **return**. { c encroaches upon e. }
{ If c encroaches upon any subpolygon of F, c encroaches upon the one that contains $\text{proj}_F(c)$. }
25 **if** c encroaches upon g
26 SplitSubpolygon(g). **return**.
{ As c does not encroach, some vertex of t must be in the ellipsoid but ineligible. We're stuck. }
27 **else** Mark t as "fenced in" so the algorithm doesn't try to split t again. **return**.

Figure 1: Procedure for splitting a tetrahedron t whose relaxed radius-edge ratio exceeds a threshold B. The subroutines SplitSegment and SplitSubpolygon implement Rules 1 and 2 (pseudocode not included here). The lemmas invoked by the comments appear in the appendix.

very small insertion radii might appear because of encroachments among polygons and segments that meet at small angles. However, our algorithm forces the inequalities to apply to the *relaxed* insertion radii. The following theorem is proven in the appendix.

Theorem 2. *Given an input PLC $\mathcal{X}$, let* $\mathrm{lfs}_{\min} = \min_{p \in |\mathcal{X}|} \mathrm{lfs}(p)$*. Let ψ be the smallest angle at which two adjoining segments in $\mathcal{X}$ meet. Let θ be the smallest dihedral angle at which the affine hulls of two adjoining polygons in $\mathcal{X}$ meet. Let ϕ be the smallest nonzero angle at which a segment meets the affine hull of an adjoining polygon. Suppose a tetrahedron is considered to be splittable if its relaxed radius-edge ratio exceeds a specified bound* $B \geq 2\sqrt{2/3} \doteq 1.63$ *Our Delaunay refinement algorithm terminates with no edge shorter than* $\min\{2, 4\sin(\psi/2), 4\sqrt{2}\sin(\theta/2), 2\sin\phi\} \cdot \mathrm{lfs}_{\min}/5$*. Moreover, no edge is shorter than* $2\,\mathrm{lfs}_{\min}/5$ *except for subsegments and edges whose endpoints are entwined with each other. Every skinny tetrahedron (having a radius-edge ratio greater than B) in the final mesh has at least one vertex that lies on a segment or polygon in $\mathcal{X}$ that fails the projection condition.*

This lower bound on edge lengths compares favorably with the $O(\phi\theta \cdot \mathrm{lfs}_{\min})$ bound of Cheng et al. [5].

The only edges our algorithm creates shorter than $O(\mathrm{lfs}_{\min})$ are subsegments of segments that participate in small angles, and edges that span cells meeting at small angles. Short edges of the latter type cannot be avoided, but we would prefer that all subsegments have length $O(\mathrm{lfs}_{\min})$. Fortunately, we can achieve this goal if we are willing to tolerate a slightly weaker bound B on the radius-edge ratio. The idea is to use off-center splits to align the type 2 vertices on the same spheres as the type 1 vertices so the former cannot encroach on small subsegments. We require not just Ruppert's power-of-two spheres, but also additional spheres between them aligned with the subsegment bisections. There is a trade-off between using coarsely spaced spheres to prevent type 2 vertices from encroaching and using finely spaced spheres to obtain high tetrahedron quality by limiting the circumcenter perturbations. We omit further details.

Our proof shows that our algorithm does not produce unnecessarily short edges relative to a global smallest feature size $\mathrm{lfs}_{\min}$. There are well-known methods for showing that the edge lengths locally adapt proportionally to the local feature size function $\mathrm{lfs}(x)$. Our method can adapt in the same way, though a proof would be quite tedious and we doubt anybody would read it. But we emphasize that our accounting method of recording relaxed insertion radii is particularly effective at limiting the propagation of the tiny edge lengths that necessarily form at tiny domain angles, and could be harnessed to give a user local control over how far small edge lengths propagate and how smoothly they attenuate.

5 A Partial Implementation and Example Meshes

We have a partial implementation of our algorithm in the software TetGen, version 1.5 (November 2013, http://www.tetgen.org). Crucial features we have included are the tracking of relaxed insertion radii and the refusal to split tetrahedra that are fenced in or not splittable. We have not yet implemented diametral ellipsoids; we are using diametral balls for subpolygon encroachment. As a torture test, we created a PLC with 64 irregular "fan blades" adjoining at a common segment separated by very small dihedral angles, ensuring a great deal of mutual encroachment. Figure 2 shows the PLC and the mesh our algorithm generates with a radius-edge ratio bound of $B = 2$. The main observations are that the algorithm successfully produces a mesh, the surviving skinny tetrahedra are all nested within the fan blades (many of them fenced in), there are surprisingly many good tetrahedra between the blades, and the spacing of vertices near the central segment is surprisingly moderate. We also show a mesh of the PLC `m1249` from INRIA's mesh repository, which has many small plane and dihedral angles.

Unfortunately, we were unable to find a conforming Delaunay triangulation code that could triangulate this example for a comparison. However, we are confident that any conforming Delaunay triangulation would necessarily have far more vertices than our constrained Delaunay mesh, because the triangles on the fan blades are squeezed between the neighboring blades.

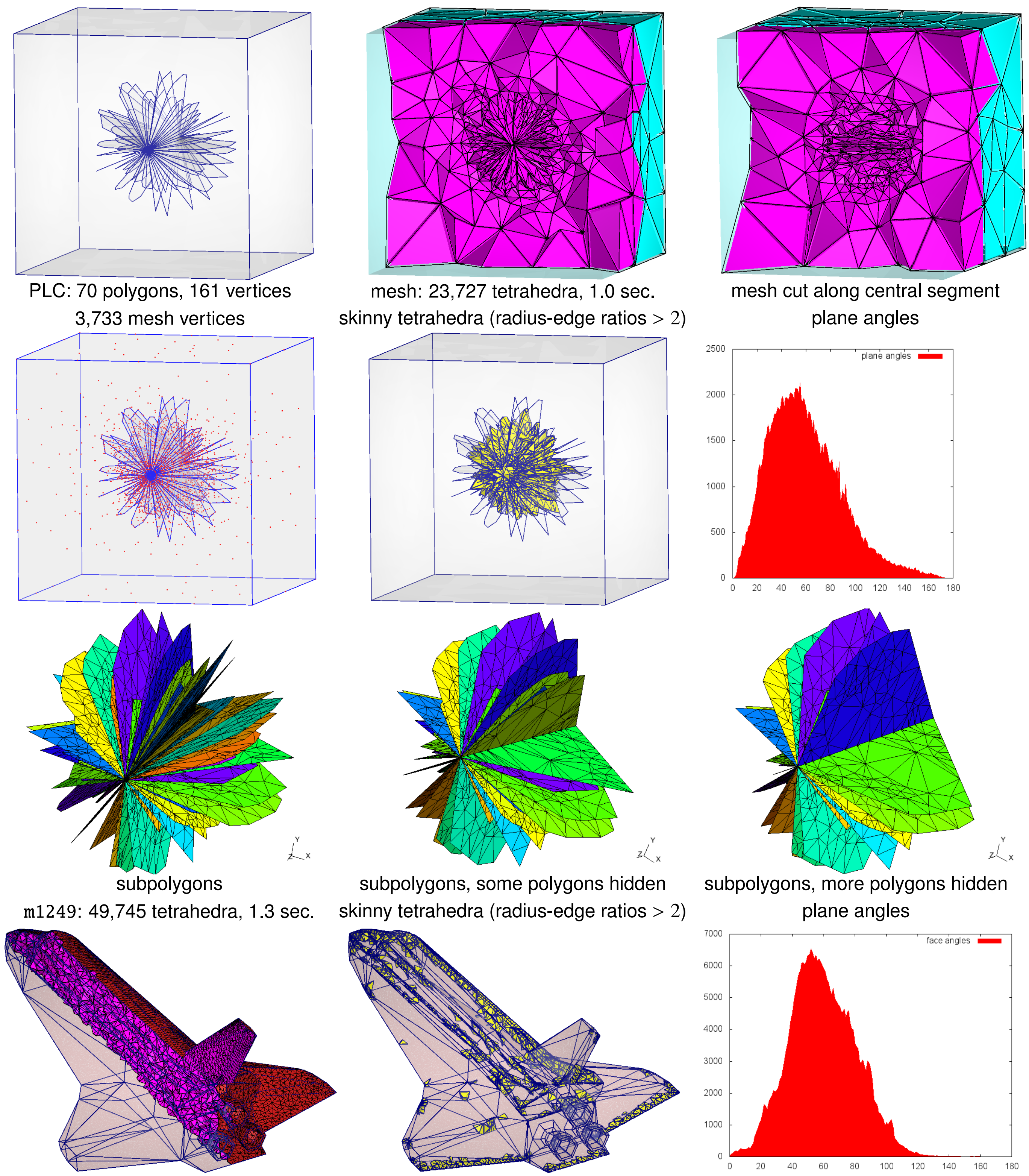


Figure 2: Our torture test (top nine images) and the PLC `m1249` from INRIA's mesh repository (bottom three).

Appendix: Proofs

Note to reviewers: *We append the missing lemmas and proofs here. You don't have to look at them, but we think the statements of Lemmas 5 and 8 are interesting and surprising. We expect that the important parts will fit in the proceedings format. Thank you.*

Let e and e' be two edges in $\mathbb{R}^3$. Say that e *overlaps* e' *from the viewpoint* q if some point of e not shared by e' lies between q and e'. In other words, there exists a point $p_e \in e \backslash e'$ and a point $p_{e'} \in e'$ such that $p_e \in qp_{e'}$. We begin by establishing that when no edge is encroached, this overlap relationship is a partial order with no cycles. Thus, it is not possible for an edge e_1 to overlap e_2, which in turn overlaps e_3, which in turn overlaps e_1, all from the same viewpoint q.

Let B be a ball with center o and radius r. The *power* of B with respect to a viewpoint q is $\Psi_q(B) = |qo|^2 - r^2$. Clearly, $q \in B$ if $\Psi_q(B) \leq 0$, and q is outside B if $\Psi_q(B)$ is positive. Given an edge e, we use $\Psi_q(e)$ as a shorthand for the power $\Psi_q(B)$ of e's diametric ball B. Thus, a point q encroaches upon e if $\Psi_q(e) \leq 0$.

Lemma 3. *Let e and e' be two edges. Suppose that no vertex of e', except perhaps a vertex shared by e, lies in the diametric ball of e, and no vertex of e not shared by e' lies in the diametric ball of e'. If e overlaps e' from the viewpoint q, then $\Psi_q(e) < \Psi_q(e')$.*

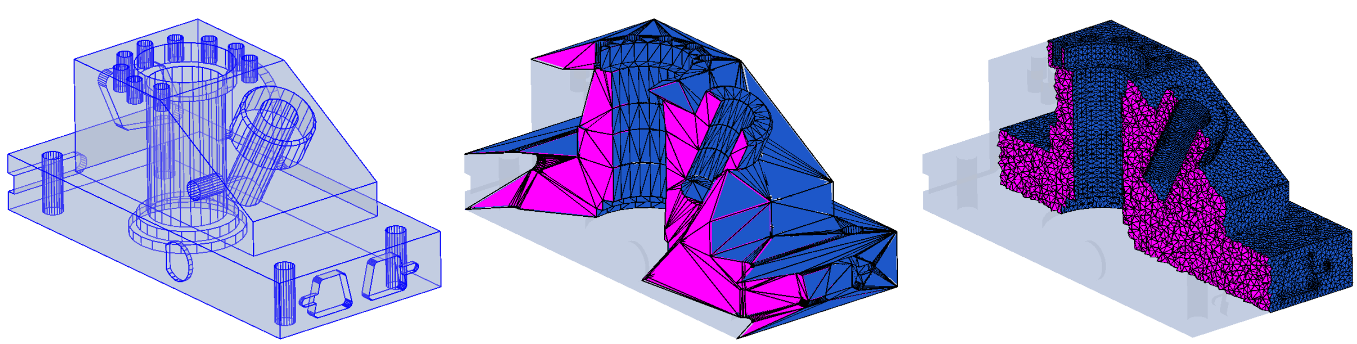
Proof. See Shewchuk [19], Lemma 3. ■

Similar results on the acyclicity of Delaunay triangulations can be found in earlier papers by Edelsbrunner [10] and Edelsbrunner and Shah [11].

Lemma 4. *Let $\mathcal{T}$ be a CDT whose vertices do not encroach upon any of its subsegments. Let c be a point in $\mathbb{R}^3$. If c encroaches upon a subsegment in $\mathcal{T}$ that is partly visible from c, then c encroaches upon a subsegment in $\mathcal{T}$ that is fully visible from c.*

Proof. By assumption, there is a subsegment that is encroached upon by c and at least partly visible from c. Among all such subsegments, let e be the one having the least power with respect to c.

If e is fully visible from c, the result follows. If e is only partly visible from c, pick a point p on e where the visibility changes from visible to occluded. The line segment cp must intersect another subsegment e' that is at least partly visible from c. Because e' overlaps e from c's viewpoint, and neither subsegment is encroached, the power of e' with respect to c is less than the power of e by Lemma 3. As c encroaches upon



`anc101`: 2,772 triangles, 1,378 vertices   Initial CDT   mesh: 258,428 tetrahedra, 6.3 sec.

Figure 3: PLC `anc101` from INRIA's mesh repository. This example demonstrates that our algorithm leaves no skinny tetrahedra behind on a PLC that has no small angles. Our algorithm behaves much like standard Delaunay refinement in the absence of difficult angles.

e, the power of e is zero or negative, thus so is the power of e', and c encroaches upon e'. This contradicts the assumption that e has the least power among all encroached subsegments at least partly visible from c. Therefore, e is fully visible from c. ∎

Lemma 5. *Let $\mathcal{T}$ be a Steiner CDT of a PLC $\mathcal{X}$, and suppose that $\mathcal{T}$'s vertices do not encroach upon any of $\mathcal{T}$'s subsegments. Let $t \in \mathcal{T}$ be a tetrahedron with circumsphere S and circumcenter c. Let q be a point in the interior of t. Let c' be a point on the line segment qc that is visible from q and does not lie on a polygon in $\mathcal{X}$. Then no vertex of $\mathcal{T}$ strictly inside S is visible from any point on qc'.*

Proof. Let H be the convex hull of t, c, and all the vertices of $\mathcal{T}$ strictly inside S. Let $E \subset \mathcal{T}$ be the set of all subsegments with these two properties: for each $e \in E$, at least one vertex of e is strictly inside S, and there is a point $p \in e \cap H$ that is visible from q. We will see that E is empty.

For the sake of contradiction, let e be the subsegment in E that has the least power with respect to q. Let v be a vertex of e that is strictly inside S, and let $p \in e \cap H$ be a point that is visible from q. As t is constrained Delaunay, v is not visible from q. However, both q and v are visible from p. Therefore, $\triangle qpv$ intersects one or more polygons in $\mathcal{X}$; moreover, $\triangle qpv$ intersects some subsegment $e' \in \mathcal{T}$ (on the boundary of one of those polygons) at a point p' that is visible from q. To see this, imagine moving p along e toward v until the instant when q loses sight of p; at that moment, the line segment qp intersects a polygon's boundary at a point visible from q. The subsegment e' found this way overlaps e from q's viewpoint, and therefore e' has lesser power with respect to q than e by Lemma 3. By assumption, e is the subsegment in E with least power, so $e' \notin E$. Observe that H includes $\triangle qpv$, and therefore contains p'. It follows that no vertex of e' is inside S; otherwise, e would be in E.

Let B be the diametric ball of e'. The boundary of B intersects t's circumsphere S in a circle C. The affine hull of C is a plane Π, which divides space into two halfspaces. The vertices of e' lie on the boundary of B but not inside S, so e' is restricted to a closed halfspace we call Π_B. No subsegment is encroached, so no vertex of $\mathcal{T}$ is in B; the vertices of $\mathcal{T}$ that lie on or inside S are restricted to the complementary open halfspace which we call Π_S. Suppose without loss of generality that Π is oriented horizontally with Π_B below and Π_S above. Let m be the midpoint of e', which is also the center of B. The line passing through m and S's center c is perpendicular to Π with m directly below c. Recall that H is a convex hull of vertices on or inside S, which lie below Π in Π_S, and the center c of S. Although c might be above or below Π, it is always below m. If c is below Π, then H is entirely below Π and cannot intersect e'. If c is above Π, then the portion of H above Π is strictly included in the cone with apex c and boundary circle C. As e' has its center m above c and its endpoints on or above C, H still cannot intersect e'. This contradicts the fact that $p' \in H$. From this contradiction we conclude that the overlapping subsegment e' does not exist, the vertex v of e cannot be hidden from q by a polygon, and E is empty.

Let us return to the original claim. Suppose for the sake of contradiction that some vertex $w \in \mathcal{T}$ lies strictly inside S and is visible from a point on qc'. As t is constrained Delaunay, w is not visible from q. As w and q are visible from a point on qc', but not from each other, $\triangle qwc$ intersects some subsegment $e \in \mathcal{T}$ (on the boundary of one of those polygons) at a point p that is visible from q. Observe that $p \in \triangle qwc \subset H$. If no vertex of e is inside S, we repeat the argument of the previous paragraph and obtain a contradiction. If e has a vertex inside S, then $e \in E$; but E is empty. It follows that no vertex inside S is visible from any point on qc'. ∎

Lemma 6. *Let S be a sphere with center c and radius r. Let B be a ball with center m and radius R, and suppose that B is the diametric ball of a subsegment or subpolygon f whose vertices are not inside S. Let $p = \mathrm{proj}_f(c)$ be the point nearest c on f's affine hull, and suppose that $p \in f$.*

If $c \in B$, then $R \geq r/\sqrt{2}$.

If $c \notin B$, then $|cm| \geq |cp| > r/\sqrt{2}$.

Furthermore, if f is a subpolygon, let E be the diametric ellipsoid of f—that is, the diametric ball scaled by a factor of $1/\sqrt{3}$ in the direction orthogonal to f.

If $c \in E$, then $R \geq \sqrt{3}r/2$.

If $c \notin E$, then $|cm| \geq |cp| > r/2$.

Proof. If f is a subpolygon, let Π be the affine hull of f. If f is a subsegment, let Π be the plane that includes f and is perpendicular to cp. In either case, Π contains m and p.

The cross-section $S \cap \Pi$ is a circle C with center p and radius $\bar{r}$, and no vertex of f is inside C. As $p \in f$, f has a vertex v for which $\angle mpv \geq 90°$, unless $m = p$ or $p = v$. In any of these three cases, $R^2 = |mv|^2 \geq |mp|^2 + |pv|^2$. As v is not inside C, $|pv| \geq \bar{r}$, thus $R^2 \geq |mp|^2 + \bar{r}^2$. By Pythagoras' Theorem, $|cm|^2 = |cp|^2 + |mp|^2$ and $r^2 = |cp|^2 + \bar{r}^2 \leq |cp|^2 + R^2 \leq |cm|^2 + R^2$.

If $c \in B$, then $|cm| \leq R$; therefore $r^2 \leq 2R^2$ and the first result follows. If $c \notin B$, then $|cm| > R$; therefore $|cm|^2 > R^2 \geq |mp|^2 + \bar{r}^2 = |cm|^2 - |cp|^2 + r^2 - |cp|^2$. Thus $2|cp|^2 > r^2$ and the second result follows.

If f is a subpolygon and $c \in E$, then $|cp| \leq R/\sqrt{3}$; therefore $r^2 \leq 4R^2/3$ and the third result follows. If $c \notin E$, then $3|cp|^2 + |mp|^2 > R^2 \geq |mp|^2 + \bar{r}^2 = |mp|^2 + r^2 - |cp|^2$. Thus $4|cp|^2 > r^2$ and the fourth result follows. ■

Lemma 7. *Let $\mathcal{T}$ be a CDT whose vertices do not encroach upon any of its subsegments. Let S be a sphere with center c and radius r. Suppose that no vertex inside S is visible from c. Suppose that c encroaches upon no subsegment that is at least partly visible from c. Then the distance from c to every point that lies on a subsegment and is partly or fully visible from c is strictly greater than $r/\sqrt{2}$. Moreover, the distance from c to every point that lies on a subpolygon and is partly* but not fully *visible from c is strictly greater than $r/\sqrt{2}$.*

Proof. Consider a subsegment e whose vertices are not inside S and that is at least partly visible from c. By assumption, e's diametric ball does not contain c. By Lemma 6, the distance from c to any point on e is strictly greater than $r/\sqrt{2}$.

Can a subsegment have a point that is visible from c and closer to c than $r/\sqrt{2}$ if it has a vertex inside S (not visible from c by assumption)? We will see that this is not possible. Suppose for the sake of contradiction that there is a subsegment e with at least one vertex u inside S, and there is a point p on e that is visible from c and no farther than $r/\sqrt{2}$ from c. Moreover, suppose that among all such subsegments, e is the subsegment with least power with respect to c. There is a point q on e between p and u where the visibility from c changes. At this point another subsegment e' overlaps e from c's viewpoint and is visible from c. By Lemma 3, e' has lesser power with respect to c than e, so e' is not such a simplex, so the distance from c to any point on e' is strictly greater than $r/\sqrt{2}$; hence so is the distance from c to q.

Let B be the diametric ball of e'. Let m be the midpoint of e', which is also the center of B. Consider two perpendicular planes that include e': a plane passing through c and a plane P perpendicular to that one. P cuts B into two hemispheres; the hemisphere H farthest from c includes every point that is both inside S and opposite P from c. As e' is not encroached, u cannot be in H. Because e' overlaps e from c's viewpoint, the point q is in H. Therefore, only one of p or u can be on the same side of P as c. It follows that it is not possible to have both u inside S and p no farther than $r/\sqrt{2}$ from c.

This contradiction establishes that the distance from c to any point visible from c on any subsegment is strictly greater than $r/\sqrt{2}$.

The same claim is true for every subpolygon that is partly but not fully visible from c. We establish this by a repetition of the reasoning in the last three paragraphs. ■

Lemma 8. *Given the assumptions of Lemma 7, suppose also that c is in the diametric ellipsoid of no subpolygon f that contains $\mathrm{proj}_f(c)$ and is fully visible from c. Then the distance from c to every segment and every polygon is strictly greater than $r/2$, and the distance from c to the center of the diametric ball of every subsegment and subpolygon is strictly greater than $r/2$.*

Proof. Let p be the point nearest c on all the polygons and segments. Because p is nearest c, p is visible from c. If p lies on a subsegment, then by Lemma 7, the distance from c to every point on every polygon and segment is strictly greater than $r/\sqrt{2}$, as claimed.

Otherwise, p lies on the interior of a polygon F. Note that because p is the point nearest c on F, $p = \mathrm{proj}_F(c)$. Let f be the subpolygon of F that contains p. Suppose for the sake of contradiction that the distance from p to c does not exceed $r/2$. By Lemma 7, partly visible subpolygons cannot be that close to c, so f is fully visible from c. By assumption, f has no vertex inside S and f's diametric ellipsoid does not contain c. By Lemma 6, $|cp| > r/2$, a contradiction. It follows that the distance from c to every point on every polygon and segment is strictly greater than $r/2$.

The center of every subpolygon's diametric ball lies on the subpolygon's polygon, so no subpolygon has a diametric ball whose center is closer to c than $r/2$. ■

Lemma 1. Let v be a vertex (inserted or rejected), and let $p = p(v)$ be its parent.

i. If v is of type 3, then $rr_v > Brr_p \geq Br_p$.
ii. If v is of type 2 and p is of type 3, then $rr_v \geq \sqrt{3}rr_p/2$.
iii. If v is a type 1 midpoint (not inserted off-center) and p is of type 3, then $r_v \geq rr_p/\sqrt{2}$.
iv. If v is a type 1 midpoint of a segment s and p is of type 2 on a polygon $F \supset s$, then $r_v \geq rr_p/\sqrt{2}$.

Proof. i. By definition, a splittable tetrahedron has a circumradius rr_v greater than B times rr_p, where p is the tetrahedron vertex that minimizes rr_p.

ii. When SplitTetrahedron considers inserting a vertex at the circumcenter p of a tetrahedron t with circumradius rr_p, it consents to split a subpolygon g with circumcenter v and circumradius rr_v in only two circumstances: when the vertices of g are not in t's circumsphere, in which case $rr_v \geq \sqrt{3}rr_p/2$ by Lemma 6, or when the distance from v to the center of the diametric ball of every subpolygon is known to be greater than $rr_p/2$ by Lemma 7, in which case every encroached subpolygon satisfies the inequality. See the comments in SplitTetrahedron for further details.

iii. As above, but a subsegment's diametric ball is known to be empty when a type 3 vertex encroaches upon it, so we can bound r_v as well as rr_v.

iv. Follows immediately from Lemma 6. ■

Theorem 2. Given an input PLC $\mathcal{X}$, let $\mathrm{lfs}_{\min} = \min_{p \in |\mathcal{X}|} \mathrm{lfs}(p)$. Let ψ be the smallest angle at which two adjoining segments in $\mathcal{X}$ meet. Let θ be the smallest dihedral angle at which the affine hulls of two adjoining polygons in $\mathcal{X}$ meet. Let ϕ be the smallest nonzero angle at which a segment meets the affine hull of an adjoining polygon. Suppose a tetrahedron is considered to be splittable if its relaxed radius-edge ratio exceeds a specified bound $B \geq 2\sqrt{2/3} \doteq 1.63$ Our Delaunay refinement algorithm terminates with no edge shorter than $\min\{2, 4\sin(\psi/2), 4\sqrt{2}\sin(\theta/2), 2\sin\phi\} \cdot \mathrm{lfs}_{\min}/5$. Moreover, no edge is shorter than $2\,\mathrm{lfs}_{\min}/5$ except for subsegments and edges whose endpoints are entwined with each other. Every skinny tetrahedron (having a radius-edge ratio greater than B) in the final mesh has at least one vertex that lies on a segment or polygon in $\mathcal{X}$ that fails the projection condition.

Proof. For every type 0 vertex $w \in \mathcal{X}$, r_w is the distance to the nearest visible vertex in $\mathcal{X}$, so $rr_w = r_w \geq \mathrm{lfs}(w) \geq \mathrm{lfs}_{\min}$. Let v be a vertex that is subsequently inserted into the mesh or rejected. We show by induction on the temporal sequence of vertices that $rr_v \geq 2\,\mathrm{lfs}_{\min}/5$, and moreover $rr_v \geq 2\sqrt{2}\,\mathrm{lfs}_{\min}/5$ if

v is of type 2, and $rr_v > 4\sqrt{2/3}\,\mathrm{lfs}_{\min}/5$ if v is of type 3. Suppose for the inductive hypothesis that these statements hold for every vertex that was inserted into the mesh or rejected before v is inserted. Let $p = p(v)$ be the parent of v. Consider the following cases.

- If v is a type 3 circumcenter of a splittable tetrahedron, then $rr_v > Brr_p$ by Lemma 1, so $rr_v > 2\sqrt{2/3}rr_p \geq 4\sqrt{2/3}\,\mathrm{lfs}_{\min}/5$ by the inductive hypothesis.

- If v is a type 2 circumcenter of an encroached subpolygon, consider the following cases.
  - If p is of type 3, then $rr_v \geq rr_p/\sqrt{2}$ by Lemma 1. By the inductive hypothesis, $rr_p > 4\sqrt{2/3}\,\mathrm{lfs}_{\min}/5$. Therefore, $rr_v > 2\sqrt{2}\,\mathrm{lfs}_{\min}/5$.
  - If p is of type 2 and entwined with v, then $|vp| \geq rr_p$; otherwise, p would not be eligible to encroach. Therefore, $rr_v = |vp| \geq 2\sqrt{2}\,\mathrm{lfs}_{\min}/5$.
  - If p is of type 1 and entwined with v, then $|vp| \geq \sqrt{2}rr_p$; otherwise, p would not be eligible to encroach. Therefore, $rr_v = |vp| \geq 2\sqrt{2}\,\mathrm{lfs}_{\min}/5$.
  - If p is of type 0–2 and the two cases above do not apply, then v and p lie on disjoint members of $\mathcal{X}$. Therefore, $rr_v = |vp| \geq \mathrm{lfs}(v) \geq \mathrm{lfs}_{\min}$.

- If v is a type 1 vertex inserted off-center on an encroached subsegment with endpoints a and b, $|va| < |vb|$, then $|va|$ is at least one-fifth the length of the original segment. At the midpoint m of the original segment, $\mathrm{lfs}(m)$ is half the length of the original segment, so $rr_v \geq |va| \geq 2\,\mathrm{lfs}(m)/5 \geq 2\,\mathrm{lfs}_{\min}/5$.

- If v is a type 1 midpoint of an encroached subsegment with endpoints a and b, consider the following cases.
  - If p is of type 3, then $r_v \geq rr_p/\sqrt{2}$ by Lemma 1. By the inductive hypothesis, $rr_p > 4\sqrt{2/3}\,\mathrm{lfs}_{\min}/5$. Therefore, $rr_v > 4\,\mathrm{lfs}_{\min}/(5\sqrt{3})$.
  - If p is a type 2 circumcenter of a subpolygon f on a polygon $F \supset e$, then $r_v \geq rr_p/\sqrt{2}$ by Lemma 1. By the inductive hypothesis, $rr_p \geq 2\sqrt{2}\,\mathrm{lfs}_{\min}/5$, so $rr_v \geq r_v \geq 2\,\mathrm{lfs}_{\min}/5$.
  - If p is of type 2 and entwined with v, then $rr_v \geq \min\{rr_p/\sqrt{2}, \min_w d_{vw}\}$ by construction, where w ranges over vertices connected to v by CDT edges. If $rr_v = rr_p/\sqrt{2}$, then by the inductive hypothesis, $rr_v \geq 2\,\mathrm{lfs}_{\min}/5$. Otherwise, let u be the vertex minimizing $\min_w d_{vw}$. Either $d_{vu} = rr_u$ or $d_{vu} = |vu|$ and v is not entwined with u. In the former case, $rr_v = rr_u \geq 2\,\mathrm{lfs}_{\min}/5$. In the latter case, if u is of type 0–2, then v and u lie on disjoint members of $\mathcal{X}$, so $rr_v = |vu| \geq \mathrm{lfs}(v) \geq \mathrm{lfs}_{\min}$. If u is of type 3, then $|vu| > rr_p/\sqrt{2}$ by Lemma 7, because SplitTetrahedron would not have inserted u unless the lemma's preconditions held. Therefore, $rr_v = |vu| > 4\,\mathrm{lfs}_{\min}/(5\sqrt{3})$.
  - If p is a, b, or a type 1 vertex entwined with v, then $rr_v = d_{vp} \geq rr_p \geq 2\,\mathrm{lfs}_{\min}/5$.
  - If p is of type 0–2 and the three cases above do not apply, then v and p lie on disjoint members of $\mathcal{X}$. Therefore, $|vp| \geq \mathrm{lfs}(v) \geq \mathrm{lfs}_{\min}$. By construction, $rr_v = |va| \geq |vp| \geq \mathrm{lfs}_{\min}$ if p is of type 2 (thus rejected), and $rr_v = d_{vp} \geq |vp| \geq \mathrm{lfs}_{\min}$ otherwise.

For tetrahedron circumcenters that are not rejected, $r_v = rr_v$. Therefore, a newly inserted type 3 vertex is no closer than $4\sqrt{2/3}\,\mathrm{lfs}_{\min}/5$ to any prior visible vertex, so the algorithm can insert only a finite number of type 3 vertices. It eventually runs out of places to insert new ones.

Although a subpolygon circumcenter can have $r_v \ll rr_v$ because of entwinement, a newly inserted type 2 vertex is no closer than $2\sqrt{2}\,\mathrm{lfs}_{\min}/5$ to any prior vertex *on the same polygon*. Hence, the algorithm can insert only a finite number of type 2 vertices. A type 2 vertex is not inserted if it encroaches upon a subsegment, so no type 2 vertex is closer than $2\,\mathrm{lfs}_{\min}/5$ to its polygon's boundary. Thus the distance between two type 2 vertices lying on different polygons is at least $\mathrm{lfs}_{\min}$ if the polygons are disjoint, and at least $(4/5)\,\mathrm{lfs}_{\min}\sin(\theta/2)$ if their affine hulls meet at a dihedral angle of θ. The distance between a type 2 vertex and a type 1 vertex not on the same polygon is at least $\mathrm{lfs}_{\min}$ if the polygon and segment are disjoint, or at least $(2/5)\,\mathrm{lfs}_{\min}\sin\phi$ if the segment meets the affine hulls of the polygon at an angle of ϕ.

Segment splitting with concentric shells has the effect that if the vertices of a subsegment e lie on two concentric shells (or one on a shell and one at the center of the shells), e can only be encroached upon by vertices between those two shells. It follows that no vertex is ever inserted closer to a type 0 vertex than a distance of $2\,\mathrm{lfs}_{\min}/5$. (There is a shell centered at the vertex with a radius between $2\,\mathrm{lfs}_{\min}/5$ and $4\,\mathrm{lfs}_{\min}/5$ in which no vertex can be placed.) The distance between two type 1 vertices lying on different segments is at least $\mathrm{lfs}_{\min}$ if the segments are disjoint, or at least $(4/5)\,\mathrm{lfs}_{\min}\sin(\psi/2)$ if they meet at an angle of ψ.

There is only one circumstance in which a subsegment shorter than $2\,\mathrm{lfs}_{\min}/5$ can be created if a subsegment that short did not already exist. We have seen that a type 2 vertex and a segment can be as close as $(2/5)\,\mathrm{lfs}_{\min}\sin\phi$, but no closer, when a polygon meets a segment at a small angle. If the vertex encroaches upon a subsegment of the segment, the new subsegments thus created can be equally short, but no shorter. Hence, the algorithm can insert only a finite number of type 1 vertices. Therefore, the algorithm terminates.

Consider a skinny tetrahedron t that survives in the final mesh; either t is not splittable or it is fenced in. If t is not splittable, the length ℓ of its shortest edge e is less than the relaxed insertion radius rr_v of e's most recently inserted vertex v. A CDT always connects a vertex to its nearest visible neighbor, which implies that when v was inserted, it was assigned a relaxed insertion radius rr_v greater than the distance to that neighbor. This is possible only if v was entwined with its nearest visible neighbor when it was inserted, or if v is type 1 and was entwined with its rejected type 2 parent. In either case, v lies on a cell in $\mathcal{X}$ that fails the projection condition.

If t is fenced in, t encroaches upon a subpolygon f that hides the visibility of t's circumcenter c from a point in t's interior, yet c is not in f's diametric ball. This implies that at least one vertex v of t that is not on F is in f's diametric ball. Yet v is not eligible to encroach upon F, so v is entwined with F. Hence v lies on a cell in $\mathcal{X}$ that, jointly with F, fails the projection condition. ∎

References

[1] Charles Boivin and Carl Ollivier-Gooch. *Guaranteed-Quality Triangular Mesh Generation for Domains with Curved Boundaries*. International Journal for Numerical Methods in Engineering **55**(10):1185–1213, 20 August 2002.

[2] Siu-Wing Cheng and Tamal Krishna Dey. *Quality Meshing with Weighted Delaunay Refinement*. Proceedings of the Thirteenth Annual Symposium on Discrete Algorithms (San Francisco, California), pages 137–146. Association for Computing Machinery, January 2002.

[3] Siu-Wing Cheng, Tamal Krishna Dey, and Edgar A. Ramos. *Delaunay Refinement for Piecewise Smooth Complexes*. Discrete & Computational Geometry **43**(1):121–166, 2010.

[4] Siu-Wing Cheng, Tamal Krishna Dey, Edgar A. Ramos, and Tathagata Ray. *Quality Meshing for Polyhedra with Small Angles*. Proceedings of the Twentieth Annual Symposium on Computational Geometry (Brooklyn, New York), pages 290–299. Association for Computing Machinery, June 2004.

[5] Siu-Wing Cheng, Tamal Krishna Dey, and Jonathan Richard Shewchuk. *Delaunay Mesh Generation*. Taylor & Francis, 2012.

[6] Siu-Wing Cheng and Sheung-Hung Poon. *Graded Conforming Delaunay Tetrahedralization with Bounded Radius-Edge Ratio*. Proceedings of the Fourteenth Annual Symposium on Discrete Algorithms (Baltimore, Maryland), pages 295–304. Society for Industrial and Applied Mathematics, January 2003.

[7] L. Paul Chew. *Guaranteed-Quality Triangular Meshes*. Technical Report TR-89-983, Department of Computer Science, Cornell University, 1989.

[8] ———. *Guaranteed-Quality Mesh Generation for Curved Surfaces*. Proceedings of the Ninth Annual Symposium on Computational Geometry (San Diego, California), pages 274–280. Association for Computing Machinery, May 1993.

[9] David Cohen-Steiner, Éric Colin de Verdière, and Mariette Yvinec. *Conforming Delaunay Triangulations in 3D*. Proceedings of the Eighteenth Annual Symposium on Computational Geometry (Barcelona, Spain), pages 199–208. Association for Computing Machinery, June 2002.

[10] Herbert Edelsbrunner. *An Acyclicity Theorem for Cell Complexes in d Dimension*. Combinatorica **10**(3):251–260, September 1990.

[11] Herbert Edelsbrunner and Nimish R. Shah. *Incremental Topological Flipping Works for Regular Triangulations*. Algorithmica **15**(3):223–241, March 1996.

[12] Paul-Louis George and Houman Borouchaki. *Delaunay Triangulation and Meshing: Application to Finite Elements*. Hermès, Paris, 1998.

[13] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, Noel J. Walkington, and Han Wang. *Control Volume Meshes Using Sphere Packing: Generation, Refinement and Coarsening*. Proceedings of the 5th International Meshing Roundtable (Pittsburgh, Pennsylvania), pages 47–61, October 1996.

[14] Michael Murphy, David M. Mount, and Carl W. Gable. *A Point-Placement Strategy for Conforming Delaunay Tetrahedralization*. Proceedings of the Eleventh Annual Symposium on Discrete Algorithms, pages 67–74. Association for Computing Machinery, January 2000.

[15] Steven E. Pav and Noel J. Walkington. *Robust Three Dimensional Delaunay Refinement*. Proceedings of the 13th International Meshing Roundtable (Williamsburg, Virginia), pages 145–156. Sandia National Laboratories, September 2004.

[16] Alexander Rand and Noel J. Walkington. *Collars and Intestines: Practical Conformal Delaunay Refinement*. Proceedings of the 18th International Meshing Roundtable (Salt Lake City, Utah), pages 481–497. Springer, October 2009.

[17] Jim Ruppert. *A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation*. Journal of Algorithms **18**(3):548–585, May 1995.

[18] E. Schönhardt. *Über die Zerlegung von Dreieckspolyedern in Tetraeder*. Mathematische Annalen **98**:309–312, 1928.

[19] Jonathan Richard Shewchuk. *A Condition Guaranteeing the Existence of Higher-Dimensional Constrained Delaunay Triangulations*. Proceedings of the Fourteenth Annual Symposium on Computational Geometry (Minneapolis, Minnesota), pages 76–85. Association for Computing Machinery, June 1998.

[20] ———. *Tetrahedral Mesh Generation by Delaunay Refinement*. Proceedings of the Fourteenth Annual Symposium on Computational Geometry (Minneapolis, Minnesota), pages 86–95. Association for Computing Machinery, June 1998.

[21] ———. *Mesh Generation for Domains with Small Angles*. Proceedings of the Sixteenth Annual Symposium on Computational Geometry (Hong Kong), pages 1–10. Association for Computing Machinery, June 2000.

[22] ———. *Delaunay Refinement Algorithms for Triangular Mesh Generation*. Computational Geometry: Theory and Applications **22**(1–3):21–74, May 2002.

[23] ———. *General-Dimensional Constrained Delaunay Triangulations and Constrained Regular Triangulations I: Combinatorial Properties*. Discrete & Computational Geometry **39**(1–3):580–637, March 2008.

[24] Hang Si. *Constrained Delaunay Tetrahedral Mesh Generation and Refinement*. Finite Elements in Analysis and Design **46**:33–46, January–February 2010.

[25] Hang Si and Klaus Gärtner. *Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations*. Proceedings of the Fourteenth International Meshing Roundtable (San Diego, California) (Byron W. Hanks, editor), pages 147–163, September 2005.