An Incremental Delaunay Meshing Algorithm*

Gary L. Miller† Steven E. Pav‡ Noel J. Walkington§

June 9, 2002

Abstract

The classical meshing problem is to construct a triangulation of a region that conforms to the boundary, is as coarse as possible, and is constructed from simplices having bounded aspect ratio. In this paper we present an implementation of a class of algorithms introduced by Ruppert and establish their correctness. This class of algorithms solves the meshing problem in two dimensions, and partially solve it in three dimensions. Since geometric degeneracies frequrently cause such algorithms to fail, care is taken to accomodate these in the proofs.

Keywords. Delaunay Triangulation, Mesh Generation, Ruppert's Algorithm

AMS subject classifications 65N50, 68U05

1 Introduction

In 1992 Ruppert [16, 17], building upon an idea of Chew [3], introduced a two dimensional meshing procedure that is conceptually very elementary, produces meshes within a constant of optimal output size, and works remarkably well in practice. The algorithm, as implemented in Triangle [18], works remarkably well in practice with thousands of users. Moreover, the ideas introduced by Ruppert to establish these results are very elegant and have been useful in a much broader context [10, 11, 12].

*Submitted SICOMP June 2000

†Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213. Supported in part by National Science Foundation Grants CCR–9902091, CCR-9706572 and ACI 0086093.

‡Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213

§Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213. Supported in part by National Science Foundation Grants DMS–9973285, CCR–9902091 and ACI-0086093. This work was also supported by the NSF through the Center for Nonlinear Analysis.

Shewchuk [20] proposed a three dimensional version of Ruppert's Delaunay refinement algorithm. At the time of this writing we do not know of any publicly available implementation of this or any other three dimensional Delaunay refinement algorithm. We have implemented a fully incremental refinement algorithm with fairly good success. One of the goals of this paper is to give a complete enough exposition so that one may relatively easily implement a three dimensional Delaunay refinement meshing algorithm.

The running time of Delaunay refinement algorithms is still open, and this is not addressed here. The running time of two dimensional algorithms is still open. Examples suggest that, in the worst case where there are both very large and very small features, that the running time is $\Omega(M^2)$ where M is the size of the output mesh. The algorithm we present below does not have blatantly quadratic run time. Many of the algorithms proposed in the literature do not make explicit how some of the tests will be performed. One could assume that they had in mind an auxiliary data structure such as a quadtree; the details are usually lacking. We are able to construct our mesh incrementally. That is, we begin by picking a small number of points, say four in 3D, and forming the mesh on these four vertices. At each stage we add either a new input or Steiner point to the mesh induced on these points. Our algorithm is based upon a well known incremental Delaunay triangulation algorithm. In order to coerce the triangulation algorithm to solve the meshing problem we modify the code to satisfy a small set of "invariants" or "induction hypotheses". These induction hypotheses form the cornerstone of both the design and analysis of our algorithm. The hypotheses were constructed so that their satisfaction would guarantee that the triangulation conformed to the input data (edges, faces etc.), and in some sense our algorithm is a minimal perturbation of the incremental Delaunay triangulation code required to maintain them.

One of the subtle points about 3D meshing over 2D, at least for refinement based algorithms, is that there is a partial order of meshes which are all being meshed at the same time, and these meshes need not agree until the algorithm terminates. In particular, we have a separate mesh for each input face. Understanding and controlling the interaction between this partial order of meshes is critical for analysis and correctness of any algorithm. By making the algorithm fully incremental this interaction becomes explicit and relatively easy to understand.

While degeneracy is frequently dismissed as "an event having probability zero," this is not the case in practice since mechanical components typically contain rectangular faces and circular holes. Also, algorithms in computational geometry involve many "real to Boolean" operations so are inherently ill posed. Even if exact arithmetic is assumed, geometric degeneracies can cause infinite recursions and other modes of failure. Our analysis assumes all quantities are computed exactly. Shewchuk [19] developed precise real to Boolean predicates, and empirical evidence shows that this approach has minimal impact upon performance. In this situation algorithms must still accommodate the geometric degeneracies. For example, if four points are cocircular, then the Delaunay triangulation is not unique. In two dimensions only one mesh is being constructed, so any runtime breaking of symmetry works. As stated above, in three dimensions there is a "partial order" of independently

maintained two and three dimensional meshes and at termination they must agree. Our algorithm was carefully constructed to accommodate this requirement and effectively breaks symmetries in each mesh in a consistent fashion.

Delaunay refinement algorithms can control the radius edge aspect ratio of tetrahedra and it is known that this will not eliminate one class of degenerate tetrahedra, "slivers" and the meshes they produce are only *pointwise omptimal*[1]. Recent results of Edelsbrunner et. al. [8, 9] extend ideas of Chew [4] to prove that a specific "smoothing" algorithm can eliminate slivers from a mesh having bounded radius edge ratio. Then all tetrahedra will have a bounded classical aspect ratio (e.g. ratio of diameter to radius of largest inscribed sphere). In this situation Ruppert's two dimensional proof can be repeated to show that the output size of a mesh produced by the composition of Ruppert's procedure with the smoothing algorithm is within a constant of optimal.

Prior to Ruppert's paper the only two-dimensional meshing algorithm having a sound theoretical footing was the quadtree algorithm introduced by Bern, Epstein, and Gilbert [1]. Meshes produced by the quadtree algorithm have "Cartesian character" and are not invariant under rotation of the input data; moreover, empirical evidence suggests that while the output size of this algorithm is within a constant of optimal, the constant is significantly larger than that of Ruppert's procedure. The quadtree algorithm was extended to three dimensions by Mitchell and Vavasis [13]. Their octree algorithm is non-trivial; however, they have implemented a very general version of it [13, 14] and it is the only provably correct three dimensional meshing algorithm that we are aware of that has actually been implemented.

As in Ruppert's original paper, our algorithm and proofs are only applicable to a limited class of input. While this restriction is easy to circumvent in two dimensions, [16, 21], currently modifications of the three dimensional algorithm to accommodate arbitrary input data are not known. Partial results in this direction are available [5].

In the next section we introduce some terminology and recall Ruppert's original procedure for constructing a two dimensional mesh. We then recall a well-known incremental Delaunay triangulation algorithm which is the kernel underlying our implementations of our incremental algorithm. In Section 3 we present a two dimensional implementation of Ruppert's procedure and establish some properties of the algorithm. Section 4 extends the ideas in Section 3 to solve the three dimensional meshing problem. This algorithm retains much of the simplicity of the two dimensional algorithm; and is an extension in the sense that the proof of correctness given in Section 5 contains the two dimensional proof.

[1]The radius edge ratio and pointwise optimality are defined in Sections 5.1 and 5.1 respectively

2 Background

Notation

In order to clearly define the input we recall the following definitions from Miller, et al. [12]

Definition 2.1 (Polytope and PLS) *A* polytope *is the convex combination of finite set of points $P \subset \mathbb{R}^d$. The dimension of the polytope is the dimension of the affine subspace generated by P.*

A piecewise-linear system*, or* PLS*, is a set of polytopes $\mathcal{W}$ with the following properties:*

- *The set $\mathcal{W}$ is closed under taking boundaries,* i.e.*, for each polytope P in $\mathcal{W}$, the boundary of P is a union of polytopes in $\mathcal{W}$.*
- *$\mathcal{W}$ is closed under intersection.*
- *For polytopes P, Q in $\mathcal{W}$, if $\dim(P \cap Q) = \dim(P)$ then $P \subseteq Q$ and $\dim(P) < \dim(Q)$.*

The dimension of a PLS is the dimension of the highest-dimensional polytope in that PLS, and we think of the polytopes of a d-dimensional PLS as being embedded in $\mathbb{R}^d$.

Two-dimensional PLS's can be represented as a set of points $\mathcal{P} \subseteq \mathbb{R}^2$, and a set of segments $\mathcal{S} \subset \mathcal{P} \times \mathcal{P}$, with the intersection properties outlined above.

We adopt Ruppert's terminology which distinguishes the set of input edges $\mathcal{S}_0$ in the PLS from other edges by referring to them as *input edges*, and we refer to (them or) their subdivisions as *segments*. In three dimensions the input will also consist of a set of planar two dimensional polytopes denoted by $\mathcal{F}_0$ which we refer to as the *input faces*. These faces will be triangulated and we will refer to the triangles in these triangulations as *facets* and denote their union as $\mathcal{F}$. The "empty sphere" property of Delaunay triangulation plays an important role of our algorithm and the following definitions are useful in this context.

Definition 2.2 *Let $K \subset \mathbb{R}^d$ be a simplex.*

- *The* diametral ball *of K, denoted $B(K)$, is the open ball of minimal diameter containing the vertices of K. If $dim(K) = d$ then $B(K)$ is the circumball of K.*
- *Let $p \in \mathbb{R}^d$, then p* encroaches *upon K if $p \in B(K)$. If K is a simplex in a triangulation, we say it is encroached if any vertex of the triangulation is inside its diametral ball.*

  The boundary sphere if $B(K)$ is denoted by $S(K)$.
- *The radius edge ratio of K is the ratio of the radius of $B(K)$ to the length of the shortest edge of K.*

Ruppert's Procedure

We briefly recall Ruppert's procedure for meshing a two dimensional PLS.

1. Encroached segments are eliminated by splitting them. Specifically, if $s = (p, r)$ is encroached, let q be the mid point of s and add it to the point set, $\mathcal{P} \leftarrow \mathcal{P} \cup \{q\}$, and replace s in the segment set by it's two halves, $\mathcal{S} \leftarrow (\mathcal{S} \setminus \{s\}) \cup \{(p, q), (q, r)\}$. This process is repeated until all encroachments are eliminated.

2. The Delaunay triangulation $\mathcal{T}$ of the point set $\mathcal{P}$ is computed.

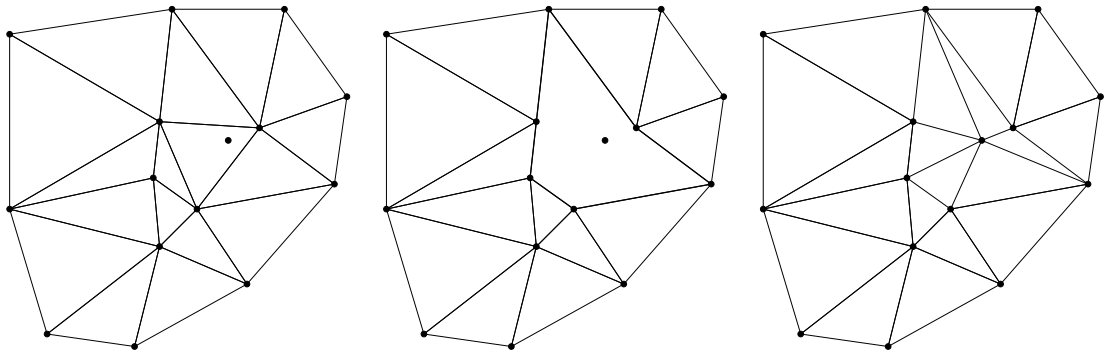
3. Skinny triangles are eliminated as follows: Set a threshold $\theta_0 < \sin^{-1}(1/2\sqrt{2})$, then if a triangle $t \in \mathcal{T}$ has an angle less than θ_0 let p be the circumcenter. If p encroaches upon a segment $s \in \mathcal{S}$, split s, and repeat steps (1) and (2); otherwise, if p doesn't encroach upon any segment update the Delaunay triangulation to include it.

It is clear that a naive implementation of this procedure would involve a lot of searching that could easily lead to run times quadratic in the output size. The algorithm we propose combines the first step to relieve encroachments with the construction of the initial Delaunay triangulation and is a modification of a well known incremental Delaunay triangulation algorithm which we describe next.

Incremental Delaunay Algorithm

The incremental algorithm for computing the Delaunay triangulation of a set of points is initialized by determining a large bounding simplex (or box) which contains all of the points $\mathcal{P}$ and initializing $\mathcal{T}$ to be this simplex. Next, the points in $\mathcal{P}$ are added one at a time to $\mathcal{T}$ as follows. Given $p \in \mathcal{P}$, remove all of the d-simplices in $\mathcal{T}$ that contain p in their circumballs $B(t)$. This leaves a "cavity" (the *Delaunay Cavity*) in the triangulation bounded by a set of $d-1$ simplices $\mathcal{K} = \{k_1, k_2, \ldots, k_n\}$ (see Figure 1). The cavity is filled by adding simplices to $\mathcal{T}$ of the form $K = (p, k_i)$ having apex p and opposite simplex $k_i \in \mathcal{K}$.

The algorithm for determining simplices in $\mathcal{T}$ containing p in their circumsphere needs to be specified. This is a crucial step in the analysis of the algorithm. One method is to assign every point $p \in \mathcal{P}$ not yet in the triangulation to a simplex K for which $p \in B(K)$. Given a point-simplex pair, (p, K), all all simplices K' for which $p \in B(K')$ can be determined by a local search since their union forms a connected patch (the cavity). As the search proceeds and simplices K' in the cavity are removed, the points $\{p'\}$ that were assigned to them are gathered, and reassigned to one of the new simplices having p as an apex, see [6]. An alternative method is to maintain a data structure for point location which essentially stores the decision tree used to assign the points to simplices in the "gather and reassign" step just described. This procedure has the advantage that the points new points (such a segment split points) can be introduced at any stage.



(a) point to be added. (b) cavity removed. (c) tent panels added.

Figure 1: The incremental Delaunay construction.

In two dimensions the expected run time of the incremental algorithm on N randomly ordered points is $\mathcal{O}(N \ln(N))$. In three dimensions the situation is more complicated since N points may result in $\mathcal{O}(N^2)$ tetrahedra. Output sensitive algorithms optimal up to logarithmic factors are known for three dimensional Delaunay triangulation [2].

Degeneracy

Given a collection of points $\mathcal{P}$, the classical Delaunay "empty ball" criteria sates that if an empty ball B contains $\ell + 1$ points on its boundary then the ℓ-simplex formed from their convex hull is present in the Delaunay triangulation of $\mathcal{P}$. This statement holds in the absence of degeneracy; that is, when the convex hull of any $\ell + 1$ points lying on the boundary of an empty ball has dimension ℓ. When the dimension is less than ℓ the empty ball criteria does not determine a triangulation of the convex hull of the input points, instead it defines a decomposition into a unique collection of *Delaunay polytopes*.

Example: If the empty ball criteria is used to "triangulate" the eight vertices of a cube, the decomposition into Delaunay polytopes would consist of the cube, its six square faces and the edges of the cube.

In this situation Delaunay *triangulation* algorithms, represent the Delaunay polytopes as a union of simplices; clearly these triangulations are not unique. The triangulation produced by the incremental algorithm will depend upon the order in which the points are processed. We have defined the circumballs of a simplex to be open; however, the incremental algorithm will also produce a consistent, but different, Delaunay triangulation if the balls are chosen to be closed. The two dimensional *meshing* algorithm introduced below will work if diametral balls are defined to be open or closed; however, in three dimensions degeneracies could result in an infinite recursion if the diametral balls are defined to be closed. By

defining circumballs to be open and assigning an order in which points are inserted into the triangulations we avoid the problem with degeneracies encountered by Shewchuk [20]. In [20] distinct Delaunay triangulations of faces input to the three dimensional meshing algorithm had to be adjusted to agree.

3 An Incremental 2d Meshing Algorithm

3.1 Overview

Our incremental algorithm is a modification of the incremental Delaunay triangulation algorithm described above. Central to the implementation and analysis are the following invariants that we maintain throughout.

Induction Hypotheses: At each stage a set of points $\mathcal{P}$, segments $\mathcal{S}$, and a Delaunay triangulation $\mathcal{T}$ of a subset of $\mathcal{P}$ is maintained. These three sets satisfy

1. If the two end points of a segment $s \in \mathcal{S}$ are in $\mathcal{T}$ then s is an edge in $\mathcal{T}$
2. Segments present as edges in $\mathcal{T}$ are not encroached by vertices in $\mathcal{T}$.

We implicitly assume that the union of the sets of points, $\mathcal{P}$, and segments, $\mathcal{S}$, form a PLS in the sense outlined in Definition 2.1. Notice that in the absence of degeneracy hypothesis (2) implies hypothesis (1).

In order to interpret the induction hypotheses it is important to correctly interpret what is and what is not a segment. Input to the algorithm is a set of points $\mathcal{P}_0$ and a set of pairs of points $\mathcal{S}_0$ which correspond to input line segments. During the execution of our algorithm we maintain a set of points, $\mathcal{P}$, and segments, $\mathcal{S}$. These sets are initialized as $\mathcal{P} = \mathcal{P}_0$ and $\mathcal{S} = \mathcal{S}_0$, and subsequently an edge (p, r) in the triangulation is called a segment if and only if $(p, r) \in \mathcal{S}$. Frequently such a segment (p, r) will be split. This corresponds to removing (p, r) from $\mathcal{S}$ and adding in the two halves $\mathcal{S} \leftarrow (\mathcal{S} \setminus \{(p, r)\}) \cup \{(p, q), (q, r)\}$ where q is the mid point of (p, r). Notice that after this operation (p, r), while still an edge in the triangulation, is no longer considered to be a segment (it is not in $\mathcal{S}$).

The function `add-point2d()` described in Figure 2 is a variant of the incremental Delaunay procedure designed to maintain the induction hypotheses. This function is the heart of our incremental algorithm and is lazy in the sense that it never splits segments until an encroachment is detected during the construction of a Delaunay cavity. Since the procedure for finding a triangle encroached by the point in Step 4 is identical to that used for the incremental Delaunay algorithm the details for doing this have been omitted. The parameter ρ_2 occurring in Step 6 is threshold edge-radius ratio; triangles having a smaller ratio are declared to be skinny.

The initial call to `add-point2d()` with the `Force` option will implement the first two steps of Ruppert's procedure. The final stage, where skinny triangles are eliminated, is accomplished by a function `rm-tri()` which simply makes judicious calls to `add-point2d()`

1. Procedure `add-point2d()`

2. INPUT: The current triangulation, a point p, a set $\mathcal{P}$ of points queued for `Force`'d addition to the triangulation, a set of segments $\mathcal{S}$, a queue of skinny triangles, and additionally an option `Force|Provisional`.

3. OUTPUT: The updated triangulation (with p inserted), segment set and queue of skinny triangles, or an exception which returns a segment mid point.

4. Let t be a triangle which p encroaches.

5. DETERMINE THE DELAUNAY CAVITY:

   - Beginning at t, search adjacent triangles to determine all triangles t' which p encroaches.
   - If removal of a triangle eliminates a segment from the triangulation split the segment and, (a) if this is a `Force`'d addition, add the mid point to the queue $\mathcal{P}$ and continue; otherwise, (b) if this is a `Provisional` addition, abort and return the segment mid point.
   - As each boundary edge (q, r) is located, determine if it is a segment. If so, and if p encroaches upon it then split the segment and (a) if this is a `Force`'d addition add the mid point to $\mathcal{P}$ and continue; otherwise, (b) if this is a `Provisional` addition, abort returning the mid point.

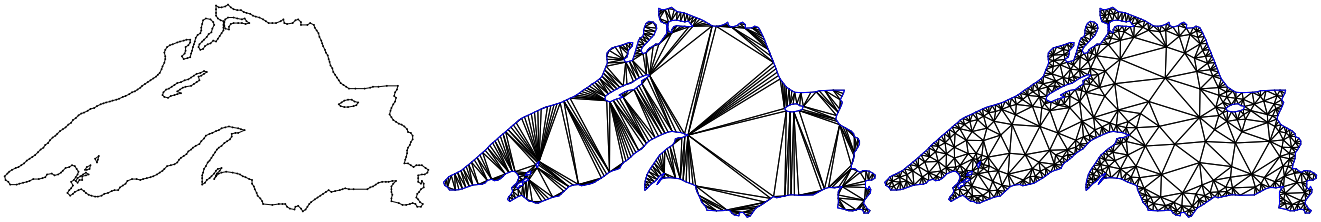
6. FILL THE CAVITY: Remove the triangles in the cavity from the mesh and for each edge (q, r) on the cavity boundary

   - Add the triangle (p, q, r) to the triangulation.
   - If (p, q) is a segment and r encroaches upon it then split (p, q) and add the mid point to $\mathcal{P}$. Similarly if (p, r) is a segment and is encroached upon by q it is split.
   - If (p, q, r) has radius edge ratio greater than ρ_2, add it to the queue of skinny triangles.

7. If p is the end point of a segment (p, q) not in the triangulation (q is in the triangulation, but not on the cavity boundary) split the segment and add the split point to the queue $\mathcal{P}$.

8. If $\mathcal{P} \neq \emptyset$, remove a point p from $\mathcal{P}$ and recursively call `add-point2d(p, ... ,Force)`.

Figure 2: Point Insertion Algorithm for the 2d Incremental Algorithm



(a) Input data. (b) Conforming triangulation. (c) Final triangulation.

Figure 3: The three steps of our incremental algorithm.

as indicated in Figure 4. The complete algorithm then consists of the following three steps which are illustrated in Figure 3

1. INITIALIZATION: After reading the input sets of points and segments, initialization is exactly as in the incremental Delaunay triangulation algorithm, *i.e.*, construct a bounding triangle that contains all the input points.
2. CONSTRUCT A CONFORMING TRIANGULATION: The initial call to `add-points( ... ,Force)` constructs a triangulation which conforms to the input and satisfies the induction hypotheses.
3. ELIMINATE SKINNY TRIANGLES: This is accomplished by a call to `rm-tri()`.

3.2 Induction Hypotheses

We wish to show that the algorithm described above is correct in the sense that it (a) solves the meshing problem, (b) terminates, and (c) produces well graded triangulations. Property (a) is established in this section by showing that the induction hypotheses are maintained. Termination will follow from the proof of (c) in Section 5 where we show that the vertex density is bounded below.

To establish the induction hypotheses we begin with the following elementary property of disks with a common chord which is readily observed in Figure 5. In the statement of the lemma an interior edge of a Delaunay triangulation refers to an edge having two adjacent triangles.

Lemma 3.1 *Let e be an interior edge in a Delaunay triangulation in the plane with empty diametral ball $B = B(e)$. If B_1 and B_2 are the circumballs of the two triangles adjacent to e, then $B_1 \cap B_2 \subset B \subset B_1 \cup B_2$. Moreover, B is empty if and only if the opposite angles of the two adjacent triangles are bounded above by $\pi/2$.*

Lemma 3.2 *After each point is inserted into the triangulation by* `add-point2d()`*, if a segment is in the triangulation then its diametral ball is empty.*

1. Procedure `rm-tri()`
2. INPUT: The current triangulation, the segment set, and the queue of triangles to be removed from the mesh.
3. OUTPUT: The triangulation and set of segments.
4. Let t be a triangle in the queue. If t it is no longer in the triangulation remove it from the queue and continue; otherwise, attempt to add its circumcenter, p, to the triangulation using the procedure `add-point2d(p, ... ,Provisional)`. This can have two outcomes:
   - The addition is successful in which case t has been eliminated, so can be removed from the queue of skinny triangles.
   - An exception is raised and a point p' is returned. At this stage the triangulation and queue of skinny triangles have not been modified. p' is now `Force`'d into the triangulation by a call to `add-point2d(p', ... ,Force)` which does update the triangulation, segment set and queue of skinny triangles.
5. Recursively call `rm-tri()` until the set of skinny triangles is emptied.

Figure 4: Function to Eliminate Triangles.

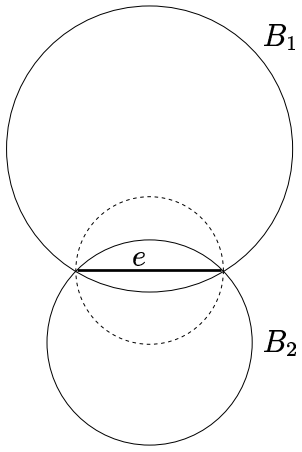


Figure 5: Empty diametral circles are contained in adjacent circumcircles.

Proof. Consider the addition of a vertex p to the triangulation. Inductively all of the segment diametral balls are empty; moreover, since edges of the bounding triangle are never segments, any segment in the triangulation has two adjacent triangles as in Lemma 3.1. If p is inside the diametral ball of a segment s Lemma 3.1 shows that it is inside at least one of the circumballs of an adjacent triangle and hence this triangle contains p in its circumball. If it is in both triangle circumballs then the segment is annihilated during the formation of the cavity, and hence split, which resolves the conflict. If it is only in one triangle circumball then s is on the boundary of the cavity and the algorithm explicitly checks to see if p is inside the diametral ball of boundary segments; if it is the segment is split which again resolves the conflict.

This argument shows that the diametral balls of segments originally in the triangulation remain empty (or the segments are split) when a new point p is inserted into the mesh. It is necessary to verify that if a new edge (p, q) is also a segment then it's diametral ball is also empty. Whenever such an edge is a segment we explicitly check to see if the opposite vertex of the triangles on each side encroach, and it is easy to verify that the empty circle criteria of Delaunay triangulations and Lemma 3.1 ensure that if any vertex encroaches upon (p, q) then so too will one of the vertices of the adjacent triangles. □

Corollary 3.3 *The two dimensional meshing algorithm maintains the induction hypotheses.*

Proof. Since the induction hypotheses trivially hold for the initial bounding triangle, it suffices to show that execution of the algorithm doesn't cause a violation.

(1) Whenever an edge is removed during the construction of the Delaunay cavity we explicitly check to see if it is a segment, and if so it is split (or the cavity construction is aborted). If the point being added to the mesh is the second point of a segment to be added to the mesh we explicitly check to see if the segment is present, and if it is not the segment is split. (Recall that when a segment is split the induction hypotheses trivially hold for the subsegments since the mid point will not be in the triangulation.)

(2) The empty diametral ball condition for segments was established in the previous lemma. □

Notice that during the execution of the algorithm there are only three types of points added to the triangulation: (0) input points, (1) segment mid points, and (2) circumcenters of skinny triangles. We next show that circumcenters added to the triangulation never get too close to segments.

Lemma 3.4 *Let p be the circumcenter of a skinny triangle that is added to the triangulation. Then p doesn't encroach upon any segment in the triangulation when it is added, and never encroaches upon any segment subsequently added to the triangulation.*

Proof. A circumcenter p is only added when all of the segments are currently edges in the triangulation, and the induction hypotheses guarantee that all their diametral balls

are empty. As in the proof of Lemma 3.2 this guarantees that the search will reveal any segments encroached upon by p, and in order for p to be added no such segments will exist. This establishes the lemma for the stage when p is added. Since subsequent segments are all refinements of segments in the triangulation when p is added, and since diametral balls of subseqments are contained in the diametral balls of parent segments, it follows that p will never encroach. □

4 An Incremental 3D Meshing Algorithm

4.1 Overview

We extend the incremental algorithm developed for the two dimensional problem to solve the meshing problem in three dimensions. Input points to the three dimensional algorithm will be handled as they were in the two dimensional algorithm; they are always inserted (`Force`'d) into the triangulation. As in the two dimensional algorithm, points will added to input features to ensure that they are represented in the final (3d) mesh. In order to see the analogy between the two and three dimensional algorithm it is convenient to recall how input edges are accommodated in the two dimensional algorithm.

- A "Delaunay triangulation" of each input edge is maintained (as a set of segments) independently of the two dimensional triangulation.
- The triangulation of an edge was refined in order to ensure it appeared in the two dimensional mesh. The refinement operation, while essentially an atomic operation in two dimensions, could be decomposed into the following steps:
  - If a "simplex" (segment) in the triangulation of an edge was found to violate the induction hypotheses it was queued for removal from the triangulation of the input edge containing it.
  - To annihilate a segment from the triangulation of an edge it's (circum)center was inserted into the edges triangulation.
  - The segment center was then queued for `Force`'d addition into the two dimensional mesh.

The three dimensional algorithm will maintain independent triangulations of each input edge and input face. We refer to triangles in the triangulation of an input face as *facets* to distinguish them from other triangles appearing in the three dimensional mesh. The facets are then accommodated in the fashion outlined above for the segments with one modification: if the addition of a facet circumcenter to a 2d triangulation encroaches upon a segment (in the same triangulation), then annihilation of the facet is postponed until the segment is annihilated. In essence facets are annihilated like the skinny triangles were

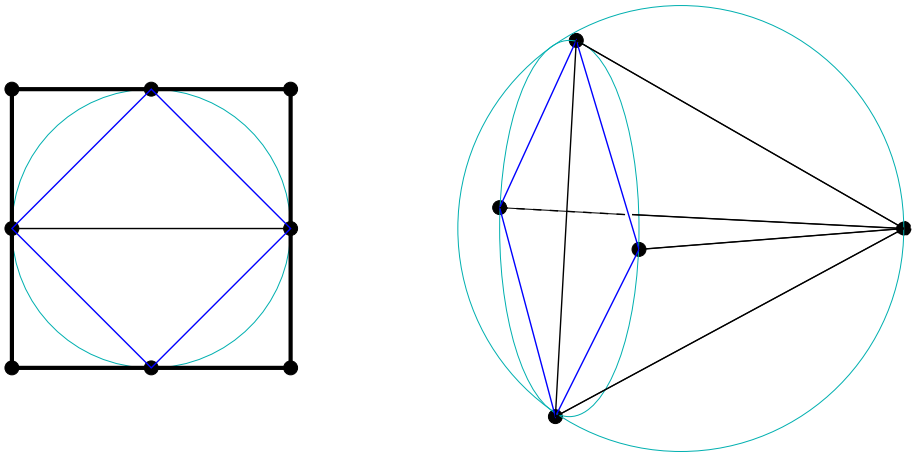


Figure 6: Distinct triangulations of cocircular points may occur in different meshes.

in two dimensions, and are processed with `rm-tri()`. If simplices in the triangulations of input edges and faces are queued for annihilation it is first necessary to insert circumcenters of the lowest dimensional simplex since these can eliminate a higher dimensional simplices. Below we precisely state the minimal (partial) order required.

Unlike the two dimensional case, angle bounds upon the input may not eliminate infinite recursion in three dimensions arising from degeneracy. This is illustrated in Figure 6 where a square face is required to be represented in a 3d mesh. Since the square has two possible Delaunay triangulations it is possible that distinct triangulations appear in the (independently maintained) 2d triangulation of the face and the 3d triangulation. This would cause the center of the square to be inserted into the 2d triangulation. Since refinement of the square by repeated insertion of triangle circumcenters always results in degeneracies (four or more cocircular points) this process may recurse indefinitely. Assuming exact arithmetic, we show that this recursion can be eliminated by inserting points into each mesh in the same order.

4.2 Description of the Algorithm

A triangulation of each input polytope will be maintained. While the triangulation of an input edge is trivial, it is convenient to view it as a triangulation since all input polytopes are processed similarly. An initial Delaunay triangulation of each face will be constructed which satisfies the two dimensional induction hypotheses stated in Section 3. Our two dimensional algorithm, with the following minor modifications, is used to maintain these triangulations:

- Whenever a segment is split, the split point is queued for addition to the mesh of

every polytope containing the edge.

- Similarly, whenever the circumcenter of a triangle in the triangulation of an input face is inserted into a two dimensional mesh, it is queued for addition to the 3d mesh.
- The queue of "skinny triangles" may now contain other facets queued for removal.

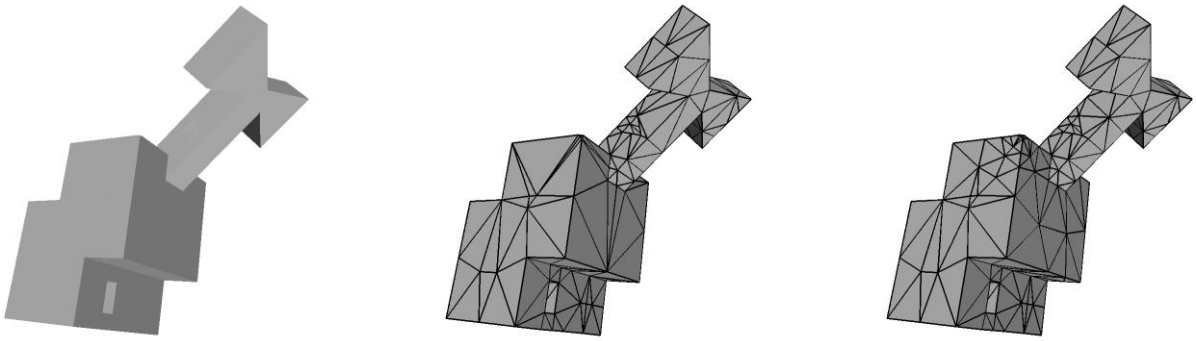
A work queue is maintained during the execution of the incremental algorithm. The queue contains pairs of the form (p, F), where p is a point to be added to a polytope F of the input PLS, or pairs of the form (K, F) where K is a simplex to be annihilated from the triangulation of F; $dim(K) = dim(F)$. A priority consistent with the following partial order is assigned to each pair, and tasks with smaller priority are precessed first.

Definition 4.1 (Work Order) *A total order is first assigned to points queued for insertion into the mesh. A fixed linear order of the input points is picked and these points have smallest priority values, and at the time a Steiner point is first inserted into any mesh of any polytope it is ordered to be greater than any point considered previously. Denote this order by $<_p$. We can now define the partial order $\prec$ on the work queue by*

- *If (p, F) a point-polytope and (K, F') is a simplex-polytope pair, then $(p, F) \prec (K, F')$ if $dim(F) \leq dim(F')$.*
- *If (p, F) and (q, F) are point-polytopes pairs with the same polytope, then $(p, F) \prec (q, F)$ if and only if $p <_p q$.*
- *If (K, F) and (K', F') are simplex-polytope pairs, then $(K, F) \prec (K', F')$ if $dim(F) < dim(F')$.*

This partial order states that points queued for addition to the mesh of a polytope F can be inserted at any time; however, they must be done in the order given by $<_p$. A simplex (K, F) can be processed for annihilation only if all queued points (p, F') with $dim(F') \leq dim(F)$ have been processed and all simplices (K', F') in polytopes having strictly lower dimension, $dim(F') < dim(F)$, have been processed. Notice that the reccursive calls in the two dimensional algorithm provide an elegant way of processing the points in an order consistant with the above. Since the three dimensional algorithm is more complex, below we explicitly implement a scheduler. In this instance it is assumed that the reccursive calls in the two dimensional algorithms are omitted. Our algorithm may attempt to add the circumcenter p of a triangle to the triangulation of an input face; however, such attempts may fail if p encroahces upon a segment. In this situation p is not included in the order $<_p$, it would only be included if the insertion succeedes.

The functions to add a point to the three dimensional mesh and to process a tetrahedra for removal are given in Figures 8 and 9 respectivly. The high level description of our incrimental algorithm is:



(a) Input PLS. (b) Conforming triangulation. (c) Final triangulation.

Figure 7: The three steps of our 3d incremental algorithm.

- Initialize the triangulations of each two and three dimensional polytope to be a large bounding simplex containing all of their input points. The initial triangulation of an input edge is the segment containing their two end points.
- Initialize the work queue with the all pairs of the form (p, F) for which p is an input point in the polytope F and $dim(F) > 1$.
- If the work queue is not empty, select a maximal element W.
  - If $W = (p, F)$ is a point-polytope pair, call `add-point`i`d( ... ,Force)` where $i = dim(F)$.
  - If $W = (K, F)$ is a simplex-polytope pair, (i) if K is a segment, split and queue the mid-point for addition to every polytope containing it. (ii) If K is a triangle or tetrahedra call `rm-tri()` or `rm-tet()` respectivly.

Figure 7 illustrates the three stages of our incremental meshing algorthm. Figure 7a shows that input PLS, and Figure 7b illustrates the conforming triangulation where all input edges and faces are first present in the mesh. Figure 7c shows that final mesh after all skinny tetrahedra have been removed.

4.3 Induction Hypotheses

As in the two dimensional algorithm, the function `add-point3d()` essentially augments the incremenatl Delaunay construction to maintain induction hypotheses which, in turn, guarantee that the input features are represented in the mesh.

Induction Hypotheses: A set of points, $\mathcal{P}$, segments, $\mathcal{S}$, and facets, $\mathcal{F}$, and a 3d Delaunay triangulation, $\mathcal{T}$, of a subset of $\mathcal{P}$ is maintained. These sets satisfy:

1. Procedure `add-point3d()`

2. INPUT: A point p to be added to the (3d) mesh and an option `Force|Provisional`. The PLS data structure is also required to facilitate interogation of other triangulations and insertion into the work queue.

3. OUTPUT: The updated data structure with the point p added to the triangulation, or an exception returning a lower dimensional segment or facet k' in the triangulation of an input polytope P'.

4. FORM THE CAVITY: The first step is to remove tetrahedra $\hat{T}$ which p encroaches upon, and determine the boundary of the cavity. The tetrahedra are determined by a search starting from a tetrathedra T for which $p \in B(T)$.

   - Beginning at T search the dual graph to find all $\hat{T}$ which p encroaches upon, $p \in B(\hat{T})$.
   - When traversing a face $\hat{t}$ of $\hat{T}$ check to see if it is a facet (a triangle in the triangulation of in input face P;). If so then
     - If this a `Force`d add, $(\hat{t}, P')$ to the work queue.
     - Otherwise, this is a `Provisional` add; raise an exception returning the pair $(\hat{t}, P')$.

5. Examine the removed cavity: if any removed edge is a segment, either raise an exception, returning the segment and the input edge containing it, or insert the segment into the queue of simplices to be anihilated from the input edge. depending on if the add is `Provisional` or `Force`d, as above.

6. Examine the cavity boundary; if any face or edge on the boundary is a facet or segment, and if p encroaches upon it, either raise an exception or queue it for anhilation as above.

7. FILL THE CAVITY: Remove tetrathedra in the cavity from the mesh and for each triangle t on the cavity boundary:

   - Form the tet $T' = (p : t)$ and insert it into the triangulation. Check if any face or edge of T is a facet or segment, and if it is encroached by a vertex of T. If so, queue the encroached segment/facet for anhilation.
   - If T' is of poor quality (radius edge ratio greater than ρ_3), queue it for anhilation.

8. If p is a vertex of a segment or a facet, and the other vertices of the segment/facet are present in the triangulation but the segment/facet is not, queue the segment or facet for anhilation.

Figure 8: Point addition routine for three dimensional meshes.

1. Procedure `rm-tet()`
2. INPUT: A tetrahedra T to be removed from the (3d) triangulation, and the PLS data structure to facilitate modification of the trianglation and work queue.
3. OUTPUT: Updated triangulation, and work queue.
4. If T is no longer in the triangulation return.
5. Let p be the circumcenter of T. Attempt to add p to the triangulation with `add-point3d(p, ... ,Provisional)`. There are two possible outcomes:
   - An exception was raised returning an encroached simplex t' in some subpolytope P'. Add (t', P') the work queue and return.
   - Otherwise the point addition succeeded and T is annihilated. Remover T from the work queue and return.

Figure 9: Function to anhilate tetrahedra for the Incremental 3-d Algorithm.

1. $\mathcal{F}$ is the union of the triangles in the (2d) Delaunay triangulations of the input faces. These triangulations satisfy the induction hypotheses for the two dimensional algorithm.
2. If the two end points of a segment are in the triangulation, then either the segment is present as an edge in $\mathcal{T}$ and its circumball is empty (contains no vertices of $\mathcal{T}$), or the segment has been queued for annihilation.
3. If the three vertices of a facet are in the triangulation, then either the facet appears as a triangle in $\mathcal{T}$ and its circumball is empty (contains no vertices of $\mathcal{T}$), or the facet is in the queue of simplices to be annihilated.

The remainder of this section is devoted to establishing that the induction hypotheses hold after each call to `add-point3d()` and to characterising the meshes our algorithm produces when geometric degeneracies exist. The following lemma is the three dimensional analogue of Lemma 3.1.

Lemma 4.2 *Let t be an interior triangle in a 3d Delaunay triangulation with empty circumball $B = B(t)$. If B_1 and B_2 are the circumballs of the two adjacent tetrahedra, then*

$$B_1 \cap B_2 \subset B \subset B_1 \cup B_2.$$

Similarly, if e is an interior edge in a 3d Delaunay triangulation having empty circumball $B = B(e)$, and $\{B_i\}$ are the circumballs of the tetrahedron containing this edge, then

$$\bigcap_i B_i \subset B \subset \bigcup_i B_i.$$

Lemma 4.3 *If a point p is added to the 3-d triangulation and it encroaches upon a segment or facet, t, then either this condition is detected during the addition of p (in which case the addition of p is aborted or t is queued for annihilation), or t has already been queued for annihilation.*

Proof. Suppose that t has not been queued for annihilation, then inductively it has not been encroached upon yet. Then the previous lemma shows that p must be in the circumsphere of a tetrahedron adjacent to t so that t is in the Delaunay cavity.

If p is in the circumsphere of both tetrahedra adjacent to a facet t, then t is "traversed" during the search to determine the Delaunay cavity, and the algorithm checks to see if traversed triangles are facets. Otherwise, the algorithm explicitly checks to see if edges in the cavity are segments, or if triangles on the cavity boundary are facets, and if so checks to see if p encroaches. □

Corollary 4.4 *The induction hypotheses are satisfied.*

Proof. (1) The two dimensional induction hypotheses were established in Corollary 3.3.

(2) and (3) When the last vertex of a segment or facet is added to the triangulation the algorithm explicitly checks to see if the segment or facet is present in the mesh. If it is not in the mesh it is queued for annihilation. If it is in the mesh the Delaunay property shows that points are inside its circumball if and only if a point in its link on the cavity boundary is inside the ball. The algorithm explicitly checks to see if the points in the link are in the ball, and if so queues the segment or facet.

It follows that at the time that the last vertex of a segment or facet is added to the mesh the induction hypotheses hold. Lemma 4.3 shows that if the addition of a subsequent point is in the circumball that this condition is detected and the segment or facet is queued for annihilation. □

Lemma 4.5 *Let $\mathcal{T}$ be a Delaunay triangulation of a convex planar face $F \hookrightarrow \mathbb{R}^3$ with bounding edges E, and suppose that the circumballs of the bounding edges are empty. Let $\mathcal{B}_1 = \cup_{e\in E} B(e)$ be the union of the circumballs of the edges in E and $\mathcal{B}_2 = \cup_{t\in F} B(t)$ be the union of the circumballs of triangles in F.*

Let $p \in F$ be specified and suppose that either (a) p is the mid point of a bounding edge, or (b) p is disjoint from $\mathcal{B}_1$. If $\mathcal{T}'$ is the triangulation of F resulting from the addition of p, define $\mathcal{B}'_1$ and $\mathcal{B}'_2$ to be the union of the circumballs of bounding edges and triangles respectively. Then

$$\mathcal{B}'_2 \cup \mathcal{B}'_1 \subset \mathcal{B}_2 \cup \mathcal{B}_1.$$

Proof. Clearly $\mathcal{B}'_1 \subset \mathcal{B}_1$, the non-trivial statement is to conclude that $\mathcal{B}'_2 \subset \mathcal{B}_2 \cup \mathcal{B}_1$.

Each new triangle, t' has p as one of its vertices and an edge $e \in \mathcal{T}$ opposite p. Consider first the situation where e is interior to F. Then there are triangles t_+ and $t_- \in \mathcal{T}$ adjacent

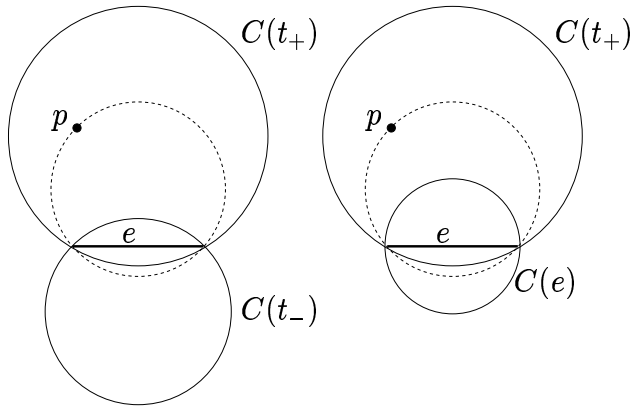


Figure 10: Circumcircles are nested

to e. For convenience assume that p is inside the circumball of t_+ and outside of the circumball of t_-, as in Figure 10a. Since the centers of the balls are all in the same plane, $B(t) \subset B(t_-) \cup B(t_+)$ will hold if the corresponding circles in the plane are are similarly nested, $C(t) \subset C(t_-) \cup C(t_+)$. Elementary geometry shows that the set of circles passing through two specified points (in this case the ends of e) are nested in this fashion.

Now suppose e is a bounding edge; then t_- and its accompanying sphere $S(t_-)$ don't exist. If H^+ is the half plane determined by e containing t^+ then $C(t) \cap H^+ \subset C(t_+)$ (see Figure 10b). Since p is outside the circumball of e the angle at p is acute. It follows that the portion of the circle through p and the end points of e on the opposite side of e is contained in the e's diametral circle so that $C(t) \subset C(t_+) \cup C(e)$. □

Corollary 4.6 *Circumcenters of tetrahedra that get added to the triangulation are never inside the circumballs of any segments or facets.*

Proof. At the time a tetrahedron circumcenter is added to the triangulation all of the facets and segments are present in the triangulation, and Lemma4.3 shows that at this time any encroachments would be detected. Since p only gets added if it is outside of the union of the circumspheres of all segments and facets it doesn't encroach any of them at insertion time. Lemma 4.5 shows that the union of all the circumspheres of segments and facets at each stage are nested by inclusion, so if a point is ever outside of the union it remains so. □

The following elementary lemma gives a precise characterization of the triangles that appear in a mesh when degeneracies occur.

Lemma 4.7 *(1) Let $\mathcal{T}$ be the Delaunay triangulation of a set of points $\mathcal{P} \subset \mathbb{R}^2$ and suppose that $B \subset \mathbb{R}^2$ is a ball, $B \cap \mathcal{P} = \emptyset$ and $\bar{B} \cap \mathcal{P} = \hat{\mathcal{P}}$. Suppose that $\mathcal{T}$ is constructed by the incremental algorithm that formed the cavity by removing triangles t if and only if*

the insertion point was in the open circumball of t, and that the points of $\hat{\mathcal{P}} = \{p_i\}_{i=1}^n$ were inserted in order of increasing index. If $i < j < k$ then the (p_i, p_j, p_k) are vertices of a triangle in $\mathcal{T}$ if and only if $\{p_m\}_{m=1}^{k-1} \subset H_{ijk}$ where H_{ijk} is the closed half space whose boundary contains p_i and p_j and is disjoint from p_k.

(2) Let $\mathcal{T}$ be the Delaunay triangulation of a set of points $\mathcal{P} \subset \mathbb{R}^3$ and suppose that $B \subset \mathbb{R}^3$ is a ball, $B \cap \mathcal{P} = \emptyset$ and $\bar{B} \cap \mathcal{P} = \hat{\mathcal{P}}$ lie in a plane F. Suppose that $\mathcal{T}$ is constructed by the incremental algorithm that formed the cavity by removing tetrahedra T if and only if the insertion point was in the open circumball of T, and that the points of $\hat{\mathcal{P}} = \{p_i\}_{i=1}^n$ were inserted in order of increasing index. Then $i < j < k$ then (p_i, p_j, p_k) are the vertices of a triangle in $\mathcal{T}$ if and only if $\{p_m\}_{m=1}^{k-1} \subset H_{ijk}$ where H_{ijk} is the closed half space in F whose boundary contains p_i and p_j and is disjoint from p_k.

Proof. Notice that if t was not created during the addition of p_k it is never created, since all new triangles contain the vertex being added. It then suffices to show that (i) at the time p_k is inserted into the triangulation the triangle $t = (p_i, p_j, p_k)$ is formed if and only if $\{p_m\}_{m=1}^{k-1} \subset H_{ijk}$, and (ii) the triangle is never annihilated subsequently.

We first establish statement (ii). Since $B \cap \mathcal{P} = \emptyset$ and in two dimensions the incremental algorithm only removes a triangle if a point is inside it's (open) circumball it follows that triangles with vertices on the boundary of B are never annihilated.

Consider next the three dimensional case where t is an interior triangle. Then t is annihilated if and only if both adjacent tetrahedra are annihilated, which would only occur during the addition of a point interior to both of their circumballs. Lemma 4.2 shows that p would then be interior to B which is excluded by hypothesis. If t is on the convex hull of a triangulation $\mathcal{T}'$ of a subset $\mathcal{P}' \subset \mathcal{P}$, then t will only be annihilated by the addition of a point interior to $B \cap F$ which is excluded by hypotheses.

To establish (i) we need to show that the edge $e = (p_i, p_j)$ is on the boundary of the cavity formed during the addition of p_k if and only if $\{p_m\}_{m=1}^{k-1} \subset H_{ijk}$. The proof proceeds inductively, the base case of $k = 3$ following from the Delaunay property which guarantees that t is present in the mesh if $\bar{B}(t) \cap \mathcal{P} = \{p_1, p_2, p_3\}$. When $k > 3$ the edge $e = (p_i, p_j)$ on the convex hull of $\{p_m\}_{m=1}^{k-1}$ for which $p_k \notin H_{ijk}$ is unique.

Let $\mathcal{T}'$ be the triangulation prior to the addition of p_k and $\mathcal{P}'$ be its vertex set. Since $B \cap \mathcal{P} = \emptyset$ and $\bar{B} \cap \mathcal{P}' = \{p_m\}_{m=1}^{k-1}$ are all coplanar it follows that $\{p_m\}_{m=1}^{k-1}$ are the vertices of a Delaunay polytope P' (see the discussion in Section 2). The triangulation $\mathcal{T}'$ induces a triangulation of P'; in particular, edges on the convex hull of P' are represented in $\mathcal{T}'$. The half space condition guarantees that e is an edge on the convex hull of P' so is an edge of $\mathcal{T}'$. After p_k is inserted into the mesh $\{p_m\}_{m=1}^{k}$ are the vertices of a Delaunay polytope P in the new mesh, and since (p_k, p_i) and (p_k, p_j) are on the convex hull of P they will exist in the augmented triangulation. To finish the proof it suffices to show that e is not annihilated when p_k was inserted.

Since $k > 3$, the the triangulation of P' induced by $\mathcal{T}'$ contains a triangle of the form $t' = (p_i, p_j, p_\ell)$. If $e = (p_i, p_j)$ was annihilated during the addition of p_k then so too was t',

and this would contradict (ii). □

Corollary 4.8 *At the time a simplex K is queued for annihilation from the mesh of the input feature F containing K ($dim(F) = dim(K)$) either (a) K was encroached, or (b) K was skinny, or (c) the circumsphere of K contains a vertex $v \notin F$ from the triangulation of anther feature.*

Proof. Non-skinny simplices are queued for annihilation in three situations: when a point is being inserted (i) any segments or facets interior to the cavity are queued, (ii) facets and segments on the cavity boundary encroached by the new point are queued, and (iii) the last vertex of K is being inserted into a triangulation and K does not appear. Clearly we need only consider cases (i) and (iii).

Suppose that an "unqueued" facet or segment is interior to a cavity being formed during the addition of a point p to the mesh containing K. The induction hypotheses guarantee that circumball of the facet/segment is empty. Lemmas 4.2 and 3.1 show that p is inside the circumball of K and hence encroaches.

For case (iii), let K be a segment or facet that is not encroached and does not appear in a mesh containing its vertices. If the circumsphere $S(K)$ only contains the vertices of K then K will appear in any mesh containing the vertices, so $S(K)$ must contain other points. If K is a segment these points are not collinear with K so must belong to some other feature. If K is a facet, Lemma 4.7 shows that if all of the points on $S(K)$ belong to the face F containing K, then K will appear in the three dimensional mesh. It follows that $S(K)$ must contain some point not in F. □

5 Correctness of the Algorithms

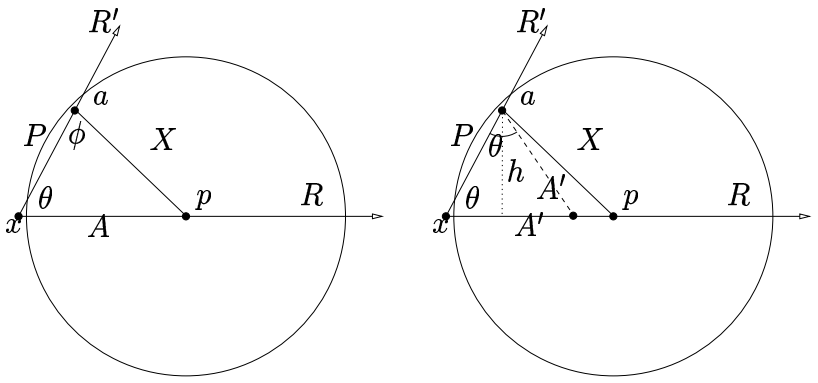
In this section we prove that the two and three dimensional meshing algorithms proposed above terminate and have a vertex density determined by the input. The two dimensional proof is obtained by omitting those portions of the proof concerning facets and tetrahedra ($CD = 2$ or 3 below). We begin with the following elementary lemma which is useful for determining how the mesh density changes when segments are split.

Lemma 5.1 *Let two rays, R and R' emanate from a point x have angle $\theta \geq \pi/4$ between them. If a ball B with center p on R intersects R' and does not contain x then*

$$\frac{dist(a,x)}{dist(a,p)} \leq 2\cos(\theta), \qquad \forall\, a \in B \cap R'.$$

Note that if $\theta \geq \pi/2$ then $B \cap R' = \emptyset$.

Proof.



(a) as stated (b) bounded by the isosceles

Figure 11: Proof of Lemma 5.1; The lemma as stated is shown in (a). It can be shown that $\theta \leq \phi$, so we may draw the isosceles triangle, as in (b) with base angle θ to get the desired bound. The altitude is also drawn in (b), and both triangle legs will be to its right.

We wish to show that $P/X \leq 2\cos(\theta)$ where P, X are as in Figure 11. We first note that $X \leq A$ because x is not inside the ball, but a is. Using the sine identity we find that $\sin\theta \leq \sin\phi$, implying that $\theta \leq \phi$. We can then draw an isosceles triangle of base angle θ and base (x, a), with side lengths A'. Since $\pi/4 \leq \theta$ the apex angle of this isosceles triangle will be acute, so the altitude h, the leg A' and the leg X are ordered left to right as shown in Figure 11(b), and thus $A' \leq X$. Using the cosine relation it is easy to show that $P/A' = 2\cos(\theta)$ so $P/X \leq 2\cos(\theta)$ as desired. □

The functions occurring in the following definition will enable us to bound the density of vertices produced by our algorithm.

Definition 5.2 *Let $\mathcal{W}$ be a PLS with dimeinson d.*

- *The* **local feature size** *at x in W, $\text{lfs}(x) = \text{lfs}_{\mathcal{W}}(x)$ is the distance to the second nearest disjoint polytope in W, i.e. the radius of the smallest closed ball centered at x which intersects two feature in W which do not intersect.*
- *For each integer $0 \leq i \leq d$ let $\mathcal{W}_i$ be the PLS consisting of the the polytopes in $\mathcal{W}$ having dimension at most i. Then $\text{lfs}_i(x) = \text{lfs}_{\mathcal{W}_i}(x)$.*

Notice that $\text{lfs}(.) = \text{lfs}_d(.)$ and $\text{lfs}_0(.)$ is the distance to the second nearest vertex in the PLS. Below lfs will always be the local feature size determined by the input data, $\mathcal{P}_0$ and $\mathcal{S}_0$ (and $\mathcal{F}_0$ in three dimensions). If the local feature size is required for another PLS (such as the final mesh) the function will be annotated appropriately. Notice that $lfs()$ is a

Lipschitz function with constant 1; that is, $|\text{lfs}[x] - \text{lfs}[y]| \leq |x - y|$, where $|\cdot|$ on the right hand side of this expression denotes Euclidean distance.

The following elementary corollary will be used frequently in our proof of termination.

Corollary 5.3 *Let p, a and x be three points in a plane or $\mathbb{R}^3$ and with $\theta = \angle pxa \geq \pi/4$. Suppose that $a \in \bar{B}_r(p)$ and $r \leq |p - x|$, and $\text{lfs}[a] \leq C|a - x|$, then $\text{lfs}[p] \leq (1 + C\rho_\theta)r$ where $\rho_\theta = 2\cos(\theta)$.*

Proof. The Lipschitz property of the local feature size gives

$$\begin{aligned} \text{lfs}[p] &\leq \text{lfs}[a] + |p - q| \\ &\leq C|a - x| + r \\ &\leq (1 + C|a - x|/|p - q|)r \\ &\leq (1 + C\rho_\theta)r \end{aligned}$$

□

The following definitions are useful in the proof of our main theorem.

Definition 5.4 *Let $\mathcal{W}$ be a d-dimensional PLS, and $p \in \mathbb{R}^d$.*

- *The* containment dimension *of p in $\mathcal{W}$, $CD(p)$, is the dimension of the lowest dimensional polytope in $\mathcal{W}$ that contains p. If no polytope contains p then $CD(p)$ is the dimension of the underlying space, i.e. d.*
- *If an attempt to add the circumcenter p of a simplex K queued for annihilation from the mesh of a polytope $F \in \mathcal{W}$ failed due to p encroaching upon a segment or facet also in the mesh of F, we say that p* yields *to the segment or facet.*

Theorem 5.5 *Suppose that the input PLS satisfies:*

- *the angles between any two non-disjoint input segments is bounded below by $\theta > \pi/3$;*
- *the angle between a non-disjoint face and segment is bounded below by $\phi > \cos^{-1}(1/2\sqrt{2})$;*
- *if two faces meet at a point x, then the angle between any pair of rays emanating from x into each plane is bounded below by $\phi > \cos^{-1}(1/2\sqrt{2})$;*
- *the dihedral angle between non-disjoint faces is bounded below by $\pi/2$.*

Let the edge-radius ratio threshold for splitting skinny triangles satisfy $\rho_2 > \sqrt{2}$, and the threshold for splitting skinny tetrahedra be $\rho_3 > 1/2\sqrt{2}$. Then there exist constants C_i, $0 \leq i \leq 3$, such that at the time an attempt is made to insert the circumcenter p of a simplex K into the (unique) mesh $\mathcal{W}'$ of dimension $dim(K)$, then $\text{lfs}[p] \leq C_i \text{lfs}_{\mathcal{W}'_0} p$ where $i = CD(p)$.

Proof. In order to minimize the number of subscripts, we will write $\mathrm{lfs}_{\mathcal{W}_0'}(x) = N(x)$. That is, $N(x)$ is the distance to the second nearest vertex from x in $\mathcal{W}'$, where $\mathcal{W}'$ is the "current" mesh into which the insertion of p is attempted.

It may happen that $CD(p) < dim(K)$ if the circumcenter of K lies in a sub-face which happens to be a facet or segment. In this situation insertion will fail due to the obvious encroachment, and points will only be inserted into the mesh when $CD(p) = dim(K)$. Typically we will prove that $\mathrm{lfs}[p] \leq C_{dim(K)} N(p)$, for $dim(K) \geq 1$, which suffices provided $C_1 \geq C_2 \geq C_3$.

We proceed inductively beginning with the input points. We consider the singleton sets of input points to be zero dimensional triangulations, and since distinct points are disjoint features it suffices to take $C_0 = 1$. We then consider the insertion of circumcenters of simplices having dimension strictly positive dimension.

Consider the attempted insertion of the circumcenter p of a Delaunay simplex K_p having dimension $i > 0$. Denote the circumball of K_p by $B_p = B(K_p)$, the circumsphere by $S_p = S(K_p)$, and their radii r_p. Let $\mathcal{P}_{insert} = \mathcal{P} \cap B_p$ be the (possibly empty) set of points that encroach upon K_p *at the time insertion of* p *is attempted.* When $\mathcal{P}_{insert} \neq \emptyset$ it follows that K_p is encroached and we use induction to establish the estimate $\mathrm{lfs}[p] \leq C_i N(p)$. When $\mathcal{P}_{insert} = \emptyset$ so K_p is not encroached it is necessary to also consider the situation that existed at the time K_p is queued.

Case $\mathcal{P}_{insert} \neq \emptyset$: Of the points closest to p in $\mathcal{P}_{insert}$ let a be the one of minimal containment dimension, then $N(p) = |a - p|$.

Let F_p and F_a be the input features (input points, edges or faces) containing p and a respectively. If $F_p \cap F_a = \emptyset$ they are disjoint input features, so $\mathrm{lfs}[p] \leq |a - p| = N(p) \leq C_i N(p)$ provided $C_i \geq 1$. In particular, if $a = F_a$ is an input point then it is a disjoint feature (it couldn't be inside B_p otherwise). It then suffices to consider the situation when $j = \dim(F_a) > 0$ and $F_a \cap \mathbb{F}_p \neq \emptyset$.

Subcase $i = j = 1$: Letting x be the point of intersection of the two segments, then x is an input point and was present at the time a was inserted into the mesh. Inductively $\mathrm{lfs}[a] \leq C_1 N(a) \leq C_1 |a - x|$, then Corollary 5.3 then gives

$$\mathrm{lfs}[p] \leq (1 + C_1 \rho_\theta) N(p),$$

so the induction hypotheses hold provided

$$1 + C_1 \rho_\theta \leq C_1. \tag{1}$$

Subcases $i = 1$, $j = 2$ and $i = 2$, $j = 1$: First suppose that $F_p \cap F_a$ is a point x where the input edge and face meet. The angle condition on the input guarantees that in the plane of (p, x, a) the angle $\angle(p, x, a)$ is bounded below by ϕ. Upon invoking Corollary 5.3 and repeating the above argument we find that the induction hypotheses are satisfied provided inequalities $1 + C_j \rho_\phi \leq C_i$, and $1 + C_i \rho_\phi \leq C_j$ that is

$$1 + C_2 \rho_\phi \leq C_1, \text{ and } 1 + C_1 \rho_\phi \leq C_2. \tag{2}$$

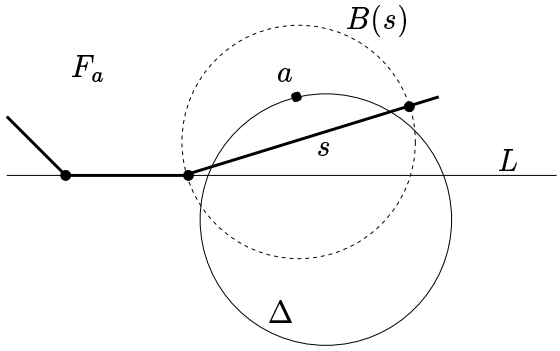


Figure 12: Points in intersecting Planes aren't close

The other possibility is that $F_p \cap F_a = F_p$, or $F_p \cap F_a = F_a$, so p and a are in a common mesh. The definition of $\mathcal{P}_{insert}$ then requires $a \in B_p$ (the open ball), and Lemma 3.4 shows that circumcenters of triangles are never inside the diametral balls of segments ($i = 1$, $j = 2$), and the Delaunay property excludes the possibility that a point is inside the circumball of a triangle ($i = 2$, $j = 1$).

Subcase $i = j = 2$: If the two input faces meet at a point x, ($\dim(F_a \cap F_p) = 0$) then using the angle condition on the input and Corollary 5.3 as above shows that the induction hypotheses are satisfied provided

$$1 + C_2\rho_\phi \leq C_2. \tag{3}$$

If the input faces meet along an edge e ($\dim(F_a \cap F_p) = 1$) we show that the lower bound of $\pi/2$ on the dihedral angle excludes the possibility that a is the closest point to p of minimal containment dimension. To argue this let $\mathbb{P}$ be the plane containing F_a oriented so that F_a is in the half space above the line of intersection L containing $F_p \cap F_a$ as illustrated in Figure 12. Let Δ be the intersection of the closed ball centered at p of radius $|p - a|$ with the plane $\mathbb{P}$. Then Δ is a closed disk with a on its boundary, and the lower bound of the dihedral angle between F_a and F_p requires the center of Δ to lie on or below L. Since a is interior to F_a and $a \in \Delta \cap \mathbb{F}_a$ it follows that Δ meets a bounding segment s of F_a; however, it can not contain any boundary vertices since, of the points closest to p, a was chosen to be the one with minimal containment dimension. It follows that the intersection of Δ with boundary of $\mathbb{F}_a$ only contains points interior to s.

We now get a contradiction by recalling that the bounding edges of an input face have empty circumballs; in particular, Lemma 3.4 shows that a can not be in the (open) the circumball, $B(s)$, of s (it would have yielded to s). Since $B(s)$ has center on or above the line L and Δ is a disk with center on or below the line L that doesn't meet the end points of s it follows that $\Delta \cap F_a \subset B(s)$, and $a \in \Delta \cap F_a \subset \bar{B}(s)$ gives the contradiction.

Subcase $i = 3$: The order in which points are processed eliminates this case. Circumcenters of tetrahedra are proposed for insertion after all other points have been added into the

mesh and the Delaunay property excludes the presence of encroached tetrahedra.

Subcase $j = 3$: This case never arises since Corollary 4.6 shows that a tetrahedra circumcenter a would never have been inserted into the mesh if it encroaches upon a facet or a segment.

Case $\mathcal{P}_{insert} = \emptyset$: If K_p is not encroached *by a point in a mesh* at the time its circumcenter is proposed for addition we need to recall the situation which caused K_p to be queued. Lemma 4.8 shows that either K_p was encroached, skinny, or "degenerate". We consider these situations separately. Since $\mathcal{P}_{insert} = \emptyset$ we know that $N(p) = r_p$, the radius of the circumball of K_p.

Subcase K_p Encroached: Let a be the point that encroached upon K_p causing it to be queued, and consider the mesh of the input face, F_a, containing a. In order for a to encroach upon K_p, K_p must be contained in the mesh of F_a. Since $a \notin \mathcal{P}_{insert}$ it was not inserted into the mesh, so it must have yielded to K_p, in which case $CD(a) = j > i = CD(p) > 0$. It follows that a is the circumcenter of a Delaunay simplex K_a of dimension $j > i$ with circumradius r_a. At the time p was queued (and insertion of a was attempted), the circumball of K_a was empty, and the induction hypotheses (2) (2d) and (2), (3) (3d) guarantee that the circumball of K_p contained no other points in the mesh of F_a. This configuration is illustrated in Figure 13.

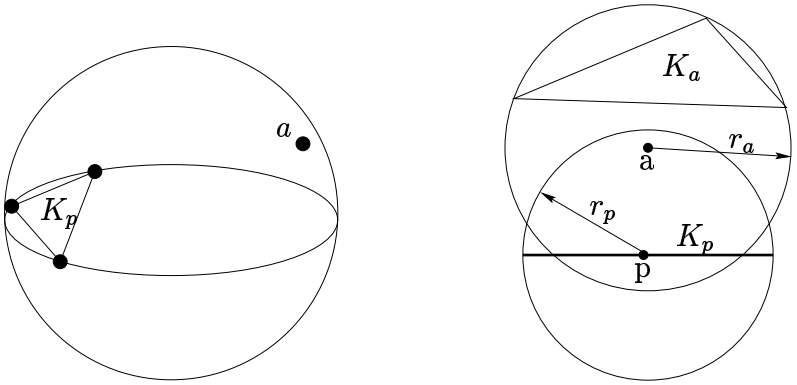


Figure 13: (a) 3d circumcenter encroaching a facet, (b) 2d circumcenter encroaching a segment.

Inductively we may assume that $\text{lfs}[a] \leq C_j N(a) = C_j r_a$ and since a is inside B_p, but B_a contains no vertices of F_p, it follows that $r_a \leq 2r_p$. Then

$$\text{lfs}[p] \leq \text{lfs}[a] + |a - p| \leq 2C_j r_p + r_p = (1 + 2C_j)N(p).$$

The induction hypotheses will be satisfied provided

$$1 + 2C_j \leq C_i \qquad \forall\, j > i > 0. \tag{4}$$

When $i = 1$, K_p is a segment and a sharper estimate holds due to the fact that the end points of the segment are antipodal, as in Figure 13. In this situation we can conclude that $r_a \leq \sqrt{2} r_p$ which leads to the requirement

$$1 + \sqrt{2} C_j \leq C_1 \qquad \forall\, j > 1. \tag{5}$$

Subcase Skinny Polytopes: If K_p is a skinny polytope, notice that at the time an attempt is made to insert its circumcenter the circumball B_p of K_p may not be empty; the Delaunay property only only guarantees that its intersection with F_p, the face containing K_p, is empty. However, the case where B_p is non-empty was addressed previously when we considered $\mathcal{P}_{insert} \neq \emptyset$. (In essence, this simplex would have been split even if it wasn't skinny). It follows that we need only address the case where B_p is empty.

Since the radius edge ratio of K_p is greater than ρ_i it follows that there is an edge $e = (a, b)$ of F_p having length $|e| < r_p/\rho_i$. If both a, b are input points, then $\mathrm{lfs}[b] \leq |b - a|$. Otherwise, without loss of generality assume b was added to F_p after a, then inductively, at the time b was added, $\mathrm{lfs}[b] \leq \max(C_i) N(b) \leq \max(C_i)\, |e|$. Then

$$\mathrm{lfs}[p] \leq \mathrm{lfs}[b] + |p - b| \leq (\max(C_i)/\rho_i + 1) r_p,$$

so the induction hypotheses hold provided

$$1 + \max_j (C_j)/\rho_i \leq C_i, \qquad i > 1. \tag{6}$$

Note this assumes $\max(C_i) \geq 1$ to take care of the case a, b are input.

Subcase Degenerate Points: Finally, consider the situation when a simplex K_p with circumsphere S_p was queued for annihilation because it never appeared in a higher dimensional mesh due to degeneracy. Let F_p be the input face containing K_p and write $CD(p) = dim(F_p) = i$. Since only segments or facets are required to be present in other meshes $i = 1$ or 2.

Lemma 4.8 shows that there are points not in the mesh of F_p on S_p. If any of these points are on a input polytopes disjoint from F_p, then such a point and F_p are disjoint features so $\mathrm{lfs}[p] \leq r_p$, where r_p is the circumradius of K_p, and the induction hypotheses hold provided $C_i \geq 1$. We then need only consider the situation where all of the points on S_p belong to input features having a common intersection, and we let a be one of them having maximal containment dimension, $CD(a) = j$.

- Suppose $j = CD(a) > CD(p) = i$ and $F_p \subset F_a$. When claim that, at the time a was added to the mesh of F_a, at least one vertex in $F_p \cap S_p$ was present in the mesh of F_a. If not, the order that the queues are processed guarantee that the closed ball $\bar{B}_p \cap F_p$ is empty, so is contained in an (open) circumball of a simplex $K \subset F_p$. Then $a \in B(K) \cap F_a$ gives rise to a contradiction; a would have yielded to K. If F_p is a segment then both vertices were present since every point of the closed ball $\bar{B}_p$, except the end points of K_p, is in the open ball of a super segment.

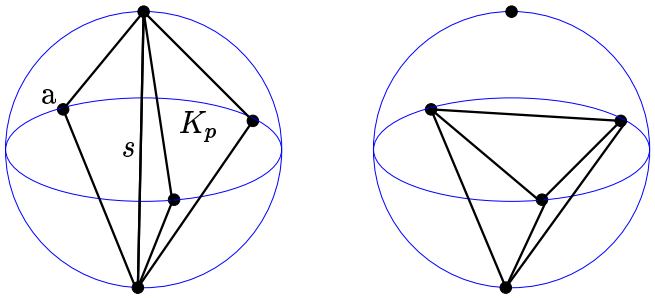


Figure 14: Degenerate facets may not appear as faces of tetrahedra; however, their circumcenter lies on a segment.

We may then use repeat the "Subcase Encroached" argument above to conclude that the induction hypotheses are satisfied provided equations 4 and 5 hold.

- When $i = j = 1$ we may repeat the argument given for $\mathcal{P}_{insert} \neq \emptyset$ for the corresponding case and deduce that the induction hypotheses hold if equation (1) is satisfied.

- When $(i, j) = (1, 2)$ or $(i, j) = (2, 1)$ and $F_a \cap F_p = \{x\}$ meet at a point ($dim(F_a \cap F_p) = 0$) we may repeat the argument given for $\mathcal{P}_{insert} \neq \emptyset$ to get the induction hypotheses holding provided equations (2) hold.

  If $dim(F_a \cap F_p) = 1$, then a is not in the mesh of F_p and the order in which the work queue is processed imply $(i, j) = (1, 2)$, i.e. K_p is a segment contained in the input face F_a. Then $j > i$ and the intersection properties of the PLS guarantee that $F_p \subset F_a$, and this situation was considered above.

- The remaining case of $i = j = 2$ requires a separate argument.

If $i = j = 2$ we use the angle condition on the input to claim that there exists a segment segment $s \subset F_p \cap F_a$ for which $B_p = B(s)$ so that $p \in s$ and $CD(p) = 1$ (see Figure 14). Assuming this, it follows that the induction hypotheses are satisfied provided $1 + \sqrt{2}C_2 \leq C_1$. To see this, notice that at the time a was inserted into its mesh the segment s must have been present. Otherwise s was contained in a super-segment and since a is on the boundary of $B(s)$ it would have been interior to the circumball of this larger segment, so would have yielded. It follows that when a was inserted $N(a) \leq \sqrt{2}r_p$ so that

$$\begin{aligned} \text{lfs}[p] &\leq \text{lfs}[a] + r_p \\ &\leq C_2 N(a) + r_p \\ &\leq (1 + \sqrt{2}C_2) r_p \\ &\leq (1 + \sqrt{2}C_2) N(p) \end{aligned}$$

Remark: It is also possible to argue that the segment doesn't appear in the (3d) mesh, so was queued to be split. Since segment centers are inserted prior to triangle circumcenters p would have been inserted into F_p annihilating K_p.

To establish the claim, observe that if the angle between F_p and F_a is bounded below by $\pi/2$ then the intersection of F_a with any empty ball B having center in F_p is contained in one of the diametral balls $B(s)$ of a segment s bounding F_a (see Figure 12). Moreover, $\bar{B} \cap \bar{B}(s) \subset B(s) \cup s$ except in the situation where (i) $s \subset F_p \cap F_a$ and (ii) $B = B(s)$. Letting $B = B_p$ we find $\bar{B}_p \subset B(s) \cup s$ unless the (i) and (ii) hold. Since $CD(a) = 2$ it follows that $a \notin s$, and since $a \in F_a \cap \bar{B}_p$ it follows that $\bar{B}_p \not\subset B(s) \cup s$, so (i) and (ii) must hold. In particular, $B_p = B(s)$, so has center $p \in s$ and $CD(p) = 1$ as claimed.

Computing the Constants: The constraints $1 + 2C_j \leq C_i$ for $j > i > 0$ show that $C_1 > C_2 > C_3$. Using these identities and the fact that $\rho_\theta, \rho_\phi < 1$ some of the inequalities in equations (1-6) are redundant; the remaining inequalities are

$$1 + \sqrt{2}C_2 \leq C_1,$$

$$1 + 2C_3 \leq C_2$$

$$1 + C_1/\rho_3 \leq C_3$$

$$1 + C_1/\rho_2 \leq C_2, \qquad 1 + C_1\rho_\phi \leq C_2,$$

and $C_1 \geq 1/(1-\rho_\theta)$, $C_2 \geq 1/(1-\rho_\phi)$.

The first three inequalities have a solution with equality provided $\rho_3 > 2\sqrt{2}$.

$$C_1 = \frac{\rho_3(1+3\sqrt{2})}{\rho_3 - 2\sqrt{2}}, \qquad C_2 = \frac{3\rho_3 + 2}{\rho_3 - 2\sqrt{2}}, \qquad C_3 = \frac{\rho_3 + (1+\sqrt{2})}{\rho_3 - 2\sqrt{2}}.$$

These constants are greater than one and tend to infinity as ρ_3 tends to $2\sqrt{2}$ from above. We can then compute

$$\rho_2 \geq (\rho_3/2)\frac{1+3\sqrt{2}}{\rho_3 + (1+\sqrt{2})}, \qquad \rho_\theta \leq \frac{\sqrt{2}(3\rho_3+2)}{\rho_3(1+3\sqrt{2})},$$

and

$$\rho_\phi \leq \min\left(2\,\frac{\rho_3 + (1+\sqrt{2})}{\rho_3(3\sqrt{2}+1)}, 2\,\frac{\rho_3 + (1+\sqrt{2})}{3\rho_3+2}\right) = 2\,\frac{\rho_3 + (1+\sqrt{2})}{\rho_3(3\sqrt{2}+1)}.$$

As $\rho_3 \searrow 2\sqrt{2}$, ρ_2, ρ_θ and ρ_ϕ tend monotonically to

$$\rho_2 \to \sqrt{2}, \qquad \rho_\phi \to 1/\sqrt{2}, \text{ and } \rho_\theta \to 1.$$

We then find solutions provided $\rho_3 > 2\sqrt{2}$, $\rho_2 > \sqrt{2}$, and $\theta > \cos^{-1}(1/2) = 60^o$, $\phi > \cos^{-1}(1/2\sqrt{2}) \simeq 69.3^o$. □

Corollary 5.6 *Suppose the algorithm is run on input that satisfies the hypotheses of the theorem to produce a three dimensional mesh $\mathcal{W}$. Then for all vertices $p \in \mathcal{W}_0$ in the mesh* $\text{lfs}[p] \leq (1 + C_1)\text{lfs}_{\mathcal{W}_0}(p)$.

Proof. Let a be the point nearest p in the mesh. If a and p are both input points, then $\text{lfs}[p] \leq |a - p| = N(p)$. Otherwise consider which was added first. If a was added first, Lemma 5.5 shows that when p was added $\text{lfs}[p] \leq C_i\,|p - a| \leq C_1 N(p)$.

Similarly, if p was added first, then at the time a was added $\text{lfs}[a] \leq C_1 N(a) = C_1 N(p)$. Using the Lipschitz continuity of lfs we deduce $\text{lfs}[p] \leq |a - p| + \text{lfs}[a] \leq (1 + C_1)N(p)$ and the corollary follows. □

Remark: The following lemma from Shewchuk [20] can be used to obtain slightly sharper estimates for the constants above.

Lemma 5.7 *Let $\mathcal{T}$ be a Delaunay triangulation of a set of points $\mathcal{P}$ and let the planar face F is represented as the union of triangles in $\mathcal{T}$. Suppose that the circumballs of all triangles and bounding edges of F are empty, and let p the the circumcenter of a tetrahedra in $\mathcal{T}$.*

If p encroaches upon a triangle in F then (i) if the orthogonal projection of p onto the plane containing F is inside F, then p also encroaches upon the triangle(s) containing the orthogonal projection; otherwise (ii) p encroaches upon a bounding segment.

If Step 5 of `rm-tet()` is modified to queue the facet or segment guaranteed by this lemma, instead of the first facet discovered by `add-point3d()`, the extreme situation shown in Figure 13 would not arise. In this situation inequality (4) with $(i, j) = (2, 3)$ can be sharpened to $1 + \sqrt{2}C_3 \leq C_2$ which results in better constants. Specifically, the modified system of inequalities will have a solution if $\rho_3 > 2$, and recomputing the constants gives

$$C_1 = \frac{\rho_3(3 + \sqrt{2})}{\rho_3 - 2}, \qquad C_2 = \frac{\rho_3 + \sqrt{2}(\rho_3 + 1)}{\rho_3 - 2}, \qquad C_3 = \frac{\rho_3 + (1 + \sqrt{2})}{\rho_3 - 2}.$$

Again these constants tend to infinity as ρ_3 tends to 2 from above, and we require

$$\rho_2 \geq (\rho_3/\sqrt{2})\,\frac{3 + \sqrt{2}}{\rho_3 + (1 + \sqrt{2})}, \qquad \rho_\theta \leq \frac{\rho_3(2 + \sqrt{2}) + 2}{\rho_3(3 + \sqrt{2})}, \qquad \rho_\phi \leq \sqrt{2}\,\frac{\rho_3 + (1 + \sqrt{2})}{\rho_3(3 + \sqrt{2})}.$$

As $\rho_3 \searrow 2$, ρ_2, ρ_θ and ρ_ϕ tend monotonically to the values computed previously.

5.1 Size Optimality

Corollary 5.6 guarantees termination of our algorithm; however, unlike the two dimensional situation doesn't directly yield size optimality. Optimality in two dimensions follows from

the observation that if a triangulation $\mathcal{T}$ has bounded aspect ratio, then $c\,\mathrm{lfs}_{\mathcal{T}}(x) \leq \mathrm{lfs}_{\mathcal{T}_0} x \leq C\,\mathrm{lfs}(x)$, and clearly $\mathrm{lfs}_{\mathcal{T}}(x) \leq \mathrm{lfs}(x)$. Since various integrals of the local feature size bound the number of vertices in a mesh with bounded aspect ratio [7, 17] these inequalities imply size optimality.

The following definition characterises the optimality criteria satisfied by the three dimensional meshes computed using Delaunay refinement.

Definition 5.8 *Let $\mathcal{W}_{in}$ be a three dimensional PLS, then a mesh $\mathcal{W}$ is* pointwise optimal *with respect to $\mathcal{W}_{in}$ if (i) each polytope in $\mathcal{W}_{in}$ is represented in $\mathcal{W}$ as a union of simplices, (ii) each simplex of $\mathcal{W}$ has a bounded radius edge ratio, and (iii) there exists a constant $c > 0$ such that $c\,\mathrm{lfs}_{\mathcal{W}_{in}}(x) \leq \mathrm{lfs}_{\mathcal{W}_0}(x) \leq \mathrm{lfs}_{\mathcal{W}_{in}}(x)$ for each $x \in \mathbb{R}^3$*

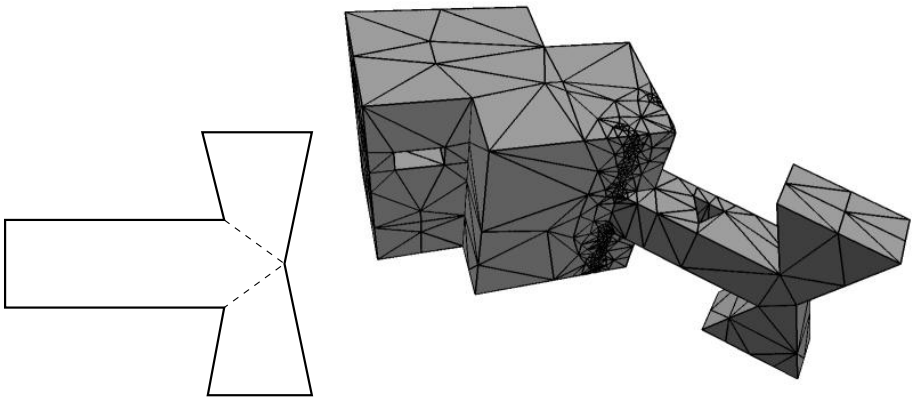
Recent results in [8, 9] show that it is possible modify a pointwise optimal mesh obtain a mesh conforming to the boundary with bounded classical aspect ratio; moreover, $\mathrm{lfs}_0(x)$ only changes by a constant factor. Arguments similar those used in two dimensions can then be used to establish size optimality of the resulting three dimensional triangulations.

6 Comments on Implementation

The two dimensional algorithm is very easy to implement once the fundamental data structures to represent a triangulation and the segments set have been developed. We implemented the algorithm in [6, Chapter 9] to associate points queued for addition to the mesh with triangles. The only time a segment is split and there is no triangle to assign the mid point to occurs in Step 7. In this step the two end points of a segment are present in the mesh but the segment does not appear. In this situation we have just inserted the second end point in the mesh, so have a triangle containing it. To locate a triangle containing the mid point we implement a geometric search along the segment starting from a triangle containing the end point. In order to accommodate input with small angles we implemented a version of the technique suggested by Ruppert [17]. Essentially the input is "groomed" so that segments forming a small angle form isosceles triangles in the mesh, and these isosceles triangles are never declared to be skinny. Similar approaches were considered by Shewchuk in [21].

The 2D code also accommodates inputs with small angles. We implemented a version of the technique suggested by Ruppert [17]. Essentially the input is "groomed" so that segments forming a small angle form isosceles triangles in the mesh, and these isosceles triangles are never declared to be skinny. Similar approaches were considered by Shewchuk in [21]. We point out that all known heuristics for handling small angle do not have optimality results associated with them. One only proves that the algorithm terminates, it generates a finite, possibly very large, mesh.

For both the two and three dimensional codes precise insphere tests were used to determine if points encroached upon simplices. These functions use the variable precision floating



(a) A non-convex face sub-dived into convex faces.

(b) Spurious facets cause anomalous refinement.

Figure 15: Triangulating non-convex faces may produce facets which intersect other features.

point arithmetic proposed in [19] and were automatically generated from "straight line code" using the compiler developed in [15]. When generating the two dimensional Delaunay meshes of an input face embedded in three dimensions the three dimensional insphere test was used to determine encroachment. This was done so that the two and three dimensional components of the code would always use the same function to determine encroachment. In particular, we did not map the faces into the the (x, y) plane and back, since the associated roundoff errors may cause the two and three dimensional routines to have a different view of degenerate data.

For our three dimensional code, when a point p was inserted into a mesh of a polytope F for the first time (the mesh of dimension $dim(F) = CD(p)$ containing p) it was inserted (`Force`'d) into the meshes of all polytopes containing F. It is easy to verify that this is consistent with the work order in Definition 4.1. As p is inserted into each mesh the simplices having p as a vertex are carried forward. This provides a convenient way to implement Steps 7 of the two dimensional algorithm and Step 8 of the three dimensional algorithm where the simplices containing the point being inserted are required.

Our incremental algorithms were always initialized with a large bounding simplex containing the input data. This avoids the need for code to add a point exterior to a triangulation. However, triangulations of the input faces then contain triangles that are not required to present in the three dimensional mesh. This was accommodated by declaring that triangles having a vertex on the bounding triangle are not facets. Moreover, two or three dimensional simplices having a vertex on the bounding simplex are never declared to be skinny. This strategy suffices provided all input faces are convex, since in this situation all trian-

gles exterior to the triangulated face contain a bounding vertex (and correspond to infinite triangles). If non-convex faces are input to the algorithm then "finite" triangles not in the face should not be identified as facets which always appear in the three dimensional mesh. Otherwise, these spurious triangles may intersect other input faces and this can result in anomalous refinement as shown in Figure 15. Currently we split non-convex input faces into a union of convex faces to avoid this problem; however, other alternatives could easily be implemented.

References

[1] M. Bern, D. Eppstein, and J. R. Gilbert. Provably good mesh generation. In *31th Annual Symposium on Foundations of Computer Science*, pages 231–241, Oct. 1990.

[2] T. M. Chan, J. Snoeyink, and C.-K. Yap. Primal dividing and dual pruning: output-sensitive construction of four-dimensional polytopes and three-dimensional Voronoi diagrams. *Discrete Comput. Geom.*, 18(4):433–454, 1997.

[3] L. Chew. Guaranteed-quality triangular meshes. CS 89-983, Cornell, 1989.

[4] L. P. Chew. Guaranteed–quality Delaunay meshing in 3d. *preprint*, 1997.

[5] D. Chhen-Steiner, E. C. de Verdiere, and M. Yvinec. Conforming delaunay triangulations in 3D. CS 4345, INRIA, 2001.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry.* Springer, 2000.

[7] H. Edelsbrunner. *Geometry and Topology for Mesh Generation.* Cambridge, 2000.

[8] H. Edelsbrunner, X. Li, Miller, G. Stathopoulos, D. A. Talmor, S. Teng, A. Ungor, and N. J. Walkington. Smoothing and cleaning up slivers. In *ACM Symposium on Theory of Computing*, pages 273–277, 2000.

[9] X.-Y. Li and S.-H. Teng. Generating well-shaped delaunay meshed in 3d. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 28–37. ACM Press, 2001.

[10] G. L. Miller, D. Talmor, S. Teng, and N. J. Walkington. A Delaunay based numerical method for three dimensions: generation, formulation and partition. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing.* ACM Press, 1995.

[11] G. L. Miller, D. Talmor, S. Teng, and N. J. Walkington. On the radius–edge condition in the control volume method. *SIAM J. Numer. Anal.*, 36(6):1690–1708, 1998.

[12] G. L. Miller, D. Talmor, S. Teng, N. J. Walkington, and H. Wang. Control volume meshes using sphere packing: Generation, refinement and coarsening. In *5th International Meshing Round Table, '96*, pages 47–62. Sandia National Laboratories, 1996.

[13] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in three dimensions. In *Proceedings of the ACM Computational Geometry Conference*, pages 212–221. ACM Press, 1992. Also appeared as Cornell C.S. TR 92-1267.

[14] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in higher dimensions. *SIAM J. Comput.*, 29(4):1334–1370 (electronic), 2000.

[15] A. Nanevski, G. Blelloch, and R. Harper. Automatic generation of staged geometric predicates. In *International Conference on Functional Programming*, pages 217–228, Florence, Italy, September 2001.

[16] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (Austin, TX, 1993)*, pages 83–92, New York, 1993. ACM.

[17] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995. Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (Austin, TX, 1993).

[18] J. R. Shewchuck. Triangle: A two-dimensional quality mesh generator and Delaunay triangulator. See: `http://www.cs.cmu.edu/~quake/triangle.html`, 1995.

[19] J. R. Shewchuk. *Delaunay Refinement Mesh Generation.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1997. Available as Technical Report CMU-CS-97-137.

[20] J. R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry (Minneapolis, Minnesota)*, pages 86–95. ACM, June 1998.

[21] J. R. Shewchuk. Mesh generation for domains with small angles. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry (Hong Kong, 2000)*, pages 1–10 (electronic), New York, 2000. ACM.