

# Delaunay Refinement Algorithms for Estimating Local Feature Size in 2D and 3D

Alexander Rand\* and Noel J. Walkington\*

July 14, 2009

## Abstract

We present Delaunay refinement algorithms for estimating local feature on the input vertices of a 2D piecewise linear complex and on the input vertices and segments of a 3D piecewise linear complex. These algorithms are designed to eliminate the need for a local feature size oracle during quality mesh generation of domains containing acute input angles. In keeping with Ruppert's algorithm, encroachment in these algorithms can be determined through only local information in the current Delaunay triangulation. The algorithms are simple enough to be implemented and several examples are given.

## 1 Introduction

The prototypical Delaunay refinement algorithm given by Ruppert[1] is an elegant method for computing a quality, conforming Delaunay triangulation of a non-acute 2D piecewise-linear complex (PLC). Ruppert's analysis involves proving that the size of the mesh at a point  $x$  (which can be measured equivalently as the distance to the second nearest vertex to  $x$  or the circumradius of the triangle containing  $x$ ) is a good approximation of the local feature size at  $x$  up to a constant depending on the minimum angle threshold used by the algorithm. As observed by Pav and Walkington[2], this means that Ruppert's algorithm is not only a method for quality mesh generation but also is an algorithm for computing local feature size.

The extension of Ruppert's algorithm to allow 3D PLCs with acute angles between input features is a topic of active research. Initial solutions to this problem have relied on algorithms for computing conforming Delaunay tetrahedralization[3, 4] and thus require the local feature size of the PLC be

---

\*Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213 USA. Supported in part by National Science Foundation Grant DMS-0811029. This work was also supported by the NSF through the Center for Nonlinear Analysis.

†AMS Classification: 65D99, 68U99

‡Submitted IJCGA, October 2008.

given to the algorithm as input, rather than be implicitly computed by the algorithm as was done by Ruppert’s algorithm[5]. The subsequent algorithm of Pav and Walkington[2] (referred to as PW3D) removed this requirement.

This paper develops a Delaunay refinement algorithm for estimating the local feature size as needed for quality mesh generation. It features two important improvements over PW3D. First, encroachment operations in the new algorithm are local in the Delaunay tetrahedralization as opposed to the operations in PW3D which are local in the Euclidean distance. An operation which is “local in the Euclidean distance” requires a breadth-first-search through the tetrahedralization up to a prescribed Euclidean distance from a given simplex, while an operation which is “local in the Delaunay tetrahedralization” only involves the immediate Delaunay neighbors of the vertices of a given simplex. Ruppert’s algorithm is local in the Delaunay triangulation. Second, our algorithm has been implemented, unlike many algorithms for conforming Delaunay tetrahedralization including PW3D.

The two algorithms for 3D quality mesh generation that have been implemented rely on relaxing the notion of Delaunay tetrahedralization and alternative feature sizes. The first is the constrained Delaunay refinement algorithm of Si and Gartner[6, 7]. In this case, a constrained Delaunay tetrahedralization is computed and local feature size is estimated at the *input vertices*. This is then interpolated incrementally on the edges as the refinement progresses. The second is the algorithm of Cheng, Dey and Ramos[8, 9] with its recent improvements[10]. This algorithm creates weighted Delaunay meshes of smooth surfaces from only local information but involves a user supplied sizing parameter. Our algorithm has been implemented and used as part of a quality mesh generator[11]: this paper serves to provide a rigorous proof of the algorithm for estimating local feature size used in this software.

To highlight the types of estimates we seek, consider a simplified version of Ruppert’s algorithm: Algorithm 1 describes Ruppert’s algorithm in two dimensions without any quality requirement on the output triangles. A segment is considered encroached in Algorithm 1 if there is another vertex in its diametral ball.

---

**Algorithm 1** Ruppert’s Algorithm - conformity only

---

Create an initial Delaunay triangulation of the input vertices.  
Queue all encroached segments.  
**while** the queue of segments is nonempty **do**  
    Insert the midpoint of the front segment into the Delaunay triangulation.  
    Update the queue of encroached segments.  
**end while**

---

Upon termination of Algorithm 1, the local feature size at any input vertex can be approximated by a quantity which is local in the Delaunay triangulation. This is summarized in the following theorem.

**Theorem 1.** *Given any non-acute 2D PLC as input, Algorithm 1 terminates. Upon termination, for any input vertex  $q_0$ ,*

$$\frac{1}{2} \text{lfs}(q_0) \leq N(q_0) \leq \sqrt{2} \text{lfs}(q_0).$$

Here,  $\text{lfs}(\cdot)$  denotes the local feature size of the input PLC and  $N(\cdot)$  denotes the distance to the nearest neighbor in the resulting Delaunay triangulation. The definitions of these functions are given in Section 2. The lower bound in Theorem 1 follows from the standard analysis of Ruppert’s algorithm, but the upper bound is not a part of the usual theory. However, it is an estimate of this type which is needed in forming a protected region around acute input angles for conforming Delaunay refinement. Our work involves finding algorithms in 2D and 3D for which analogous upper bounds hold independent of the smallest angle in the input.

A generic Delaunay refinement algorithm which will be specified in the later sections is described in Section 3. Theorem 1 will be duplicated in Section 4 for an algorithm which allows acute input angles. These estimates are analogous to those needed in the full 3D algorithm which is stated and analyzed in Section 5. Finally, some examples of are given in Section 6.

## 2 Preliminaries

The typical input to a Delaunay refinement algorithm is a piecewise linear complex.

**Definition 1.** In two dimensions:

- A **2D piecewise linear complex** (PLC),  $\mathcal{C} = (\mathcal{P}, \mathcal{S})$ , is a pair of sets of input vertices  $\mathcal{P}$  and input segments  $\mathcal{S}$ , such that the endpoints of each segment of  $\mathcal{S}$  are contained in  $\mathcal{P}$  and the intersection of any two segments of  $\mathcal{S}$  is also contained in  $\mathcal{P}$ .
- A PLC  $\mathcal{C}' = (\mathcal{P}', \mathcal{S}')$  is a **refinement** of the PLC  $\mathcal{C} = (\mathcal{P}, \mathcal{S})$  if  $\mathcal{P} \subset \mathcal{P}'$  and each segment in  $\mathcal{S}$  is the union of segments in  $\mathcal{S}'$ .

**Definition 2.** In three dimensions:

- A **3D piecewise linear complex** (PLC),  $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$ , is a triple of sets of input vertices  $\mathcal{P}$ , input segments  $\mathcal{S}$ , and polygonal input faces  $\mathcal{F}$  such that the boundary of any feature or the intersection of any two features is the union of other lower-dimensional features in the complex.
- A PLC  $\mathcal{C}' = (\mathcal{P}', \mathcal{S}', \mathcal{F}')$  is a **refinement** of the PLC  $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$  if  $\mathcal{P} \subset \mathcal{P}'$  and each segment in  $\mathcal{S}$  is the union of segments in  $\mathcal{S}'$  and every face in  $\mathcal{F}$  is the union of faces in  $\mathcal{F}'$ .

When refining a PLC, certain simplices near the boundaries of input features have special importance in the analysis. These are defined below.

**Definition 3.** Consider a refinement  $(\mathcal{P}', \mathcal{S}', \mathcal{F}')$  [or  $(\mathcal{P}', \mathcal{S}')$ ] of an input PLC  $(\mathcal{P}, \mathcal{S}, \mathcal{F})$  [or  $(\mathcal{P}, \mathcal{S})$ ].

- An **end segment** is a segment in  $\mathcal{S}'$  for which at least one endpoint is an input vertex in  $\mathcal{P}$ . vertex lies on an input segment in  $\mathcal{S}$ .
- The **spindle** of a segment  $s$  in  $\mathcal{S}'$ , denoted  $\text{Spind}(s)$ , is the set containing
  - $s$  if  $s$  is not an end segment, or
  - $s$  and all end segments adjacent to  $s$  if  $s$  is an end segment.

For a simplex  $s$ ,  $R_s$  denotes its circumradius. For any vertex  $q$  inserted into the mesh by our refinement algorithms,  $r_q$  denotes the insertion radius of vertex  $q$ , i.e. the distance from  $q$  to its nearest neighbor in  $\mathcal{P}'$  when it is inserted into the Delaunay triangulation. Denote the diametral ball of the segment between  $a$  and  $b$  by  $B(\overline{ab})$  and the circumball of the triangle with vertices  $a$ ,  $b$ , and  $c$  by  $B(\overline{abc})$ .

An appropriate notion of feature size is essential in the analysis of Delaunay refinement algorithms. The standard definition of local feature size is given below as well as another related sizing function (called mesh feature size).

**Definition 4.** Let  $\mathcal{C}$  be a PLC with refinement  $\mathcal{C}'$  and let  $\mathcal{P}'$  be the vertex set of  $\mathcal{C}'$ .

- The  **$i$ -local feature size** at point  $x$  with respect to  $\mathcal{C}$ ,  $\text{lfs}_i(x, \mathcal{C})$  is the radius of the smallest closed ball centered at  $x$  which intersects two *disjoint* features of  $\mathcal{C}$  of dimension no greater than  $i$ .
- The  **$i$ -mesh feature size** at point  $x$  with respect to  $\mathcal{C}$ ,  $\text{mfs}_i(x, \mathcal{C})$  is the radius of the smallest closed ball centered at  $x$  which intersects two features of  $\mathcal{C}$  of dimension no greater than  $i$ .
- The **nearest neighbor function**,  $N(x, \mathcal{P}') := \text{lfs}_0(x, \mathcal{C}')$ , returns the distance from  $x$  to its second nearest neighbor in  $\mathcal{P}'$ .

The above definitions do not require any distinction between the input PLC and its refinement. However, we state the definitions in this way as local feature size functions will usually be evaluated with respect to some initial PLC while the nearest neighbor function will be analyzed on the intermediate or resulting triangulations. To simplify the notation in these cases, a few conventions will be followed.

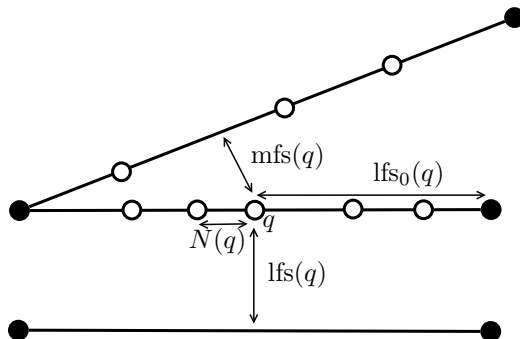


Figure 1: Example of sizing functions in Definition 4 for a 2D PLC. The black dots represent input vertices while the white dots represent vertices inserted during the refinement.

### Conventions

- If the PLC argument is omitted in the mesh or local feature size function, it is assumed to be the input complex, e.g.  $\text{lfs}_i(x) := \text{lfs}_i(x, \mathcal{C})$ .
- If the vertex set argument is omitted in the nearest neighbor function, it is assumed to be the vertex set of the current refined complex, e.g.  $N(x) := N(x, \mathcal{P}')$ .
- If the dimension argument is omitted in the mesh or local feature size function, it is assumed to be  $(d - 1)$ , e.g.  $\text{lfs}(x, \mathcal{C}) := \text{lfs}_{d-1}(x, \mathcal{C})$ .

Figure 1 depicts the feature size at the vertex of a mesh during a possible refinement. Note that the feature size is defined at all points in  $\mathbb{R}^d$ , not just vertices of the mesh. Each of these functions is Lipschitz (with constant 1). For a fixed PLC, local feature size is strictly positive while mesh feature size can equal zero.

If the argument supplied to any of the above feature size functions is a set of points, rather than a point, then the result is defined to be the infimum of the function over the set, i.e.

$$\text{lfs}_i(s, \mathcal{C}) := \inf_{x \in s} \text{lfs}_i(x, \mathcal{C}).$$

Often it will be important to show identical estimates on the local feature size of end segments and the mesh feature size of non-end segments. It is useful to refer to these two cases with the same notation.

**Definition 5.** The  $i$ -feature size of segment  $s$  is defined as follows.

$$\text{fs}_i(s) = \begin{cases} \text{lfs}_i(s) & \text{if } s \text{ is an end segment,} \\ \text{mfs}_i(s) & \text{if } s \text{ is a non-end segment.} \end{cases}$$

Given segment  $s$  in  $\mathcal{S}'$ , point  $x$  is called an  **$i$ -feature size witness** for  $s$  if  $x$  is contained in a feature of  $\mathcal{C}$  of dimension at most  $i$  which is disjoint from  $s$ . Given simplex  $s$  in  $\mathcal{C}'$ , point  $x$  is called a **local feature size witness** for  $s$  if  $x$  is contained in a feature of  $\mathcal{C}$  which is disjoint from another feature of  $\mathcal{C}$  containing  $s$ . Simplex  $s'$  is called a  $i$ -feature size witness for simplex  $s$  if every point of  $s'$  is an  $i$ -feature size witness for  $s$ .

The definition of feature size is closely related to that of local gap size used by Cheng and Poon[12]. The notion of  $i$ -feature size witness will be used by recognizing that if  $x$  is an  $i$ -feature size witness of segment  $s$ , then

$$\text{fs}_i(s) \leq \text{dist}(x, s).$$

Finally, the following form of the Delaunay property is used often throughout the analysis.

**Proposition 1.** *[Delaunay Property] Let  $\mathcal{P}$  be a finite subset of  $\mathbb{R}^d$ . Let  $B$  be a ball with vertex  $q \in \mathcal{P}$  on the boundary of  $B$ . If  $\mathcal{P} \cap B \neq \emptyset$ , then  $q$  has a Delaunay neighbor  $B$ .*

### 3 Generic Delaunay Refinement Algorithm

Each of the Delaunay refinement algorithms considered will be in the form of Algorithm 2.

---

**Algorithm 2** Delaunay Refinement

---

Create an initial Delaunay triangulation.  
 Queue all unacceptable simplices.  
**while** the queue of simplices is nonempty **do**  
   **if** it is safe to split the front simplex **then**  
     Take an action based on the front simplex.  
     Update the queue of unacceptable simplices.  
   **else**  
     Remove the front simplex from the queue.  
   **end if**  
**end while**

---

To specify a concrete algorithm from Algorithm 2, it necessary to describe the following statements.

Action	Where should a vertex (a Steiner point) be inserted to “split” a simplex? Should other (usually lower dimensional) features be queued for splitting?
Priority	In what order should be queue be processed?
Unacceptability	Which simplices are unacceptable?
Safety	Which simplices are safe to split?

First, Algorithm 3 is a restatement of Algorithm 1 (Ruppert’s algorithm with a  $0^\circ$  minimum angle threshold) as a specialization of the general algorithm.

---

**Algorithm 3** Ruppert’s Algorithm - conformity only

---

Action	Insert the midpoint of the segment.
Priority	Process segments in any order.
Unacceptability	A segment is unacceptable if it has a nonempty diametral disk.
Safety	Any simplex is safe to split.

---

It is important to note that each of these specifications for the algorithm can be computed locally in the Delaunay triangulation of the current vertex set. This is an essential property of Delaunay refinement algorithms. In our view, any algorithm which matches the form of Algorithm 2 and can be updated based on the local Delaunay triangulation is a Delaunay refinement algorithm and any algorithm that doesn’t fit these to requirements is not.

Some of these specifications for Ruppert’s algorithm are so simple that they can be easily overlooked. However, it is important to generalize Delaunay refinement algorithms in each of the four ways considered above for different purposes. Here is a brief description of how this has been done in the literature.

**Action** The general action for removing an unacceptable simplex involves inserting a vertex inside the circumball of the simplex to ensure that it no longer exists in the Delaunay triangulation. Chew’s first Delaunay refinement algorithm[13] used the insertion of circumcenters to remove poor quality triangles from the mesh. The circumcenter is a natural choice since this gives the furthest guaranteed distance between the new vertex and any others in the Delaunay triangulation based only on the existence of the original simplex. Ruppert’s algorithm added the idea of yielding to lower dimensional features of the input. Off-center vertices and general selection regions have also been studied[14, 15, 16] using the same yielding procedure as Ruppert’s algorithm. An example of a different action taken by the algorithm can be seen with Chew’s second Delaunay refinement algorithm[17]. This method maintains a constrained Delaunay triangulation, involves a different yielding procedure, and removes vertices from the mesh following certain midpoint insertions. The algorithm of Miller, Hudson, and Phillips includes a yielding procedure in which circumcenters yield to input vertices which have not been inserted into the mesh[18].

**Priority** The priority queue for most Delaunay refinement algorithms involves prioritizing lower dimensional simplices before higher dimensional ones[1, 19]. For time efficient algorithms, this priority queue must be modified[20, 18, 21], typically requiring simplices queued for quality to be processed before those

queued based on encroachment. Prioritizing queued simplices of equal dimension (often by circumradius) has also been used in some algorithms[20, 22].

**Unacceptability** There are typically two types of unacceptability criteria: encroachment criteria which ensure that the refined triangulation conforms to the input PLC, and quality requirements which are desirable of the resulting simplices. For the encroachment criteria, the most common approach involves asking if a simplex has a nonempty circumball. This is useful since any simplex with a empty circumball must appear in the Delaunay triangulation. Methods which utilize constrained Delaunay triangulations often relax this requirement and consider protecting a smaller lens around each segment or ignore an explicit encroachment criteria all together[17, 23].

The quality criteria is usually based on the radius-edge ratio (or the closely related Voronoi quality) of the triangulation. Quality also may be specified via a user defined sizing parameter. Most Delaunay refinement algorithms allow sizing functions of this type, but a few require this type of criteria explicitly in the proofs of correctness[13, 9].

**Safety** Meshing non-acute domains does not typically require any check that a simplex is safe to split. When handling domains with small angles, typical approaches involve not splitting triangles based on quality if they are near a small input angle in some sense[24, 25]. In 3D, the Tetgen code[6, 7] relies on a similar principle for determining when to stop refining near small input angles.

## 4 Estimating Feature Size in 2D

We develop an algorithm for estimating the local feature size of a mesh at input vertices of a 2D PLC. Estimates of this time are often useful in ensuring the termination of quality Delaunay refinement algorithms in the presence of acute angles between adjacent input segments. While there are a number of effective algorithms for quality mesh generation in 2D[24, 25], this algorithm is developed as a natural predecessor to the 3D version given in Section 5.

Local feature size is estimated at each input vertex in terms of the distance to its nearest Delaunay neighbor in the resulting triangulation. The algorithm is very similar to Ruppert’s algorithm with two key differences: triangles are not split based on radius-edge quality and certain segments are not split to prevent infinite encroachment sequences near acute angles.

The algorithm for estimating feature size is divided into two steps as given in Algorithm 4. These steps are labeled according to the highest dimensional features in the input complex refined: Step 0 only depends upon the input vertices while Step 1a involves input vertices and segments.

(Step 0) Compute the Delaunay triangulation of the input vertices.
--

Step 0 involves computing the Delaunay triangulation of the set of input vertices. This yields a simple estimate on  $lfs_0$  at each of the input vertices.

---

**Algorithm 4** Estimate Feature Size 2D

---

(Step 0) Compute the Delaunay triangulation of the input vertices.  
(Step 1a) Estimate lfs at all input vertices via Delaunay refinement.

---

**Lemma 2.** *Upon the termination of Step 0, for each vertex  $q_0$  in the input PLC,  $N(q_0) = \text{lfs}_0(q_0)$ .*

(Step 1a) Estimate lfs at all input vertices via Delaunay refinement.

Step 1a of Algorithm 4 is a Delaunay refinement algorithm specified by the four rules given in Algorithm 5. It is important to recognize that adjacent segments have *not* been split to equal lengths as a preprocess to this algorithm. To ensure termination, a segment  $s$  is not allowed to split if the encroaching vertex is on a segment adjacent to  $s$  and the resulting segments are shorter than the shortest segment in  $\text{Spind}(s)$ . This criteria is reflected in the unacceptability rule.

---

**Algorithm 5** Estimate Feature Size 2D - Step 1a

---

Action	Insert the circumcenter of a segment.
Priority	Simplices (only segments in this case) may be processed in any order.
Unacceptability	A segment $s$ is unacceptable if it has an endpoint $q$ with a Delaunay neighbor $p$ inside the diametral disk of $s$ and either $p$ is a 1-feature size witness for $s$ or $s$ is more than twice the length of the shortest segment in $\text{Spind}(s)$ .
Safety	It is safe to split any simplex.

First, it is shown that the algorithm terminates and that the distance to the nearest neighbor provides an appropriate upper bound on local feature size in the resulting mesh. This estimate is similar to those shown in Ruppert's analysis.

**Theorem 3.** *Algorithm 4 terminates. For any input vertex  $q_0$ ,*

$$\frac{1}{2} \text{lfs}(q_0) \leq N(q_0, \mathcal{P}')$$

*holds throughout the algorithm.*

*Proof.* Let  $q_0 \in \mathcal{P}$  be any input vertex. Initially,  $N(q_0) = \text{lfs}_0(q_0) \geq \text{lfs}(q_0)$  so the base case holds. Suppose a vertex  $q$  is inserted as the midpoint of segment  $s$  and  $q$  is the closest neighbor to an input vertex  $q_0$ . If  $s$  is disjoint from  $q_0$ , then  $\text{lfs}(q_0) \leq |q_0 - q|$ . If  $s$  is incident to  $q_0$ , then (by the unacceptability rule) the vertex encroaching  $s$ , denoted  $q'$ , must be on a segment which is disjoint from  $q_0$ . Thus,  $\text{lfs}(q_0) \leq |q_0 - q'| \leq 2|q_0 - q|$ . This bound ensures the termination of the algorithm.  $\square$

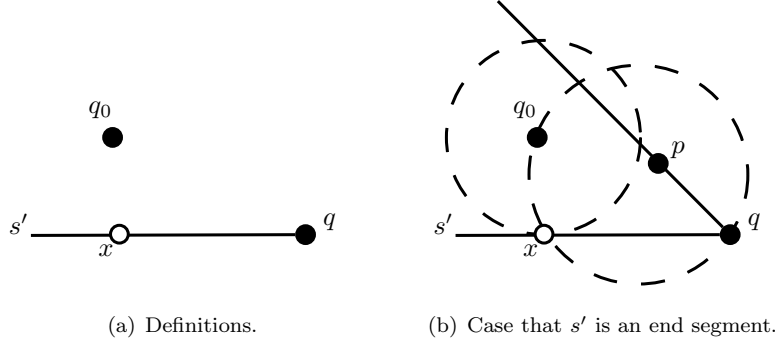


Figure 2: Diagrams for the proof of Theorem 4.

Next, it can be shown that the distance from an input vertex to its nearest neighbor in the resulting triangulation also provides a lower bound on the local feature size.

**Theorem 4.** *Upon the termination of Algorithm 4,*

$$N(q_0, \mathcal{P}') \leq \sqrt{2} \text{lfs}(q_0)$$

for any input vertex  $q_0 \in \mathcal{P}$ .

*Proof.* If  $N(q_0, \mathcal{P}) = \text{lfs}(q_0)$  (i.e. the local feature size at  $q_0$  is realized by an input vertex), then the statement follows by

$$N(q_0, \mathcal{P}') \leq N(q_0, \mathcal{P}) = \text{lfs}(q_0).$$

Otherwise,  $\text{lfs}(q_0) = \text{dist}(q_0, s)$  for some segment  $s \in \mathcal{S}$  disjoint from  $q_0$  (i.e. the local feature size of  $q_0$  is realized by a segment  $s$ ). Let  $x$  be the nearest point on segment  $s$  to  $q_0$ . Let  $s' \in \mathcal{S}'$  be a subsegment of  $s$  containing  $x$  and let  $q$  be the nearest endpoint of  $s'$  to  $q_0$ . This situation is depicted in Figure 2(a).

Now, suppose that  $N(q_0, \mathcal{P}') > \sqrt{2} \text{lfs}(q_0)$ . Then the following inequalities hold.

$$\begin{aligned} 2 \text{lfs}(q_0)^2 &< N(q_0, \mathcal{P}')^2 \\ &< |q_0 - q|^2 \\ &= \text{lfs}(q_0)^2 + |x - q|^2. \end{aligned}$$

Conclude that  $|x - q_0| = \text{lfs}(q_0) \leq |x - q|$ . This inequality implies that  $q_0$  lies in the diametral disk of  $s'$ . So either  $s'$  is unacceptable or  $q$  is an input vertex and there is a vertex  $p \in \mathcal{P}'$  in  $B(\overline{q_0q}) \setminus B(q_0, |x - q|)$  which lies on a segment adjacent to  $s'$  since  $q_0$  and  $q$  cannot be Delaunay neighbors. The ball  $B(q_0, |x - q|)$  must be empty by the assumption on the local feature size of  $q_0$ . Thus vertex  $p \in B(\overline{q_0q}) \setminus B(q_0, |x - q|)$  and it follows that  $|p - q| \leq |x - q| \leq \frac{|s'|}{2}$  as seen in

Figure 2(b). Thus  $|s'| \geq |\overline{pq}|$  which is in the spindle of  $s'$ . So  $s'$  is unacceptable. Since upon termination there are no unacceptable segments, the desired bound must hold.  $\square$

The constants in Theorem 3 and Theorem 4 are both sharp and independent of the smallest input angle. Note that the inequality in Theorem 4 is identical to the upper bound in Theorem 1 for Ruppert's algorithm in the non-acute case: acute input angles slightly complicate the algorithm but do not weaken the result.

## 5 Estimating Feature Size in 3D

The idea of the previous chapter can be extended to 3D Delaunay refinement. In this case, the goal is to estimate local feature size and 1-feature size on all segments (the  $(d - 2)$ -dimensional features) of the input complex. While the distance from an input vertex to its nearest neighbor was used to estimate feature size in 2D, the 3D analogy uses the length of segments in the refined PLC.

Algorithm 6 will yield the desired feature size estimates in terms of segment lengths. Step 1b and Step 2b are specific Delaunay refinement algorithms which will be described later. Each of the other steps is a simple procedure which occurs in a single pass over the Delaunay triangulation.

---

### Algorithm 6 Estimate Feature Size 3D

---

- (Step 0) Compute the Delaunay tetrahedralization of the input vertices.
  - (Step 1a) Split adjacent segments at equal lengths based on 0-local feature size.
  - (Step 1b) Estimate  $fs_1$  on all segments via Delaunay refinement.
  - (Step 2a) Split segments to improve the 1-feature size estimate.
  - (Step 2b) Estimate  $lfs$  on all segments via Delaunay refinement.
- 

The following two theorems demonstrate that in the refined PLC the length of each segment is a good estimate for the local feature size or 1-feature size of that segment. The lower bound on segment lengths given in Theorem 5 involves standard techniques used in the analysis of Ruppert's algorithm. The upper bound on segment lengths given in Theorem 6 requires a more in depth analysis than that of previous algorithms.

**Theorem 5.** *Throughout Algorithm 6, all segments  $s \in \mathcal{S}'$  satisfy*

$$\min \left( \frac{1}{16} fs_1(s), \frac{1}{4} lfs(s) \right) \leq |s|.$$

**Theorem 6.** *Following the termination of Algorithm 6, all segments  $s \in \mathcal{S}'$  satisfy*

$$|s| \leq \min \left( \sqrt{2} fs_1(s), \frac{5}{3} lfs(s) \right).$$

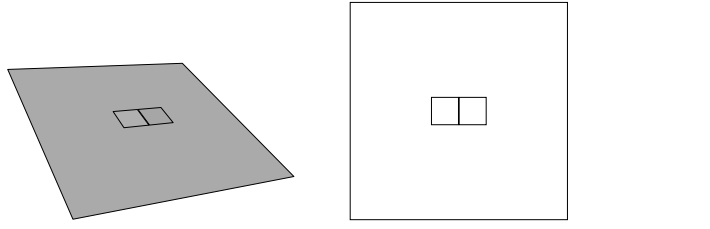


Figure 3: This illustrative example consists of 3 faces: one large square which is slightly below two smaller squares which are side by side.

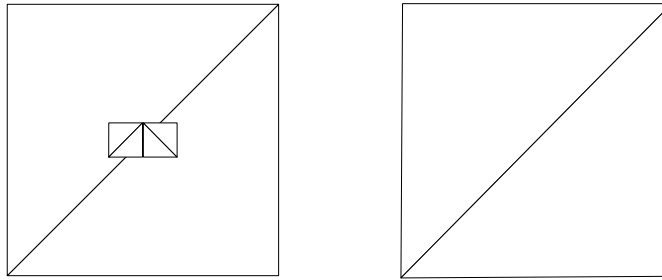


Figure 4: (Left) Example mesh following Step 0. (Right) Enlarged mesh of one of the smaller squares.

In order to show these two theorems, output conditions on the PLC are determined following each step. Step 2b will yield a mesh satisfying precisely these conditions in the theorems.

We illustrate this algorithm by considering the results of each step on a simple PLC. The example consists of three squares (contained inside a sufficiently large bounding box): one large square which is slightly below two coplanar, disjoint squares as seen in Figure 3. Observe that the small feature size between the sides of the two small squares will be realized in Step 1b, while the feature size between the small planes and the large plane will be realized in Step 2b.

(Step 0) Compute the Delaunay tetrahedralization of the input vertices.

Computing the Delaunay tetrahedralization of the input vertices is a common first step in many Delaunay refinement algorithms. In our running example, this simply leads to the Delaunay triangulation of each of the squares as seen in Figure 4.

**Lemma 7.** *Upon the termination of Step 0, the following inequalities hold at all input vertices of the mesh,*

$$\text{lfs}_2(q_0) \leq \text{lfs}_1(q_0) \leq \text{lfs}_0(q_0) = N(q_0, \mathcal{P}').$$

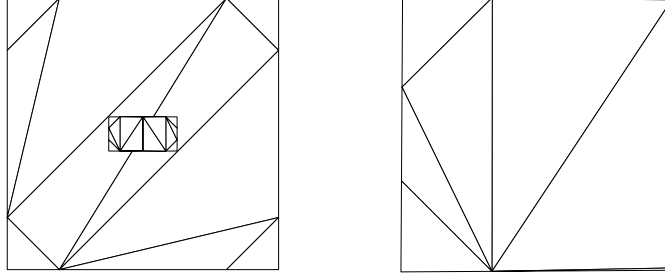


Figure 5: (Left) Example mesh following Step 1a. (Right) Enlarged mesh of one of the smaller squares.

(Step 1a) Split adjacent segments at equal lengths based on 0-local feature size.

This step consists of a single pass of each of the input vertices. For each input vertex  $q_0$ , all segments containing this vertex are split at a distance of  $\frac{N(q_0, \mathcal{P}')}{3}$  away from  $q_0$ . The result of this step on the running example can be seen in Figure 5. Notice that small faces are split at a small distance on one side due to the close proximity of their corners to each other.

After completing Step 1a, a number of properties hold which are summarized in the next proposition.

**Lemma 8.** *Upon the termination of Step 1a, the following hold.*

- (I)  $N(q_0, \mathcal{P}') = \frac{1}{3} \text{lfs}_0(q_0)$  holds for all input vertices  $q_0 \in \mathcal{P}$ .
- (II) If  $s_n$  is a non-end segment and  $s_e$  is an adjacent end segment,  $|s_n| \geq |s_e|$ .
- (III) If  $s_e$  and  $s'_e$  are end segments and  $s'_e \notin \text{Spind}(s_e)$ , then

$$\text{dist}(s_e, s'_e) \geq \max(|s_e|, |s'_e|).$$

Property III is particularly important. As segments are refined further, this property continues to hold ensuring that the spindles of end segments corresponding to different input vertices are sufficiently far apart.

(Step 1b) Estimate  $\text{fs}_1$  on all segments via Delaunay refinement.

The goal of this stage is to bound the length of each segment by the 1-feature size of that segment. This occurs via a Delaunay refinement algorithm specified in Algorithm 7.

By the specification given, checking if a simplex is unacceptable requires that only Delaunay neighbors of the endpoints of the segment in question need

---

**Algorithm 7** Estimate Feature Size 3D - Step 1b

---

Action	Insert the midpoint of a segment.
Priority	Longer segments are processed first.
Unacceptability	Segment $s$ is unacceptable if there is an endpoint $q$ of a segment in $\text{Spind}(s)$ with Delaunay neighbor $p$ such that $p$ is a 1-feature size witness for $q$ and $ q - p  <  s $ .
Safety	It is safe to split any segment.

---

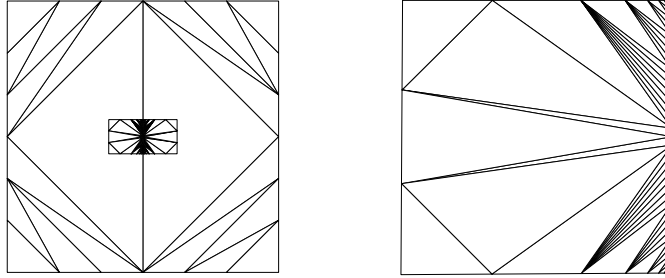


Figure 6: (Left) Example mesh following Step 1b. (Right) Enlarged mesh of one of the smaller squares.

to be queried. This is an important property of Ruppert’s algorithm that has been carefully maintain.

Figure 6 shows the running example following Step 1b. Notice that the main effect is that the nearby edges of the two small squares refine to realize the feature size. The other segments are only split a few times.

First, we show that the algorithm described terminates and that the length of each segment is bounded below by its feature size. This argument uses the same arguments as the “usual” proofs of termination and grading of typical Delaunay refinement algorithms.

**Lemma 9.** *Throughout Step 1b, any segment  $s$  in the refinement satisfies*

$$\frac{1}{4} \text{fs}_1(s) \leq |s|.$$

*Proof.* Inductively, we show that the lower bound holds at all segments throughout this step.

Base Case. Following Step 1a, any end segment,  $s_e$ , containing input vertex  $q_0$  has length

$$|s_e| = \frac{1}{3} \text{lfs}_0(q_0).$$

The definition of the 1-feature size implies that

$$\text{lfs}_0(q_0) \geq \text{lfs}_1(q_0) \geq \text{lfs}_1(s_e) = \text{fs}_1(s_e).$$

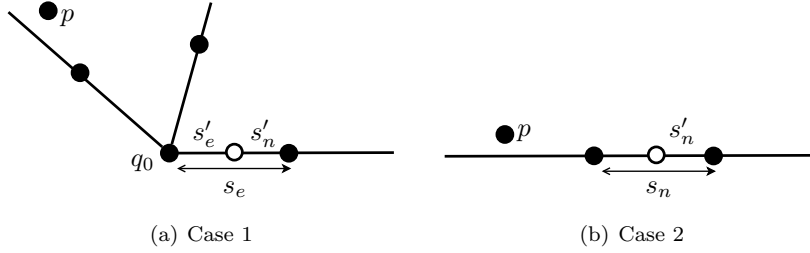


Figure 7: Two Cases in Lemma 9.

Thus  $|s_e| \geq \frac{1}{3} \text{fs}_1(s_e)$ .

For any initial non-end segment,  $s_n$ , there is an adjacent end segment  $s_e$  such that  $|s_n| \geq |s_e|$ . Since  $s_e$  contains an input vertex (which is a 1-feature size witness for  $s_n$ ), it follows that

$$|s_n| \geq |s_e| \geq \text{fs}_1(s_n).$$

Thus, the lower bound on segment lengths holds initially.

The inductive step is shown in two cases corresponding to the insertion of end segment midpoints and the insertion of non-end segment midpoints. These cases are depicted in Figure 7.

Case 1. Consider an end segment  $s_e$  from  $q_0$  to  $q'$  which is split forming a new end segment  $s'_e$  and a non-end segment  $s'_n$ . Then there exists a vertex  $p$  on an input feature disjoint from  $s_e$  which is of distance at most  $|s_e|$  from some adjacent end segment to  $s_e$ .

$$\text{fs}_1(s'_e) \leq \text{dist}(s'_e, p) \leq \text{dist}(q_0, p) \leq |s_e| + |s_e| = 4|s'_e|.$$

For the non-end segment  $s'_n$ ,  $q_0$  is a 1-feature size witness, and thus

$$\text{fs}_1(s'_n) \leq \text{dist}(s'_n, q_0) = |s'_n|.$$

Case 2. Consider non-end segment  $s_n$  which is split. This means that there is a 1-feature size witness  $p$  such that  $\text{dist}(s_n, p) \leq |s_n|$ . Then for either of the new end segments created, denoted  $s'_n$ ,

$$\text{fs}_1(s'_n) \leq \text{dist}(s'_n, p) \leq |s'_n| + |s_n| = 3|s'_n|.$$

We conclude that  $\frac{1}{4} \text{fs}_1(s) \leq |s|$  for all segments created during Step 1b. This 1-lower bound on feature size of all segments ensures termination of the algorithm.  $\square$

Next we seek to bound the length of each segment from *above* in terms of the feature size. In the previous lemma, the ordering of the queue of segments is irrelevant. In order to get the upper bound, an arbitrary order does not

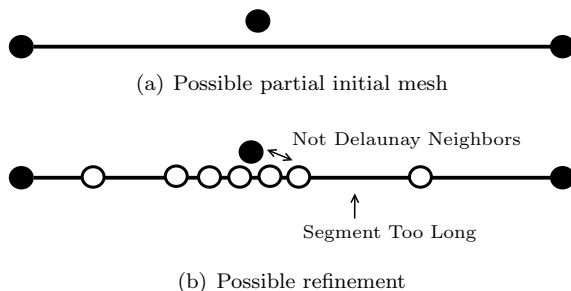


Figure 8: Lemma 10 does not hold without specifying a refinement order.

work. To see this, consider a mesh including a portion similar to Figure 8(a). If segments to the left are refined first, a situation similar to Figure 8(b) could arise. Then, there is a segment on the right side which is longer than its distance to the input vertex which is not on the segment. However, this segment may not “see” this nearby vertex on its Delaunay cavity. Note: this requires another vertex to block the long segment from seeing the nearby disjoint vertex, but this vertex could be far away and thus not causing the long segment to split.

Prioritizing the queue by segment length prevents this problem. This requirement, coupled with some geometric facts found in the appendix, lead to the desired bound on segments following Step 1b.

**Lemma 10.** *Upon the termination of Step 1b, all segments satisfy*

$$|s| \leq \sqrt{2} fs_1(s).$$

*Proof.* The two key properties below will be verified throughout the algorithm and will lead to the desired result.

**Inductive Hypothesis:** If segment  $s$  is not queued and  $|s| > \sqrt{2} fs_1(s)$ , then the following two statements hold.

1. If  $q_0 \in \mathcal{P}$ ,  $q_0 \notin s$  and  $q$  is an endpoint of  $s$ , then  $|q - q_0| \geq |s|$ .
2. If  $\bar{s}$  is a 1-feature size witness for  $s$  such that  $\text{dist}(s, \bar{s}) < \frac{|s|}{\sqrt{2}}$ ,  $q$  is an endpoint of  $s$ , and  $\bar{q}$  is an endpoint of  $\bar{s}$ , then  $|q - \bar{q}| \geq |s|$ .

**Split Size Property.** Any longest segment  $s$  such that  $|s| > \sqrt{2} fs_1(s)$  is on the queue.

By showing the following three claims, we are able to conclude the lemma.

- The split size property  $\Rightarrow |s| \leq \sqrt{2} fs_1(s)$  upon termination.
- The inductive hypothesis  $\Rightarrow$  the split size property.

- The inductive hypothesis holds.

**The split size property**  $\Rightarrow |s| \leq \sqrt{2} \text{fs}_1(s)$  **upon termination.** Upon termination of the algorithm, the queue is empty. This  $|s| \leq \sqrt{2} \text{fs}_1(s)$  holds for all segments: if any segment failed this bound, then the split size property would imply that the queue is not empty.

**The inductive hypothesis**  $\Rightarrow$  **the split size property.** Let  $s$  be a segment such that  $|s| > \sqrt{2} \text{fs}_1(s)$  and  $s$  is not on the queue. Then by the first property of the inductive hypothesis, the feature size of  $s$  is not realized by an input vertex. Thus, there exists a segment  $\bar{s}$  which is a 1-feature size witness for  $s$  and  $\text{dist}(s, \bar{s}) = \text{fs}_1(s)$ . By Proposition 2,  $\bar{s}$  is longer than  $s$  (otherwise the segments would have endpoints that are nearby, which violates the inductive hypothesis). The inductive hypothesis and Proposition 3 imply that  $s$  must be a 1-feature size witness for  $\bar{s}$  (since otherwise  $\bar{s}$  must be an end segment adjacent to the input feature containing non-end segment  $s$ ). Combining these facts yields

$$|\bar{s}| > |s| > \sqrt{2} \text{fs}_1(s) \geq \sqrt{2} \text{fs}_1(\bar{s}).$$

We conclude that  $s$  is not the longest segment failing the feature size bound.

**The inductive hypothesis holds.** The technical details of the proof lie in verifying the inductive hypothesis.

Base Case. First consider initial end segments. Let  $q_0$  be an input vertex contained in end segment  $s_e$ . Proposition 8 ensures that for all other vertices  $\bar{q}$  at the end of Step 1a which are not on an end segment adjacent to  $s_e$ ,  $|q_0 - \bar{q}| \geq 2|s_e|$ . Thus, the inductive hypothesis holds for all end segments. Next, consider non-end segments. Let  $s_n$  be a non-end segment between end segments  $s_e$  and  $s'_e$ . Initially, any vertex which is not an endpoint of  $s_n$  is a 1-feature size witness for  $s_n$ . If there is another vertex which is of distance less than  $|s_n|$  to an endpoint of  $s_n$ , then  $s_n$  must be queued by some 1-feature size witness which is a Delaunay neighbor to an endpoint of  $s_n$ .

Next, we proceed to the inductive step. The inductive hypothesis must be checked on all segments. There are two types of segments for which this must be verified: segments that existed before the most recent vertex insertion and segments that were formed by this insertion.

Case 1. Consider any *newly* formed segment  $s$  and suppose  $s$  violates the inductive hypothesis. If the first criterion of the inductive hypothesis fails, let  $\bar{q}$  denote the input vertex such that  $|\bar{q} - q| < |s|$  for some endpoint  $q$  of  $s$ . Otherwise, the second criterion fails meaning that  $|s| > \sqrt{2} \text{fs}_1(s)$ ,  $s$  is not on the queue, and (by Proposition 2) there is a vertex  $\bar{q}$  and endpoint  $q$  of  $s$  such that  $|q - \bar{q}| < |s|$  and  $\bar{q}$  is a feature size witness for  $s$ . In either case,  $\bar{q}$  is a feature size witness for  $s$  and the distance between  $q$  and  $\bar{q}$  is less than  $|s|$ . Since  $s$  is not queued, this means that  $q$  and  $\bar{q}$  cannot be Delaunay neighbors.

Now by the Delaunay property there is some vertex  $p$  in the diametral ball between  $q$  and  $\bar{q}$  which is a Delaunay neighbor of  $q$ . Again,  $p$  cannot be a 1-feature size witness for  $s$ , as this would cause  $s$  to be queued.

If  $s$  is an end segment, notice that no such  $p$  can exist. Every non-end segment adjacent to a segment in the spindle of  $s$  has length of  $|s|$  or  $2|s|$ , and thus there is no vertex  $p$  on one of these segments at a distance of less than  $|s|$  which is not an endpoint of some segment in  $\text{Spind}(s)$ .

If  $s$  is a non-end segment, consider the segment  $\hat{s}$  which was split forming  $s$ . If  $\bar{q}$  is a 1-feature size witness for  $\hat{s}$ , then  $\hat{s}$  fails the desired feature size bound (see Figure 9(a)). However vertex  $p$ , which was inserted before  $q$ , was inserted as the midpoint of a segment of length  $2|p-q| < 2|s|$ . This violates the split size property (and thus the inductive hypothesis). Otherwise  $\hat{s}$  is an end segment and  $\bar{q}$  lies on an adjacent input segment (see Figure 9(b)). Let  $q_0$  be the input vertex which is an endpoint of  $\hat{s}$ . In this case, there is a vertex, denoted  $\bar{p}$ , on the input segment containing  $\bar{q}$  such that

$$|q - q_0| = |\bar{p} - q_0|.$$

The diametral ball between  $q$  and  $\bar{p}$  only intersects line containing segment  $s$  in the interior of  $s$ . So  $q$  has a Delaunay neighbor in this ball and this Delaunay neighbor must be a 1-feature size for  $s$ . Thus  $s$  is unacceptable.

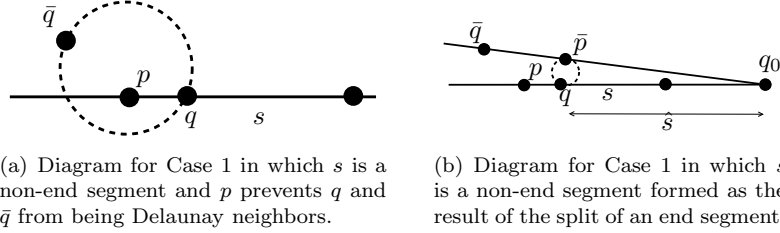


Figure 9: Diagrams for Lemma 10 Case 1

Case 2. Consider any segment  $s$  which is not newly formed. Again we assume this segment  $s$  fails the inductive hypothesis and seek a contradiction. The first criteria of the inductive hypothesis cannot fail as the input vertices and endpoints of  $s$  did not change in the most recent insertion to the mesh (and thus this statement holds by the inductive hypothesis). So the second criterion must fail:  $s$  cannot be queued,  $|s| > \sqrt{2} \text{fs}_1(s)$ , and there is a segment  $\bar{s}$  which is a 1-feature size witness for  $s$  with  $\text{dist}(s, \bar{s}) \leq \frac{|s|}{\sqrt{2}}$ , and  $\bar{s}$  has an endpoint  $\bar{q}$  is such that  $|\bar{q} - q| < |s|$  for some endpoint  $q$  of  $s$ . This vertex  $\bar{q}$  must be the most recent vertex added to the mesh since the inductive hypothesis held at the previous step.

Let  $\hat{s}$  denote the super-segment of  $\bar{s}$  which was split by the insertion of  $\bar{q}$  and let  $q'$  denote the unlabeled endpoint of  $s$ . Next, we possibly relabel  $\bar{s}$  and  $q$  if there is a better selection for our purposes.

(Relabel 1) By possibly relabeling,  $\bar{s}$  can be selected to be the subsegment of  $\hat{s}$  which is closer to  $s$ . This swap can be made because if the original selection of  $\bar{s}$  was incorrect, then the closer subsegment also satisfies

the same set of necessary properties. Then the nearest vertex on  $\hat{s}$  to  $s$  must be in the interior of  $\hat{s}$  since  $\text{dist}(s, \hat{s}) \leq \frac{|s|}{2}$  while  $\text{dist}(q, \hat{s}) > \frac{|s|}{\sqrt{2}}$  and  $\text{dist}(q', \hat{s}) > \frac{|s|}{\sqrt{2}}$  (by the inductive hypothesis).

(Relabel 2) Suppose  $q'$  is the nearest vertex on  $s$  to  $\bar{s}$  and  $|q' - \bar{q}| \leq |s|$ . Then replace  $q$  by  $q'$ .

Since  $s$  is not on the queue, then  $q$  and  $\bar{q}$  cannot be Delaunay neighbors. As in Case 1,  $q$  must have a Delaunay neighbor  $p$  in the diametral ball between  $q$  and  $\bar{q}$  which cannot be a 1-feature size witness for  $q$ . If  $p$  is the endpoint of some segment on the spindle of  $s$ , replace  $q$  with  $p$  and  $s$  with the segment in  $\text{Spind}(s)$  which contains  $p$ . This new  $s$  must have the same length as the original  $s$  as otherwise  $s$  would be queued. The Delaunay property can be applied again since the new  $q$  and  $\bar{q}$  cannot be neighbors. This can be repeated until a vertex  $p$  is found which is not the endpoint of a segment in  $\text{Spind}(s)$ , lies in the diametral ball between  $q$  and  $\bar{q}$  and is not a 1-feature size witness for  $s$ . This configuration is depicted in Figure 10. As  $p$  cannot be a 1-feature size witness for  $s$ ,  $p$  cannot be an input vertex and thus  $p$  belongs to some segment  $s_p$ .

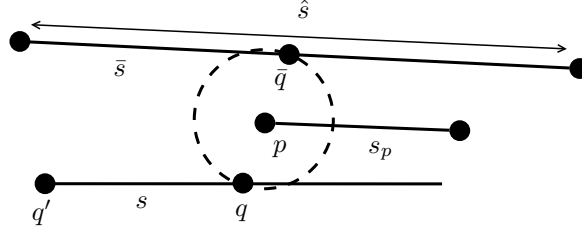


Figure 10: Segment  $s$  fails the inductive hypothesis,  $\bar{q}$  is a nearby feature size witness to  $s$  and  $p$  is not a 1-feature size witness for  $s$ .

Next, we show that  $s$  is a 1-feature size witness for  $\bar{s}$ . If not,  $s$  must be a non-end segment on an input feature which is adjacent to  $\bar{s}$  (since  $\bar{s}$  is a 1-feature size witness for  $s$ ). In this situation,  $p$  cannot exist since it would lie in the end segment adjacent to  $q$  (which is in  $\text{Spind}(\bar{s})$ ). See Figure 11. Thus  $s$  is a 1-feature size witness for  $\bar{s}$ .

Now, we will utilize the Delaunay property, the split size property and a couple geometric facts to assert the following inequalities:

$$|s| > |q - \bar{q}| > |q - p| > |s_p| \geq |\bar{s}| \geq \frac{|s|}{2}.$$

Each of these inequalities is now justified.

- (i)  $|s| > |q - \bar{q}|$  follows from the assumption that  $s$  fails the inductive hypothesis.
- (ii)  $|q - \bar{q}| > |q - p|$  follows from the construction of  $p$  and the Delaunay property.

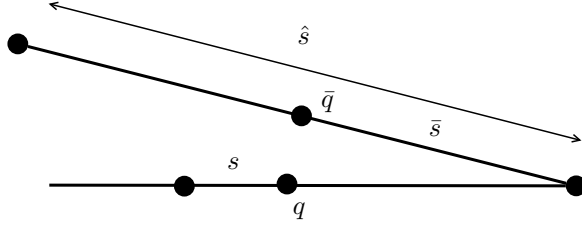


Figure 11: If  $s$  is a non-end segment and  $\bar{s}$  is an end segment,  $p$  cannot exist as it must lie in the end segment adjacent to  $q$ .

- (iii)  $|q - p| > |s_p|$  is a result of Proposition 4.
- (iv)  $|s_p| \geq |\bar{s}|$  follows from the split size property at the time when  $p$  was inserted.
- (v)  $|\bar{s}| \geq \frac{|s|}{2}$  is a result of the split size property before  $\bar{q}$  is inserted.

Finally, a contradiction will be achieved by showing  $|\bar{s}| \geq |q - \bar{q}|$  in three different subcases.

Subcase A. Suppose that  $q$  is the nearest vertex on  $s$  to  $\bar{s}$ . Letting  $\bar{x}$  be the nearest vertex on  $\bar{s}$  to  $s$  as in Figure 12(a), observe that

$$\frac{|s|^2}{2} + |\bar{x} - \hat{q}|^2 > |q - \bar{x}|^2 + |\bar{x} - \hat{q}|^2 = |q - \hat{q}|^2 \geq |s|^2.$$

Thus,  $|\bar{x} - \hat{q}| > \frac{|s|}{\sqrt{2}} > |q - \bar{x}|$ . See Figure 12(a). Then  $|\bar{s}|$  can be estimated using the Pythagorean theorem:

$$\begin{aligned} |\bar{s}|^2 &\geq |\bar{q} - \bar{x}|^2 + |\bar{x} - \hat{q}|^2 \\ &> |\bar{q} - \bar{x}|^2 + |q - \bar{x}|^2 \\ &= |q - \bar{q}|^2. \end{aligned}$$

Thus  $|\bar{s}| \geq |q - \bar{q}|$ .

Subcase B. Let the nearest vertex on  $s$  to  $\bar{s}$ , denoted  $x$ , be in the relative interior of  $s$ . In this case, note that the nearest points between the lines containing  $s$  and  $\bar{s}$  occur in the segments  $s$  and  $\bar{s}$ . Denote these nearest points on  $s$  and  $\bar{s}$  by  $x$  and  $\bar{x}$  and conclude that  $|x - \bar{x}|$  is orthogonal to  $s$  and  $\bar{s}$ .

Let  $P$  be the plane containing  $\bar{s}$  which is orthogonal to  $|x - \bar{x}|$  and let  $\pi$  denote the projection of points into plane  $P$ , depicted in Figure 12(c). Let  $r$  be the radius of the disk  $D = P \cap B(q, |s|)$  so  $r^2 + |x - \bar{x}|^2 = |s|^2$ . Then  $\bar{q} \in D$  (by the failure of the inductive hypothesis) and  $\hat{q} \notin D$  (since  $|\bar{s}| \geq |s|$ ). For an appropriate vertex  $p$  to exist,  $(x - q) \cdot (\bar{q} - \pi(q)) < 0$ . If  $s$  is a non-end segment, this is a result of the fact that  $p$  cannot lie in the interior of  $s$  and thus does not lie in  $\pi(s)$ . In the case of an end segment,  $q'$  is an input vertex and the distance

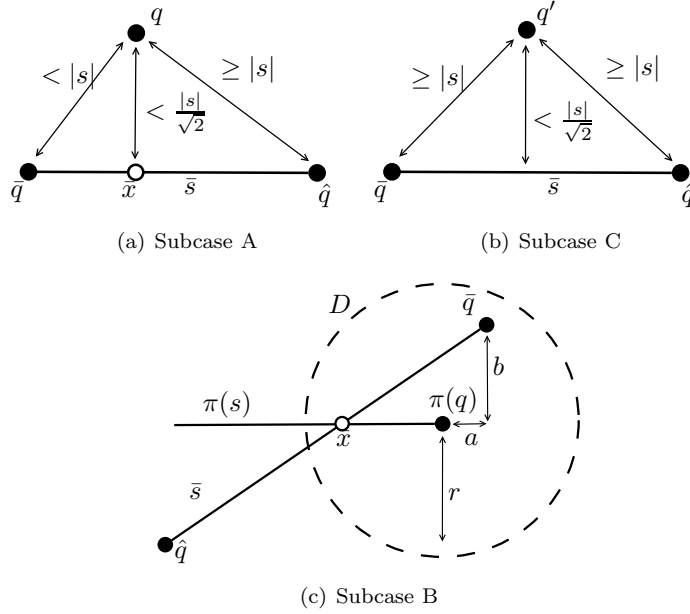


Figure 12: Diagram for Case 2 of Lemma 10

from  $q'$  to  $p$  must be at least  $\frac{3}{2}|s|$ , since  $|s_p| \geq \frac{|s|}{2}$ . The law of cosines gives that  $\cos(\angle p\pi(q)\pi(q')) \leq -\frac{1}{4}$  and thus  $(x - q) \cdot (\bar{q} - \pi(q)) < 0$  holds.

Let  $a$  be the length of the component of  $\bar{q} - q$  in the direction of  $s$  and let  $b$  be the length of the component of  $q - \bar{q}$  which is orthogonal to both  $s$  and  $x - \bar{x}$  as seen in Figure 12(c). This gives the following sequence of inequalities:

$$\begin{aligned}
 |\bar{s}|^2 = |\bar{q} - \hat{q}|^2 &\geq (r + a)^2 + b^2 \\
 &> r^2 + a^2 + b^2 \\
 &\geq \frac{|s|^2}{2} + a^2 + b^2 \\
 &\geq |x - \bar{x}|^2 + a^2 + b^2 \\
 &= |q - \bar{q}|^2.
 \end{aligned}$$

Thus we have achieved the desired inequality,  $|\bar{s}| > |q - \bar{q}|$ .

Subcase C. Suppose that  $q'$  is the nearest vertex on  $s$  to  $\bar{s}$  as depicted in Figure 12(b). By (Relabel 2), we can conclude that the distance from each of the endpoints of  $\bar{s}$  to  $q'$  is at least  $|s|$ . The minimum distance from  $q'$  to  $\bar{s}$  is less than  $\frac{|s|}{\sqrt{2}}$ , and so  $|\bar{s}| > |s|$ . Combining with inequality (i), this implies that  $|\bar{s}| > |q - \bar{q}|$ .

The inequality  $|\bar{s}| > |q - \bar{q}|$  holds in each of the three cases, and thus a contradiction has been reached in each case.  $\square$

The estimates in the Lemma 13 will prove essential in the later steps. The current refinement ensures that the length of each segment is a good estimate (up to a factor of  $4\sqrt{2}$ ) of the 1-feature size on the segment.

Lemma 10 would be much simpler to prove if the split size property could be replaced with a requirement that the length of segments which are split form a non-increasing sequence. Unfortunately, this is not the case. Consider a mesh as outlined in Figure 13(a). Notice that the length two segment is not queued initially, as all vertices are sufficiently far from its endpoints. When the leftmost of the length one segments is split, this midpoint causes the longer length two segment to be queued. See Figure 13(b).

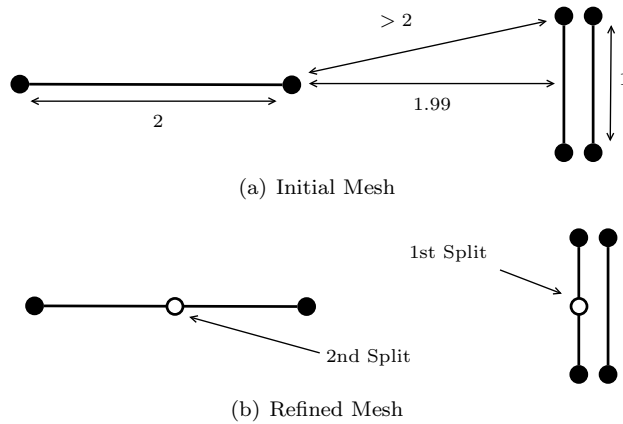


Figure 13: Sequence of split segment lengths is not monotone.

While the length of segments which are split increases, this example does not break the inductive hypothesis! It is important to notice in this case that the initial long segment (of length two which will be denoted by  $s$ ) has a feature size of 1.99 meaning that initially,  $|s| < \sqrt{2} fs_1(s)$ .

(Step 2a) Split segments to improve the 1-feature size estimate.

This step is a simple operation: split all segments into fourths. This is needed as the 1-feature size bound on the segment lengths found in the previous section is not quite strong enough for the algorithm in the next step. Figure 14 shows the result of this step on the example input. This additional refinement strengthens to the bound determined in Step 1b which will be needed in the analysis of Step 2b.

The bounds from Lemmas 9 and 10 are now replaced with slightly different bounds.

**Lemma 11.** *Following Step 2a, for any segment  $s$ ,*

$$\frac{1}{16} fs_1(s) \leq |s|.$$

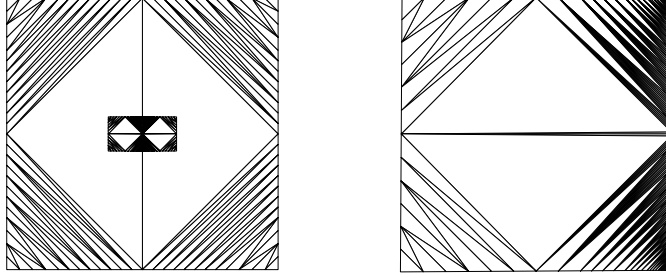


Figure 14: Example mesh following Step 2a.

*Proof.* Let  $s$  be a subsegment of some segment  $\hat{s}$  which existed at the end of Step 1b. It follows that

$$fs_1(s) \leq fs_1(\hat{s}) \leq 4|\hat{s}| = 16|s|$$

and the lemma holds.  $\square$

**Lemma 12.** *Following Step 2a, for any segment  $s$  satisfies*

$$|s| \leq \frac{1}{2\sqrt{2}} lfs_1(s).$$

*Proof.* Following Step 1b,  $|\hat{s}| \leq \sqrt{2} fs_1(\hat{s}) \leq \sqrt{2} lfs_1(\hat{s})$ , for all segments  $\hat{s}$ . If  $s$  is a subsegment of  $\hat{s}$ , then  $lfs_i(\hat{s}) \leq lfs_i(s)$ . Now let  $s$  be one of the four segments created during this step from segment  $\hat{s}$ . Then,

$$\begin{aligned} |s| &= \frac{1}{4} |\hat{s}| \\ &\leq \frac{\sqrt{2}}{4} lfs_1(\hat{s}) \\ &\leq \frac{1}{2\sqrt{2}} lfs_1(s). \end{aligned}$$

$\square$

Note that the estimate  $|s| \leq \frac{1}{2\sqrt{2}} fs_1(s)$  may *not* hold for some segments created during Step 2a. When end segments are split, newly formed non-end segments may have 1-feature size which is much smaller than the 1-feature size of the original end segment .

(Step 2b) Estimate lfs on all segments via Delaunay refinement.

In this step, segments and triangles (in the current Delaunay triangulation of the faces) are split to estimate the local feature size on the segments. This is performed via a Delaunay refinement algorithm which is given in Algorithm 8.

**Algorithm 8** Estimate Feature Size 2D - Step 2b

Action	Insert the circumcenter of a segment or triangle.
Priority	Triangles are given highest priority, in any order. Segments are then prioritized by length.
Unacceptability	A segment $s$ is unacceptable if it has an endpoint $q$ with a Delaunay neighbor $p$ such that $ q - p  <  s $ and either $p$ is a local feature size witness for $s$ or $p$ is a 1-feature size witness for $s$ . A triangle $t$ is unacceptable if it has a vertex $q$ with Delaunay neighbor $p$ such that $ p - q  < 2R_t$ and $p$ does not lie in the face containing $t$ .
Safety	It is not safe to split a triangle in face $f$ if its circumcenter $c$ will have a Delaunay neighbor $q$ which is the endpoint of a segment $s$ in face $f$ and $ c - q  <  s $ .

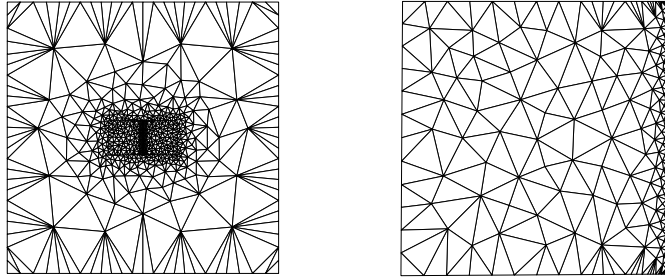


Figure 15: (Left) Example mesh following Step 2b. (Right) Enlarged mesh of one of the smaller squares.

The priority rule in this algorithm is atypical: higher dimensional simplices are usually processed first. This fact will be used in the proof but it is mainly used to simplify the arguments. It is likely that the same (or very similar) results hold using a more traditional priority queue.

The mesh resulting from Step 2b in our running example is given in Figure 15. This is the only step in which vertices are added in faces rather than just on segments.

First, Theorem 5 can be shown using standard ideas from the analysis of Delaunay refinement algorithms.

*Proof of Theorem 5.* Lemma 11 implies  $|s| \geq \frac{1}{16} \text{fs}_1(s)$  for all initial segments. It must be shown that any segment  $s$  formed during Step 2b also satisfies the bound. A segment is only split if it is queued and segments are only queued if there is a nearby 1-feature size witness or local feature size witness. In the first case, an identical argument to that in Lemma 9 implies that  $\frac{1}{4} \text{fs}_1(s) \leq |s|$ . In the second case, a very similar argument yields  $\frac{1}{4} \text{lfs}(s) \leq |s|$ .  $\square$

Theorem 6 is an immediate result of the next two lemmas.

**Lemma 13.** *Upon termination of Step 2b, each segment  $s$  satisfies*

$$|s| \leq \frac{5}{3} \text{lfs}(s).$$

*Proof.* This inequality is shown by induction. Specifically, we show the following inductive hypothesis.

**Inductive Hypothesis** Let  $s$  be a segment such that  $|s| > \frac{5}{3} \text{lfs}(s)$ . If  $s$  is not on the queue then there is some triangle  $t$  which is on the queue.

First, if the inductive hypothesis holds, then the desired bound holds at termination since whenever the desired bound fails, the queue is not empty. Next, suppose that  $s$  is some segment such that  $|s| > \frac{5}{3} \text{lfs}(s)$  and  $s$  is not queued. We will show that this implies that some triangle must be on the queue.

Following Step 2a,  $|s| \leq \frac{\text{lfs}_1(s)}{2\sqrt{2}}$  for all segments  $s$ . Since splitting a segment decreases its length and cannot increase its *local* feature size, this bound will hold on all segments throughout the step. This ensures that no segment or input vertex can be the witness to the local feature size of  $s$ . Thus, there must be an input face  $f$  such that  $\text{dist}(s, f) = \text{lfs}(s)$ . Using Proposition 6, conclude that there is some  $x$  in a face  $f$  and an endpoint  $q$  of  $s$  such that  $\text{lfs}(s) = |x - q|$ , and the vector  $q - x$  is orthogonal to the plane containing  $f$ .

Let  $L$  denote the line containing  $s$ ,  $P$  denote the plane containing  $f$ , and  $\pi$  denote the projection function into  $P$ . Suppose there is a segment  $s' \in \text{Spind}(s)$  with endpoint  $q'$  which is closer to  $P$  than  $q$ . First, estimate the distance from  $x$  to the boundary of  $f$ ,  $\partial f$ :

$$\begin{aligned} \text{dist}(x, \partial f)^2 &= \text{dist}(q, \partial f)^2 - |x - q|^2 \\ &\geq 8|s|^2 - \frac{9}{25}|s|^2. \end{aligned}$$

So  $\text{dist}(x, \partial f) \geq 2|s|$ . Considering any vertex  $p \in s'$ ,

$$|\pi(q') - x| \leq |q' - q| \leq 2|s|.$$

Thus  $\pi(q') \in f$ . This means that  $s$  and  $q$  can be replaced with  $s'$  and  $q'$  and the local feature size bound still fails. Thus without loss of generality, assume that  $s$  is the nearest segment in  $\text{Spind}(s)$  to  $P$ .

In two cases, we show that the triangle  $t$  in  $f$  which contains  $x$  has been placed on the queue.

**Case 1.** Suppose that  $q$  is an input vertex. Now, let  $B$  be the ball of radius  $\frac{|s|}{2}$  with  $q$  on the boundary and  $x$  on its diameter containing  $q$  as in Figure 16(a). The segment  $s$  is not on the queue, so  $q$  cannot have any Delaunay neighbors in  $B$  which witness the feature size of  $s$ . Since  $q$  is an input vertex and the nearest vertex on any segment containing  $q$  to face  $f$ , this means that  $B$  must be empty.

Proposition 7 implies that  $x$  belongs to some triangle in  $f$  with circumradius of at least  $\sqrt{\frac{2}{3}}|q - x|$  as in Figure 17. Then applying Proposition 8 ensures

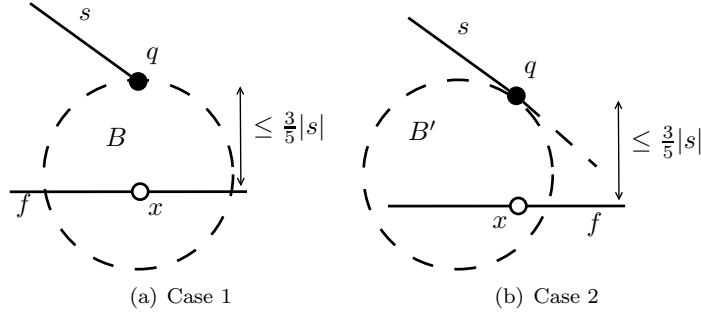


Figure 16: Delaunay neighbors to vertex  $q$  are considered in different balls in the two different cases.

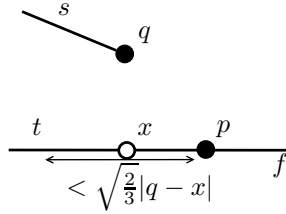


Figure 17: Diagram for Case 1.

that a vertex  $p$  of  $t$  must have a Delaunay neighbor which is not in the face at distance of at most  $\sqrt{\frac{5}{3}}|q - x| < 2R_t$ . Thus  $t$  has been queued at some step of the algorithm.

Case 2. Suppose that  $q$  is not an input vertex. Let  $q_0$  be an input vertex on the segment containing  $s$ . First, claim that  $|q_0 - q| \geq |s|$ . If  $s$  is an end segment, this is trivial. If  $s$  is the subsegment of an end segment which existed at the end of Step 0, then this holds because  $s$  must be produced by a sequence of midpoint insertions. If  $s$  is a subsegment of a non-end segment which existed following Step 0, then  $q_0$  is a 1-feature size witness for  $s$  and Step 2a ensures that  $|s| \leq \frac{1}{2\sqrt{2}} \text{fs}_1(s) < |q - q_0|$ .

Next, consider the angle  $\theta$  between  $L$  and  $\pi(L)$  as in Figure 18(a). Using the fact that  $q$  is interior to an input segment, we will show that  $\sin \theta \leq \frac{3}{5}$ . Let  $y = L \cap P$ . If  $\sin \theta > \frac{3}{5}$  then  $|q - y| \leq |s|$  which means that  $y$  is contained in the input segment containing  $s$  and thus cannot be contained in  $f$ . This means that there is some point  $z$  on the segment  $\overline{xy}$  contained in the boundary of  $f$ . Then the distance between  $z$  and  $q$  is less than  $|s|$ , meaning  $\text{lfs}_1(s) < |s|$ . This violates the bound given in Step 2a which is maintained by the algorithm.

Let  $B'$  be a ball of radius  $\frac{|s|}{2}$  which has a diameter with one endpoint at  $q$  and intersects  $\pi(L)$  as in Figure 16(b). We assert that if  $B'$  is not empty, then the neighbor of  $q$  which lies in  $B'$  must be a local feature size witness for  $s$ . If

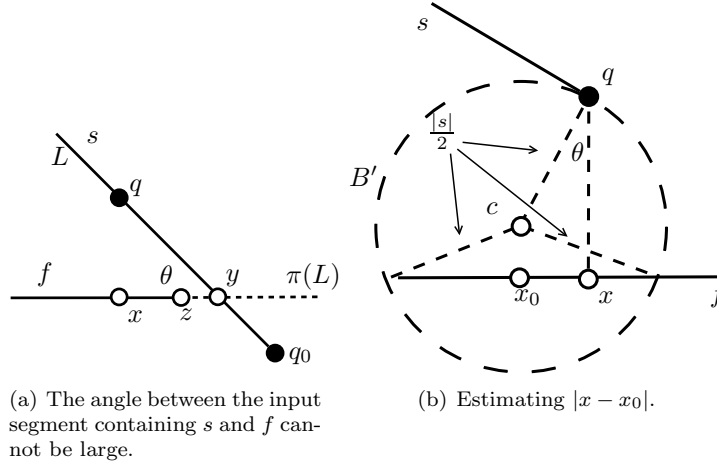


Figure 18: Diagrams for Case 2.

$s$  is a non-end segment, this is clear as  $B'$  only touches the line containing  $s$  at  $q$ . If  $s$  is an end segment, let  $q_0$  be the input vertex which is an endpoint of  $s$ . The ball  $B'$  is below the cone formed by rotating  $L$  around the line containing  $q_0$  and  $\pi(q_0)$ . Since  $q$  is the nearest vertex to  $P$  on the spindle of  $s$ , this implies that  $B'$  does not intersect any input segment containing  $q_0$ .

Since  $s$  is not queued and any vertex in  $B'$  would serve as an appropriate witness to cause  $s$  to be queued,  $B'$  must be empty.

Next we seek to apply Proposition 7 based on the fact that  $B' \cap P \cap f = \emptyset$ . Let  $c$  be the center of  $B'$  and let  $x_0 = \pi(c)$ . As seen in Figure 18(b),  $|x - x_0| = \frac{|s| \sin \theta}{2}$  and the radius of  $B' \cap P$  is

$$\sqrt{\frac{|s|^2}{4} \sin^2 \theta - |q - x|^2 + |q - x| |s| \cos \theta}.$$

Using Proposition 7, conclude that the triangle  $t$  containing  $x$  has circumradius of at least

$$R_t \geq \sqrt{|q - x| |s| \cos \theta - |q - x|^2}.$$

Since  $|q - x| < \frac{3}{5}|s|$  and  $\cos \theta \geq \frac{4}{5}$ ,

$$\begin{aligned} R_t &\geq \sqrt{|q - x| |s| \cos \theta - |q - x|^2} \\ &\geq |q - x| \sqrt{\frac{5}{3} \cdot \frac{4}{5} - 1} \\ &\geq \frac{|q - x|}{\sqrt{3}}. \end{aligned}$$

Now, by Proposition 8, there is a vertex of triangle  $t$  which has a Delaunay neighbor which is not in  $f$  and thus  $t$  has been queued.

In Case 1 and Case 2 it was shown that  $t$  must have been put on the queue. If  $t$  is on the queue, then the inductive hypothesis holds. If the triangle queue is empty, deduce that  $t$  was processed and its circumcenter was rejected for being too close to a nearby edge based on the safety rule.

The circumcenter of  $t$  is only rejected if there was some segment  $\hat{s}$  with endpoint  $\hat{q}$ , such that  $|c_t - \hat{q}| < |\hat{s}|$  and  $\hat{s}$  lies in  $f$ . Since  $f$  is disjoint from the input feature containing  $s$ , this means that  $s$  must be a 1-feature size witness for  $\hat{s}$  and vice versa. Then,

$$\begin{aligned}
|q - \hat{q}|^2 &= |q - x|^2 + |x - \hat{q}|^2 \\
&\leq 3R_t^2 + (|c_t - \hat{q}| + |x - c_t|)^2 \\
&\leq 3R_t^2 + (|\hat{s}| + R_t)^2 \\
&\leq 7|\hat{s}|^2.
\end{aligned} \tag{1}$$

Above, the fact  $|\hat{s}| > |c_t - \hat{q}| \geq R_t$  was used to estimate  $R_t$  by  $|\hat{s}|$ . The second inequality holds since the circumdisk of  $t$  must be empty by the Delaunay property.

The estimate in Lemma 12 is maintained throughout this step, so  $\text{dist}(s, \hat{s}) \geq 2\sqrt{2}|\hat{s}|$ . This inequality contradicts the (1) since  $\sqrt{7} < 2\sqrt{2} = \sqrt{8}$ .  $\square$

Also, it is important that this step maintains the estimate on the 1-feature size derived in Step 1b.

**Lemma 14.** *Upon termination of Step 2b, each segment  $s$  satisfies*

$$|s| \leq \sqrt{2} \text{fs}_1(s).$$

This lemma is immediate for most of the segments. Any end segment must satisfy this bound as  $|s| \leq \frac{1}{2\sqrt{2}} \text{fs}_1$  following Step 2a and reducing the length of an end segment can only increase its 1-feature size. Similarly, for any segment which is a subsegment of a non-end segment which existed at the end of Step 1b, the same argument applies. This leaves only newly formed non-end segments which are subsegments of end segments of the mesh produced by Step 1b.

This proof is nearly identical to the proof of Lemma 10. In the base case, any segment which fails the bound must be queued since adjacent segments have the same length and thus no vertices on the same input segment can prevent the segment in question from being queued. In each step of the proof, nearby Delaunay neighbors of the endpoints of a segment are considered. In the Step 1b proof, either these neighbors are appropriate feature size witnesses to cause the segment to be queued, or they lie on an input segment. In Step 2b, this is still the case, due to the safety rule. This ensures that the endpoints of segment  $s$  will not have any Delaunay neighbors in any plane containing  $s$  within a distance of  $|s|$ .

## 6 Examples

The following three examples are given to demonstrate the 3D algorithm for estimating local feature size.

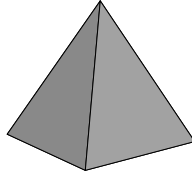


Figure 19: Initial PLC input and final refined mesh for the pyramid example.

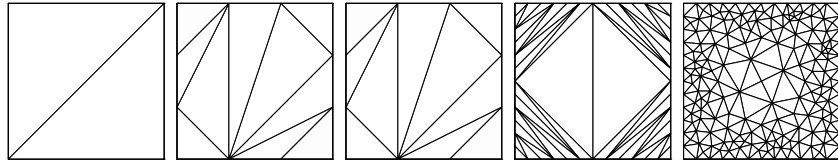


Figure 20: Base of the pyramid following steps 0, 1a, 1b, 2a, and 2b.

*Example 1.* The first example is a square pyramid shown in Figure 19. The mesh of the square base produced following each step of the algorithm can be seen in Figure 20. Similar output for one of the triangular sides is given in Figure 21.

*Example 2.* This example consists of a wheel of 20 faces which lies slightly above a disjoint square as depicted in Figure 22. The mesh of the square base produced following each step of the algorithm can be seen in Figure 23. Similar output for one of the rectangular “spokes” of the wheel is given in Figure 24. Note that the algorithm still terminates even in the presence of acute angles in the input. The number of vertices after each step is listed in Table 1.

*Example 3.* In the final example, we consider a PLC containing two non-convex faces shown in Figure 25. The refinement of one of these faces is shown in Figure 26.

In these examples, nearby edges typically cause more refinement than nearby faces. This is a result of Step 2a which causes segments to be split in fourths *after* they have been refined to realize  $fs_1$ . This can also be seen in Theorem 5 as each segment is guaranteed to have length of at least  $\frac{1}{4} lfs(s)$  or  $\frac{1}{16} fs_1(s)$ . A

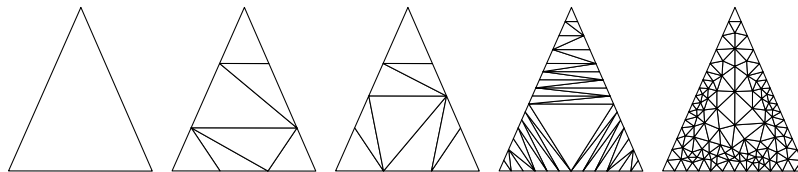


Figure 21: Side of the pyramid following steps 0, 1a, 1b, 2a, and 2b.

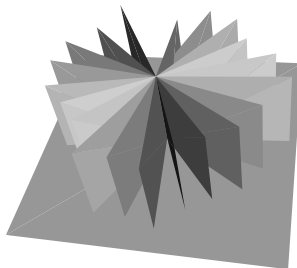


Figure 22: Wheel example input.

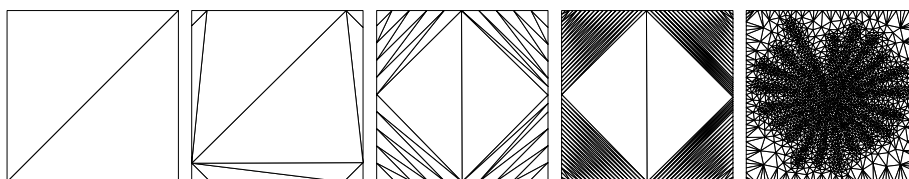


Figure 23: Base plane of the wheel example following steps 0, 1a, 1b, 2a, and 2b.

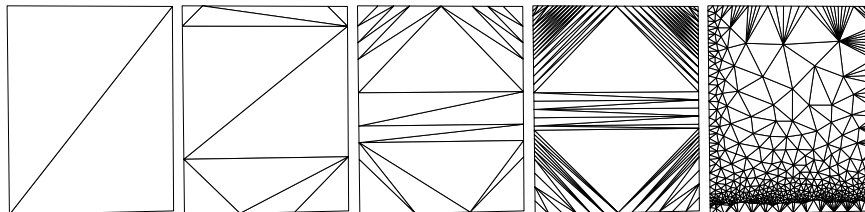


Figure 24: One "spoke" in the wheel example following steps 0, 1a, 1b, 2a, and 2b. The center of the wheel is at the bottom while the disjoint square is to the left of this face.

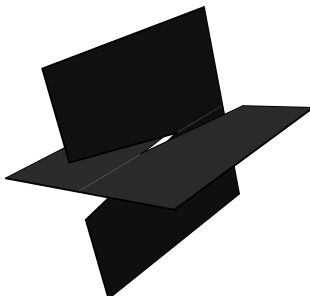


Figure 25: Example containing non-convex input faces.

Table 1: Number of vertices following each step of the algorithm for the wheel example.

Step	0	1a	1b	2a	2b
Vertices	72	202	518	2,051	11,351

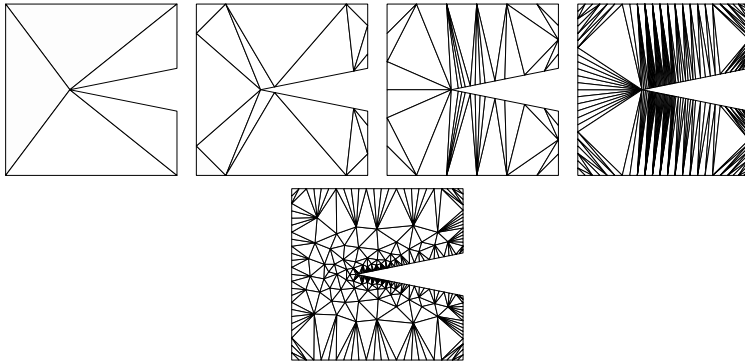


Figure 26: One of the faces in Example 3 following steps 0, 1a, 1b, 2a, and 2b.

small  $fs_1$  does in practice lead to more refinement than simply a small  $lfs$  as was suggested by the constants in the proof.

The proof of Lemma 13 (and thus Theorem 6) uses Step 2a to ensure a bound on each segment's length by  $lfs_1$ . For some segments, this is an over-refinement since they were refined based on  $mfs_1$  and not  $lfs_1$ . We continue to study an adaptive variant of Step 2a which attempts to only split segments in fourths when absolutely necessary.

In practice, the algorithm has been seen to terminate even after changing Step 2a to only split segments in half (instead of fourths). This significantly reduces the output size (often by 50% or more in cases containing small input angles between faces). In further studies, we will seek to justify this modification of the algorithm in the proof or give a counterexample showing that the algorithm can fail without performing Step 2a as specified.

## A Geometric Facts

*Proof of Proposition 1.* Let  $q \in \mathcal{P} \cap \partial B$  and let  $p \in \mathcal{P} \cap B$ . Then there exists a ball  $B'$  such that  $B' \subsetneq B$  and  $\{q, p\} \subset \partial B'$  (see Figure 27). If  $\mathcal{P} \cap B' \neq \emptyset$ , repeat the previous construction using  $B'$  instead of  $B$ . Eventually,  $\mathcal{P} \cap B' = \emptyset$ , since  $\mathcal{P}$  is finite. Then  $q$  must have some Delaunay neighbor in  $\partial B'$ , denoted  $p'$ . (If  $\mathcal{P}$  is in general position, then  $p = p'$ .) Since  $\overline{B'} \setminus B = q$ , conclude that  $p' \in B$ .  $\square$

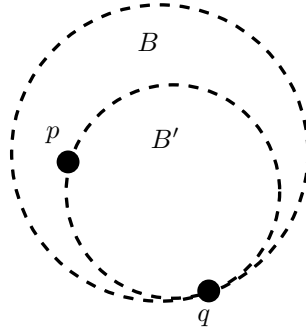


Figure 27: Diagram for the proof of Proposition 1

**Proposition 2.** *Let  $s$  and  $\bar{s}$  be segments with  $|s| \geq |\bar{s}|$ . If  $\text{dist}(s, \bar{s}) < \frac{|s|}{\sqrt{2}}$ , then there are endpoints  $q$  on  $s$  and  $\bar{q}$  on  $\bar{s}$  such that  $|q - \bar{q}| < |s|$ .*

*Proof.* Suppose that all pairs of endpoints are such that  $|q - \bar{q}| \geq |s|$ . The Pythagorean theorem and the fact that  $|s| \geq |\bar{s}|$  imply that

$$\text{dist}(q, \bar{s}) \geq \frac{\sqrt{3}}{2}|s|$$

for either endpoint  $q$  of  $s$ . Again applying the Pythagorean theorem yields that

$$\text{dist}(s, \bar{s}) \geq \frac{|s|}{\sqrt{2}}$$

which completes the proof.  $\square$

The constant in Proposition 2 is sharp. Consider two skew segments, the first with endpoints  $(-1, 0, 0)$  and  $(1, 0, 0)$  and the second between  $(0, -1, \sqrt{2})$  and  $(0, 1, \sqrt{2})$ . Then the distance between the two segments is  $\sqrt{2}$  and the distance between any pair of endpoints is 2.

The next proposition characterizes a special property which holds when the spindle of an end segment contains segments of equal length.

**Proposition 3.** *Let  $s_e$  be an end segment such that*

$$|s_e| = \min_{s' \in \text{Spind}(s_e)} |s'|$$

*Let  $s_n$  be a non-end segment on an input segment which is adjacent to  $s_e$ . If  $|s_e| > |s_n|$  and  $\text{dist}(s_n, s_e) \leq \frac{|s_n|}{\sqrt{2}}$ , then there are endpoints of  $s_n$  and  $s_e$ , given by  $q_n$  and  $q_e$ , respectively, such that  $|q_n - q_e| \leq |s_n|$ .*

*Proof.* Let  $q_0$  be the input vertex contained in  $s_e$ . Pick  $x_e \in s_e$  and  $x_n \in s_n$  such that

$$|x_e - x_n| = \text{dist}(s_n, s_e).$$

Since  $s_n$  and  $s_e$  are coplanar, at least one of the points (either  $x_n$  or  $x_e$ ) is an endpoint of its segment. Since  $s_n$  and  $s_e$  are not parallel, the choice of  $x_n$  and  $x_e$  is unique. First, we argue that  $x_n$  is an endpoint of  $s_n$ . This follows because if not, then the nearest point on  $s_n$  is the nearest point on the line containing  $s_n$  to  $x_e$ . For either endpoint of  $s_e$ , the nearest point on this line is at most a distance  $|s_e|$  away from  $q_0$ . Since  $s_e$  is the shortest segment in its spindle, this point cannot be contained in  $s_n$ .

So,  $x_n$  must be an endpoint of  $s_n$ . Since  $x_n$  is a vertex, it will be denoted  $q_n$ . If  $x_e$  is an endpoint of  $s_e$ , the desired bound holds since by letting  $q_e$  be this endpoint, we observe,

$$|q_e - q_n| = \text{dist}(s_n, s_e) \leq \frac{|s_n|}{\sqrt{2}} < |s_n|.$$

Otherwise,  $x_e$  lies in the interior of  $s_e$ . Let  $a, b, c$  and  $d$  denote the distances shown in Figure 28(a). Note that  $c = \text{dist}(s_n, s_e)$  and  $b + d = |s_e|$ . Also,  $a \geq |s_e|$  and thus

$$\angle q_0 q_e q_n \geq \angle q_0 q_n q_e. \quad (2)$$

Moreover,

$$\begin{aligned} |s_e|^2 &\leq a^2 \\ &= b^2 + c^2 \\ &\leq b^2 + \frac{|s_n|^2}{2} \\ &\leq b^2 + \frac{|s_e|^2}{2}. \end{aligned}$$

Thus  $b^2 \geq \frac{|s_e|^2}{2} \geq c^2$ . This means that  $\angle q_n q_0 q_e \geq \frac{\pi}{4}$ . Combining this with (2) implies that

$$\angle q_0 q_e q_n \geq \frac{3\pi}{8} > \frac{\pi}{4}.$$

This means that  $c > d$  and thus

$$|q_e - q_n|^2 = c^2 + d^2 \leq 2c^2 \leq |s_n|^2$$

which completes the proof.  $\square$

**Proposition 4.** *Let  $s$  be a segment with endpoint  $q$  such that*

$$|s| = \min_{s' \in \text{Spind}(s)} |s'|.$$

*Let  $p$  be a Delaunay neighbor of  $q$  such that  $p$  is not a feature size witness for  $s$ ,  $p$  is not an endpoint of any segment in the spindle of  $s$ , and  $|q - p| < |s|$ . Then  $p$  belongs to a segment  $s_p$  such that  $|s_p| \leq |q - p|$ .*



- There is some endpoint  $q$  of  $s$  and  $x$  in the interior of a disjoint face such that

$$\text{dist}(q, x) = \text{lfs}(s),$$

and  $\overline{qx}$  is orthogonal to the face containing  $x$ .

*Proof.* Suppose  $\text{lfs}(s) \neq \text{lfs}_1(s)$ . Then there exists a face  $f$  and point  $x \in f$  such that  $f$  is disjoint from  $s$  and  $\text{lfs}(s) = |x - y|$  for some  $y \in s$ . For any such  $f$ ,  $x$  and  $y$ ,  $x \notin \partial f$  since otherwise  $x$  would be contained in a segment of the PLC which is disjoint from  $s$  and thus  $x$  would be a witness that  $\text{lfs}(s) = \text{lfs}_1(s)$ . This means that  $x - y$  is orthogonal to the face  $f$ . Further,  $y$  is an endpoint of  $s$  or  $s$  is parallel to some vector in the face  $f$ . In the former case, the proposition has been shown. In the latter, let  $P$  be the plane containing  $f$ . Then

$$\min_{x \in P} |x - y| = \min_{x \in P} |x - y_1|$$

holds for any  $y, y_1 \in s$ . Since

$$\arg \min_{x \in P} |x - y| \notin \partial f$$

for any  $y \in s$ , conclude that  $\arg \min_{x \in P} |x - y| \in f$  for all  $y \in s$ . Thus  $y$  can be selected as an endpoint of  $s$  which completes the proof.  $\square$

**Proposition 7.** Consider a set of coplanar vertices  $\mathcal{P}$ . Suppose ball  $B(x_0, R)$  contains no vertices of  $\mathcal{P}$ . Consider  $x \in B(x_0, R)$  and  $x$  in the convex hull of  $\mathcal{P}$ . Let  $t$  be a triangle in the Delaunay triangulation of  $\mathcal{P}$  containing  $x$ . Then

$$R_t \geq \sqrt{R^2 - |x - x_0|^2}.$$

Using the fact that  $B(x, R - |x - x_0|) \subset B(x_0, R)$  only ensures that  $R_t \geq R - |x - x_0|$ . This weaker bound will hold whenever  $x$  is in the circumdisk of  $t$ . The stronger bound comes from the fact that  $x$  is actually inside triangle  $t$ . This is depicted in Figure 30.

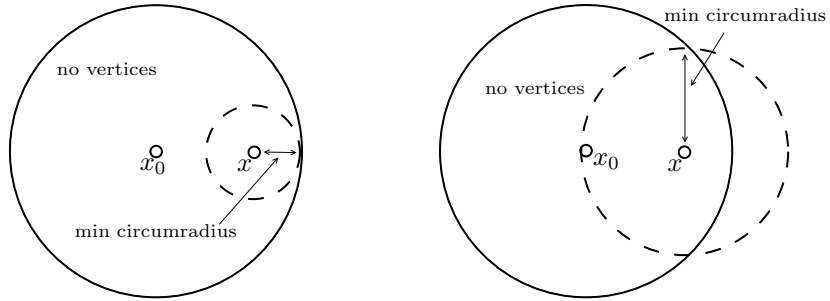
*Proof.* If  $B(x_0, R) \subset B(t)$  or  $B(x_0, R) = B(t)$ , then  $R_t \geq R$  and the result follows. Next, no triangle  $t$  exists such that  $B(t) \subset B(x_0, R)$  since then  $\partial B(t) \setminus B(x_0, R)$  contains at most one point and  $B(x_0, R)$  contains no vertices of  $\mathcal{P}$ . In the remaining case, both  $B(x_0, R) \setminus B(t)$  and  $B(t) \setminus B(x_0, R)$  are nonempty. Let  $\{p_1, p_2\} = \partial B(x, R) \cap \partial B(x_0, R)$  and let  $s = \overline{p_1 p_2}$ . We consider two cases depicted in Figure 31.

Case 1.  $s$  lies between  $x$  and  $x_0$ . Then by the Pythagorean theorem,

$$R^2 = \text{dist}(x_0, s)^2 + \frac{|s|^2}{4}.$$

Applying  $\text{dist}(x_0, s) \leq |x - x_0|$  leads to the inequality

$$R_t \geq \frac{|s|}{2} \geq \sqrt{R^2 - |x - x_0|^2}.$$



(a) If point  $x$  is contained in the circum-circle of (Delaunay) triangle  $t$  and lies in an empty disk which contains no vertices, then the circumradius of  $t$  is at least the distance from  $x$  to the boundary of the empty disk.

(b) If point  $x$  is contained in (Delaunay) triangle  $t$  and lies in an empty disk, then the lower bound on the circumradius of  $t$  is larger than the distance from  $x$  to the boundary of the empty disk.

Figure 30: Diagrams for Proposition 7.

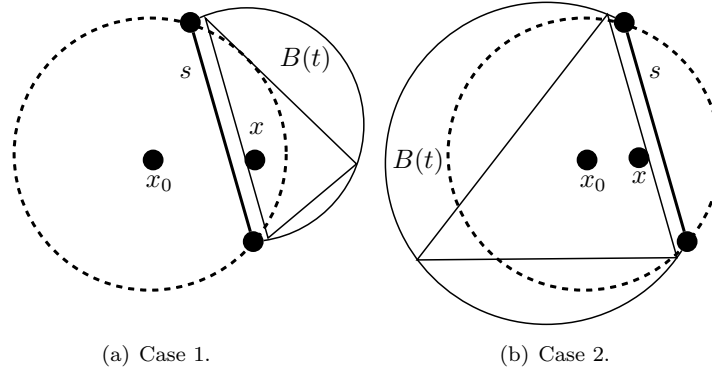


Figure 31: Two cases in the proof of Proposition 7.

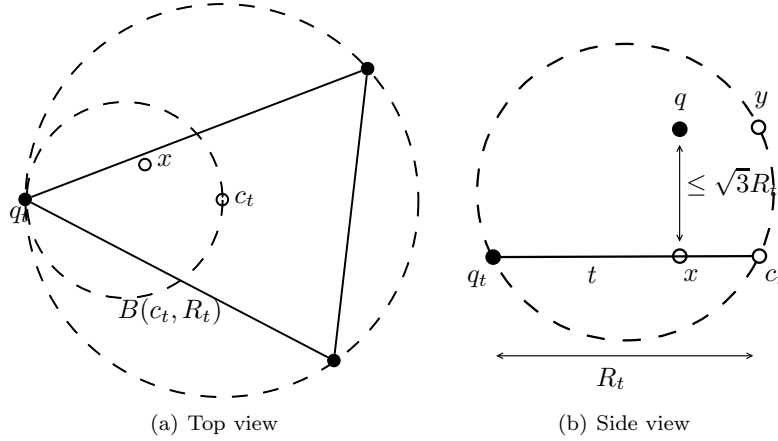


Figure 32: Triangle  $t$  in Proposition 8

Case 2.  $s$  does not lie between  $x$  and  $x_0$ . Let  $L$  be the line which is parallel to  $s$  and passes through  $x_0$ . In this case, observe that

$$L \cap B(x_0, R) \subset B(t).$$

Thus  $R_t \geq R \geq \sqrt{R^2 - |x - x_0|^2}$ .  $\square$

**Proposition 8.** *Let  $t$  be a Delaunay triangle in a face  $f$ . Let  $q$  be a vertex which is not in  $f$  such that the nearest point to  $q$  on the plane containing  $t$ , denoted  $x$ , lies in  $t$ . If  $|q - x| < \sqrt{3}R_t$ , then there is a vertex of  $t$ ,  $q_t$ , which has a Delaunay neighbor,  $p$ , such that  $p$  is not in the face containing  $t$  and  $|q_t - p| < 2R_t$ .*

*Proof.* Let  $q$ ,  $t$ ,  $x$  be as in the statement of the proposition. Let  $c_t$  be the circumcenter of  $t$ . Since  $x$  lies in  $t$ , by Proposition 5, there is a vertex of  $t$ , denoted  $q_t$ , such that  $|x - q_t| \leq R_t$ . See Figure 32. Let  $y = c_t + q - x$ . Then observe the following properties.

- $\partial B(\overline{q_t y}) \cap f$  is the diametral circle between  $c_t$  and  $q_t$ .
- $q \in B(\overline{q_t y})$ .

Finally, applying Proposition 1, it follows that  $q_t$  must have a Delaunay neighbor  $p$  in  $B(\overline{q_t y})$ , and thus  $|q_t - p| \leq |q_t - y| \leq 2R_t$ . Moreover,  $p$  cannot be in the face  $f$  since the diametral circle of  $c_t$  and  $q_t$  must be empty since  $t$  is a Delaunay triangle in the face.  $\square$

## References

- [1] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

- [2] Steven E. Pav and Noel J. Walkington. Robust three dimensional Delaunay refinement. In *Proc. 13th Internat. Meshing Roundtable*, pages 145–156, 2004.
- [3] Michael Murphy, David M. Mount, and Carl W. Gable. A point-placement strategy for conforming Delaunay tetrahedralization. *Internat. J. Comput. Geom. Appl.*, 11(6):669–682, 2001.
- [4] David Cohen-Steiner, Éric Colin de Verdière, and Mariette Yvinec. Conforming Delaunay triangulations in 3D. *Comput. Geom.*, 28(2-3):217–233, 2004.
- [5] Siu-Wing Cheng and Sheung-Hung Poon. Three-dimensional Delaunay mesh generation. In *Proc. 14th Symp. Discrete Algorithms*, pages 295–304, 2003.
- [6] Hang Si and Klaus Gartner. Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. In *Proc. 14th Internat. Meshing Roundtable*, pages 147–163, 2005.
- [7] Hang Si. On refinement of constrained Delaunay tetrahedralizations. In *Proc. 15th Internat. Meshing Roundtable*, pages 510–528, 2006.
- [8] Siu-Wing Cheng, Tamal K. Dey, and Edgar A. Ramos. Delaunay refinement for piecewise smooth complexes. In *Proc. 18th Symp. Discrete Algorithms*, pages 1096–1105, 2007.
- [9] Siu-Wing Cheng, Tamal K. Dey, and Joshua A. Levine. A practical Delaunay meshing algorithm for a large class of domains. In *Proc. 16th Internat. Meshing Roundtable*, pages 477–494, 2007.
- [10] Tamal K. Dey and Joshua A. Levine. Delaunay meshing of piecewise smooth complexes without expensive predicates. Technical Report OSU-CISRC-7108-TR40, Computer Science and Engineering Research Center, Ohio State University, 2008.
- [11] Alexander Rand and Noel Walkington. 3D Delaunay refinement of sharp domains without a local feature size oracle. In *Proc. 17th Internat. Meshing Roundtable*, 2008.
- [12] Siu-Wing Cheng and Sheung-Hung Poon. Three-dimensional Delaunay mesh generation. *Discrete Comput. Geom.*, 36(3):419–456, 2006.
- [13] L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report TR-89-983, Computer Science Department, Cornell University, 1989.
- [14] Alper Üngör. Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed delaunay triangulations. In *Proc. 6th Latin Amer. Symp. Theor. Inform.*, pages 152–161, 2004.

- [15] Andrey Chernikov and Nikos Chrisochoides. Three-dimensional semi-generalized point placement method for delaunay mesh refinement. In *16th Internat. Meshing Roundtable*, pages 25–44, October 2007.
- [16] Ravi Jampani and Alper Üngör. Construction of sparse well-spaced point sets for quality tetrahedralizations. In *Proc. 16th Internat. Meshing Roundtable*, pages 63–80, 2007.
- [17] L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proc. 9th Symp. Comput. Geom.*, pages 274–280, 1993.
- [18] Benoît Hudson, Gary Miller, and Todd Phillips. Sparse Voronoi refinement. In *Proc. 15th Internat. Meshing Roundtable*, pages 339–356, 2006.
- [19] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. Fully incremental 3D Delaunay refinement mesh generation. In *Proc. 11th Internat. Meshing Roundtable*, pages 75–86, 2002.
- [20] Gary L. Miller. A time efficient Delaunay refinement algorithm. In *Proc. 15th Symp. Discrete Algorithms*, pages 400–409, 2004.
- [21] Umut A. Acar, Benoît Hudson, Gary L. Miller, and Todd Phillips. SVR: practical engineering for a fast 3D meshing algorithm. In *Proc. 16th Internat. Meshing Roundtable*, pages 45–62, 2007.
- [22] Alexander Rand. Reordering Ruppert’s algorithm. In *Proc. 18th Fall Workshop Comput. Geom.*, 2008.
- [23] Charles Boivin and Carl Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *Internat. J. Numer. Methods Engrg.*, 55(10):1185–1213, 2002.
- [24] Jonathan Richard Shewchuk. Mesh generation for domains with small angles. In *Proc. 16th Symp. Comput. Geom.*, pages 1–10, 2000.
- [25] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. When and why Ruppert’s algorithm works. In *Proc. 12th Internat. Meshing Roundtable*, pages 91–102, 2003.