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

Alexander Rand* and Noel J. Walkington*

November 2, 2008

Abstract

We present Delaunay refinement algorithms for estimating local feature on the 0-skeleton of a 2D piecewise linear complex (2D) and on the 1-skeleton of a 3D PLC. These algorithms are designed to eliminate the need for a local feature size oracle in 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[18], is an elegant method for computing a quality, conforming Delaunay triangulation of a non-acute 2D piecewise-linear complex (PLC). Initial attempts to extend this algorithm to handle 3D PLCs with acute input angles have required the ability to evaluate the local feature size function on the 1-skeleton of the mesh[5]. As observed by Pav and Walkington[16], Ruppert's algorithm does not require local feature size and instead, it can be used to compute local feature size. This can be seen by considering a simplified version of Ruppert's algorithm: Algorithm 1 describes Ruppert's algorithm without any quality requirement on the output triangles.

Algorithm 1 Ruppert's Algorithm - conformity only

```
Queue all encroached segments.
while the queue of segments is nonempty do
  Insert the midpoint of the front simplex.
  Update the queue based on changes in the Delaunay triangulation.
end while
```

Upon termination of Algorithm 1, the local feature size at any input point can be approximated well 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 point q_0 in the mesh, the following estimates hold.*

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

Here, $\mathrm{lfs}(\cdot)$ denotes the local feature size of the input PLC and $N(\cdot)$ denotes the distance to the nearest neighbor in the resulting Delaunay triangulation. Complete definitions are given in Section 2.

*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.

This result means that the distance from an input point to its nearest neighbor in the resulting Delaunay triangulation is a good estimate for the local feature size of the input complex at that input point. This is useful, as the distance to the nearest neighbor can be computed locally in a Delaunay triangulation while generically local feature size depends on the entire input. (Local feature size is a local quantity in terms of the Euclidean distance but, despite its name, is a global quantity with respect to the input data.)

While Ruppert's algorithm can be used to compute both a conforming triangulation and approximate local feature size, initial 3D Delaunay refinement algorithms for computing conforming Delaunay tetrahedralizations required a local feature size oracle available to the algorithm[15, 8]. As these ideas were extended to the problem of quality meshing of non-acute domains[5, 4], this requirement that local feature size must be precomputed was preserved. Naively, computing local feature size requires a brute force $O(n^2)$ search through the input data.

An algorithm of Pav and Walkington[16] removes this requirement. It produces a quality, conforming Delaunay mesh without using local feature size. This algorithm does not quite preserve the original features of Ruppert's algorithm: the encroachment operations of the Pav-Walkington algorithm are local in the Euclidean distance but not in the Delaunay triangulation.

While the algorithm of Pav-Walkington has not been implemented, two other meshing algorithms have been implemented which avoid a brute force calculation of local feature size before performing the Delaunay refinement. In both cases, the mesh produced is graded to an alternative sizing function which can be computed locally in the Delaunay triangulation. The first is the constrained Delaunay refinement algorithm of Si and Gartner[20, 19]. In this case, a constrained Delaunay tetrahedralization is computed and local feature size is estimated at the *input points*. This is then interpolated incrementally on the edges as the refinement progresses. The second is the algorithm of Cheng, Dey and Ramos[4, 3] with its recent improvements[9]. This algorithm creates weighted Delaunay meshes of smooth surfaces from only local information. However, their algorithm still requires an user specified length scale parameter (as opposed to the full local feature size function) and does not guarantee output is local feature size graded.

We give a Delaunay refinement algorithm for estimating local feature size based on only local information in the current Delaunay triangulation. The result of the algorithm will be that local quantities in the resulting Delaunay triangulation are equivalent to the local feature size function. This algorithm is suitable to replace the brute force preproccess for existing Delaunay refinement algorithms for computing conforming Delaunay meshes using existing algorithms[8, 5], and it has been practically implemented for quality mesh generation[17].

The remainder of the paper is organized as follows. Section 2 gives the fundamental definitions and some geometric results. In Section 3, a generic Delaunay refinement algorithm is described which the algorithms for estimating local feature size specialize. Section 4 describes and analyzes a simpler 2D algorithm for estimating local feature size which demonstrates the key concepts. Then, the full 3D version is analyzed in Section 5. Finally, some examples of this algorithm are shown in Section 6.

2 Preliminaries

A piecewise linear complex is a pair of sets of input points and input segments in 2D, $\mathcal{C} = (\mathcal{P}, \mathcal{S})$ or a triple of sets of input points, input segments and input faces in 3D, $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$. In two dimensions, a PLC $\mathcal{C}' = (\mathcal{P}', \mathcal{S}')$ is a refinement of the PLC $(\mathcal{P}, \mathcal{S})$ if $\mathcal{P}' \subset \mathcal{P}$ and each segment in $\mathcal{S}$ is the union of segments in $\mathcal{S}'$. In three dimensions, a PLC $(\mathcal{P}', \mathcal{S}', \mathcal{F}')$ refining $(\mathcal{P}, \mathcal{S}, \mathcal{F})$ must satisfy the additional condition that each face in $\mathcal{F}$ must be the union of faces in $\mathcal{F}'$.

Definition 1. Consider a refinement $(\mathcal{P}', \mathcal{S}', \mathcal{F}')$ of an input PLC $(\mathcal{P}, \mathcal{S}, \mathcal{F})$.

- An **end segment** is a segment in $\mathcal{S}'$ for which at least one end point is an input point in $\mathcal{P}$.
- An **end triangle** is a triangle in $\mathcal{F}'$ for which at least one vertex lies on an input segment in $\mathcal{S}$.
- The **spindle** of segment s in $\mathcal{S}'$, denoted Spind(s) is set containing

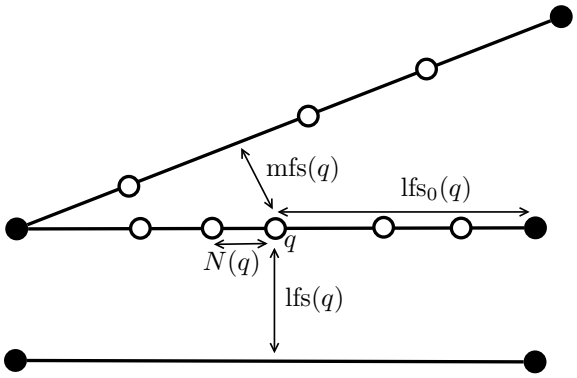


Figure 1: Example of sizing functions in Definition 2 for a 2D PLC. The black points represent input points while the white points represent vertices inserted during the refinement.

- 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 a point q inserted into the mesh, r_q denotes the insertion radius of point q: this is the distance from q to its nearest neighbor when it is inserted into the mesh. 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 $B(\widehat{abc})$.

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 sizing function (called mesh feature size) which is used throughout the arguments.

Definition 2. Let $\mathcal{C} = (\mathcal{P}, \mathcal{S}, \mathcal{F})$ be a PLC with refinement $\mathcal{C}' = (\mathcal{P}', \mathcal{S}', \mathcal{F}')$.

- The i-**local feature size** at point x with respect to $\mathcal{C}$, $\operatorname{lfs}_i(x, \mathcal{C})$ 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 $\mathcal{C}$, $\operatorname{mfs}_i(x, \mathcal{C})$ 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**, $N(x, \mathcal{P}') := \operatorname{lfs}_0(x, \mathcal{C}')$, returns the distance from x to its second nearest neighbor in $\mathcal{P}'$.

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 the be infimum of the function over the set. The dimension argument will be omitted when considering the $d-1$ dimensional feature sizes. For instance $\operatorname{lfs}_i(s, \mathcal{C}) := \inf_{x \in s} \operatorname{lfs}_i(x, \mathcal{C})$ and, in 3D, $\operatorname{lfs}(x, \mathcal{C}) := \operatorname{lfs}_2(x, \mathcal{C})$.

Convention: Throughout the analysis, local/mesh feature size will always be evaluated with respect to the input PLC. By adhering to this usage, the PLC argument can be omitted.

Finally, much of the analysis involves giving identical estimates for 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 3. The i-**feature size** of segment s is defined as follows.

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

Given simplex s in $\mathcal{C}'$, 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.

This definition is very similar to the local gap size used by Cheng and Poon[5]. The notion of i-feature size witness is 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] Consider the Delaunay triangulation or tetrahedralization of a set of points $\mathcal{P}$. Let B be a ball with point $q \in \mathcal{P}$ on the boundary of B. If there is a point of $\mathcal{P}$ inside B, then q has a Delaunay neighbor that is inside B.*

3 The Generic Delaunay Refinement Algorithm

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

Algorithm 2 Delaunay Refinement

```
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.
    Queue additional encroached simplices.
  end if
  Remove the front simplex from the queue.
end while
```

This generic algorithm leaves four of operations to be specified in order to have a concrete algorithm. These are described below.

Action	Where should Steiner points be inserted to "split" a simplex? Should this point yield to a lower dimensional feature?
Priority	In what order should be queue be processed?
Unacceptability	Which simplices are unacceptable?
Safety	Which simplices are safe to split?

First, consider how Algorithm 1 (the simplified version of Ruppert's algorithm that only considers splitting segments based on conformity) specializes the generic algorithm.

Action	When a segment is processed, its midpoint is inserted.
Priority	Segments can be processed in any order.
Unacceptability	A segment is unacceptable if it has a nonempty diametral disk.
Safety	It is safe to split any simplex.

It is important to note that each of these specifications for the algorithm can be computed locally in the Delaunay triangulation of the current point 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.

There are many different actions that the mesh can take to remove bad simplices. Inserting the circumcenter is a natural choice for Delaunay triangulations since this gives the furthest guaranteed distance between the point and any others in the mesh based on only the simplex which is being split. Off-center points and general selection regions have also been studied[21, 6, 11] using the same yielding procedure as Ruppert's algorithm. An example of a very different action taken by the algorithm can be seen with Chew's "second" Delaunay refinement algorithm[7]. This method maintains a constrained Delaunay triangulation, involves a different yielding procedure and removes points from the mesh following circumcenter splits.

The priority queue for most algorithms involves prioritizing lower dimensional simplices before higher dimensional ones[13]. For time efficient algorithms, this priority queue must be further specified[12, 10, 1], often requiring simplices queued for quality to be processed before those queued based on encroachment.

There are typically two types of unacceptability criteria: encroachment criteria to ensure that the required input features exist in the refined mesh, and quality requirements which are desirable of the output mesh. Delaunay based methods usually look for a nonempty circumball to determine if a segment is acceptable. The quality criteria is usually based on the circumradius-shortest edge ratio. Throughout this paper, we are not concerned with this type of criteria and thus segments will be unacceptable only if they are encroached in some sense. These two terms (unacceptability and encroachment) will be used interchangeably.

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 skinny angle[14]. In 3D, the Tetgen code[20, 19] relies on a similar principle for determining when to stop refining near small input angles.

In the Delaunay refinement algorithms we describe for estimating local feature size, it is important to define unacceptability, priority and safety properties in a different fashion from previous examples. We will always insert the circumcenter of the simplex to be split in all of the algorithms and thus the action specification will not be discussed any further.

4 Estimating Feature Size in 2D

We give a Delaunay refinement algorithm which estimates the local feature size at input points in terms of the distance to its nearest Delaunay neighbor. The refinement algorithm is similar to Algorithm 1, but care must be taken to ensure that skinny angles to not prevent termination. The algorithm is summarized below.

Algorithm 3 Estimate Feature Size 2D

(Step 0) Compute the Delaunay tetrahedralization of the set of input points.
(Step 1) Estimate lfs at all input points via Delaunay refinement.

Step 1 of Algorithm 3 is a Delaunay refinement algorithm specified by the following four rules.

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 end point 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 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

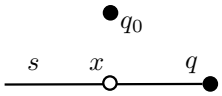


Figure 2: Diagram for proof of Theorem 3

shown in Ruppert's analysis.

Theorem 2. *Algorithm 3 terminates. At any point during the Algorithm 3, each input point q_0 satisfies*

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

Proof. Initially, $N(q_0) = \mathrm{lfs}_0(q_0) \geq \mathrm{lfs}(q_0)$ so the base case holds. Suppose a point p is inserted in the mesh and p is the closest neighbor to an input point q_0. Then p must be the midpoint of a segment s. If s is disjoint from q_0, then $\mathrm{lfs}(q_0) \leq |q_0 - p|$. If p is on an adjacent segment, then (by the unacceptability rule) the point encroaching s, denoted p', must be on a segment which is disjoint from q_0. Thus, $\mathrm{lfs}(q_0) \leq |q_0 - p'| \leq 2|q_0 - p|$. This bound ensures the termination of the algorithm. □

Next, it is seen that the distance from an input point to its nearest neighbor in the resulting mesh also provides a lower bound on the local feature size.

Theorem 3. *Upon the termination of Algorithm 3,*

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

for all input points $q_0 \in \mathcal{P}$.

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

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

Otherwise, the local feature size of q_0 is realized by a segment s. Let q be the nearest end point of s to q_0 and let x be the nearest point on segment s to q_0 as in Figure 2. Suppose $N(q_0, cP') > \sqrt{2}\mathrm{lfs}(q_0)$. Then the following inequalities hold.

$$\begin{aligned} 2\,\mathrm{lfs}(q_0)^2 &< N(q_0)^2 \\ &< |q_0 - q|^2 \\ &= \mathrm{lfs}(q_0)^2 + |x - q|^2 \\ &< \mathrm{lfs}(q_0)^2 + \frac{|s|^2}{4} \end{aligned}$$

Conclude that $2\,\mathrm{lfs}(q_0) \leq |s|$. This inequality implies that q_0 lies in the diametral disk of s. Thus s must be encroached unless there is a point p in the $B(\overline{q_0 q})$ which lies on a segment adjacent to s. If this p exists, then $|p - q| \leq |x - q| \leq \frac{|s|}{2}$ as seen in Figure 3. Thus $|s|$ is at least double the length of the segment between p and q which is in the spindle of s. Thus s is unacceptable.

Since s is unacceptable and the any segment is considered safe to split, conclude that the algorithm has not terminated. Thus, upon termination, the desired bound holds.

□

The constants in Theorem 2 and Theorem 3 are both sharp and independent of the smallest input angle in the mesh. The exact same estimates which are achieved by Algorithm 1 for non-acute PLCs have been shown for Algorithm 3 allowing general input.

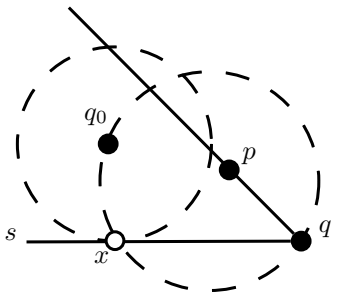


Figure 3: In the case that s is an end segment and q does not "see" q_0, input point q_0 must encroach s as $|p-q| \leq \frac{|s|}{2}$

5 Estimating Feature Size in 3D

In the 3D case, it becomes important to estimate local feature size and 1-feature size on all segments (the $d-2$ dimensional features) of the input complex. While the distance to the nearest neighbor of the resulting mesh was used to estimate feature size in 2D, the 3D analogy uses the length of segments to estimate the feature size of those segments.

Algorithm 4 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 4 Estimate Feature Size 3D

(Step 0) Compute the Delaunay tetrahedralization of the set of input points.
(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 refine the 1-feature size estimate.
(Step 2b) Estimate lfs on all segments via Delaunay refinement.

The following two theorems show that in the resulting PLC, the length of each segment is a good estimate for the local feature size or 1-feature size of that segment.

First, the lower bound on segment lengths. This theorem will be shown by techniques used in the standard analysis of Ruppert's and other Delaunay refinement algorithms.

Theorem 4. *At any point during Algorithm 4, all segment segments $s \in \mathcal{S}'$ satisfy*

$$\min\left(\frac{1}{16}\text{fs}_1(s), \frac{1}{4}\text{lfs}(s)\right) \leq |s|.$$

The upper bound on segment lengths is an estimate which is generally not seen in previous analysis.

Theorem 5. *Following Algorithm 4, all segment segments $s \in \mathcal{S}'$ satisfy*

$$|s| \leq \min\left(\sqrt{2}\text{fs}_1(s), \frac{5}{3}\text{lfs}(s)\right).$$

In order to show these two theorems, output conditions on the PLC are determined following each step of the algorithm. The output conditions of Step 2b will be exactly the results of the theorems.

While describing this algorithm, the result at each step is considered on a simple example. The example consists of three squares: one large square which is slightly below two coplanar, disjoint squares as seen in Figure 4. 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.

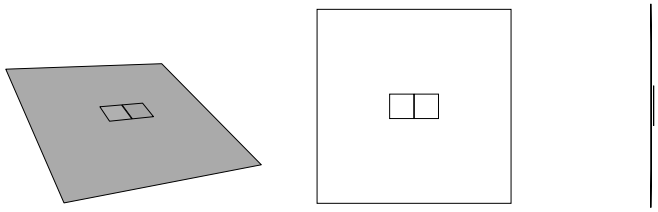


Figure 4: This descriptive example consists of 3 faces: one large square which is slightly below two smaller squares which are side by side.

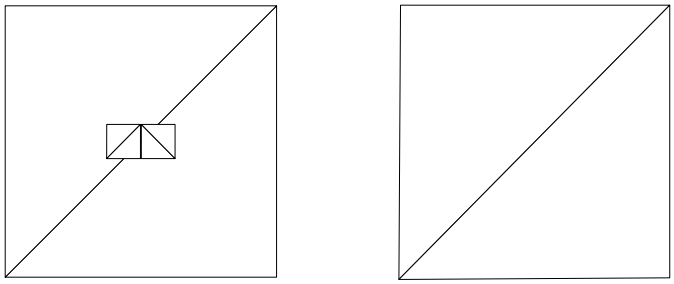


Figure 5: (Left) Example mesh following Step 0. (Right) Enlarged mesh of one of the smaller squares.

Step 0

The first step of the algorithm involves computing the Delaunay triangulation of the input point set in each of the meshes to be maintained (typically with the Bowyer-Watson algorithm[2, 22]). In our running example, this simply leads to the Delaunay triangulation of each of the squares as seen in Figure 5.

Then the following inequalities hold following this step for all points in the mesh.

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

While this procedure does not admit an optimal run-time in the worst case, this is common part of to all current Delaunay refinement algorithms which handle small angles in the input[13]. Algorithms which rely on constrained Delaunay triangulation[20, 19] also typically begin with this step.

Step 1a

This step consists of a single pass of each of the input points. For each input point q_0, split all segments containing this point at a distance of $\frac{N(q_0)}{3}$ away from q_0.

The result of this step on our example can be seen in Figure 6. 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.

Following this step, the following properties hold.

- $N(q_0, \mathcal{P}') = \frac{1}{3}\text{lfs}_0(q_0)$ holds for all input points $q_0 \in \mathcal{P}$.
- Adjacent segments cannot encroach each other.
- If s_n is a non-end segment and s_e is an adjacent end segment, $|s_n| \geq |s_e|$.
- If s_e and s'_e are end segments and $s'_e \notin \text{Spind}(s_e)$, then $\text{dist}(s_e, s'_e) \geq \max(|s_e|, |s'_e|)$.

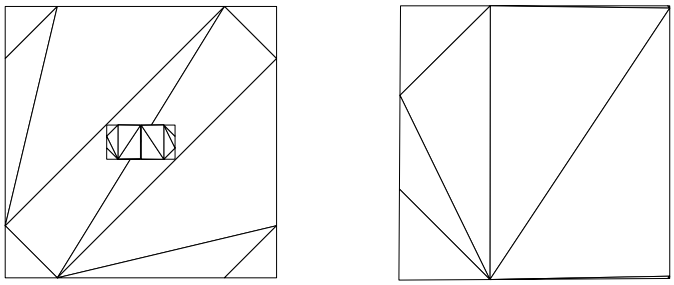


Figure 6: (Left) Example mesh following Step 1a. (Right) Enlarged mesh of one of the smaller squares.

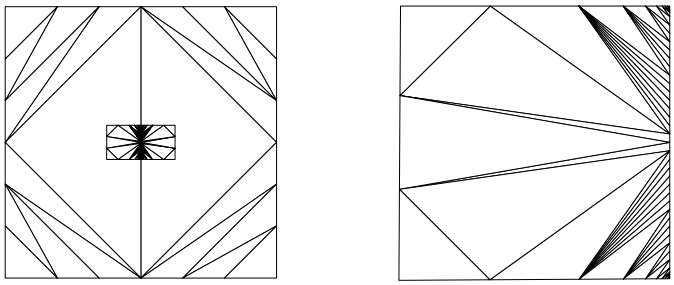


Figure 7: Example mesh following Step 1b.

This final property is particularly important. Observe that after splitting any segments in the mesh, this property continues to hold and thus will hold for the remainder of the algorithm. This ensures that spindles of end segments corresponding to different input point will not interact at any point in the upcoming arguments.

Step 1b

The goal of this stage is to bound the length of each segment in the mesh by the 1-feature size of that segment. The Delaunay refinement algorithm is specified as follows.

Action	Insert the midpoint of a segment.
Priority	Longer segments are processed first.
Unacceptability	Segment s is unacceptable if it there is an end point q of a segment in 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.

By the specification given, checking if a simplex is unacceptable requires that only Delaunay neighbors of the endpoints of the segment in question need to be queried. This is an important property of Ruppert's algorithm that we are careful to maintain.

Consider the result of applying this step to the earlier example as seen in Figure 7. Notice that the main effect is that the nearby edges of the two small squares refine to realize the feature size. The other segments only need to split a few times.

First, see 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 for Delauany refinement algorithms.

Lemma 6. *Throughout this Step 1b, the following bound on segment lengths holds.*

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

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

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

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 point (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, initial, the lower bound on segment lengths holds.

The inductive step is shown in two cases corresponding to splits of end and non-end segments.

Case 1. Consider an end segment s_e from q_0 to q which is split at midpoint p, forming an new end segment s'_e and a non-end segment s'_n. This means that there is a point $\bar{q}$ on a disjoint feature to s_e which is of distance at most $|s_e|$ from some adjacent end segment to s_e.

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

For the non-end segment s_n, q_0 becomes a 1-feature size witness.

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

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

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

Conclude that $\frac{1}{4}\text{fs}_1(s) \leq |s|$ for all segments in the mesh created during Step 1b.

This lower bound on feature size of all segments ensures termination. □

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 not necessary. In order to get the upper bound, an arbitrary order does not 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 point which is not on the segment. This segment may not see this nearby point on its Delaunay cavity. Note: this requires another point to block the long segment from seeing the nearby disjoint point, but this point could be far away and thus not causing the long segment to split.

By prioritizing the queue by segment length, this situation cannot arise, and it is possible to bound the resulting segments lengths by 1-feature size. In order to prove this, a number of geometric facts are necessary.

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 p and $\bar{p}$ such that $|p - \bar{p}| < |s|$.*

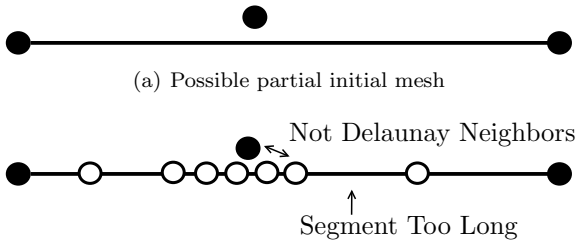
Proof. Suppose that all pairs of endpoints are such that $|p - \bar{p}| \geq |s|$. The Pythagorean theorem and the fact that $|s| \geq |\bar{s}|$ imply that

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

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

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

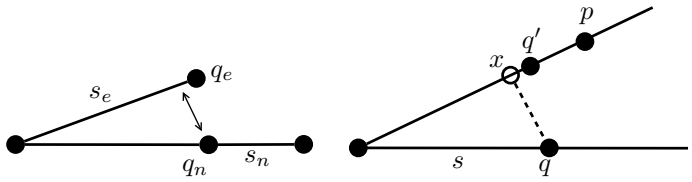
□



(a) Possible partial initial mesh

(b) Possible refinement

Figure 8: Lemma 7 does not hold without specifying a refinement order.



(a) Proposition 3 (b) Proposition 4

Figure 9: Diagrams for the proofs of two propositions.

The constant above 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 of $\sqrt{2}$ and the distance between any pair of endpoints is 2.

Proposition 3. *Suppose that all adjacent end segments have the same length. Let s_n be a non-end segment and s_e be an adjacent end segment to the input segment containing s_n. 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. This proof follows from the Pythagorean theorem. See Figure 9(a). □

Proposition 4. *Let s be a segment in the mesh 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 $|s_p| \leq |q - p|$.

Proof. If p lies on the same input segment as q, then there clearly exists s_p between p and q such that $|s_p| \leq |p - q|$.

Otherwise, s is an end segment and p lies on an adjacent input segment segment, denoted s_0. Let x be the nearest point on this adjacent input segment to q. Then $|q - p| > |x - p| > |q' - p| > |s_p|$. See Figure 9(b). Thus the result holds. □

With these facts, it is possible to show the desired bound on segments following Step 1b.

Lemma 7. *At the end of Step 1b, all segment satisfy*

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

Proof. The following statement is shown inductively.

Inductive Hypothesis: If segment s is not queued and $|s| > \sqrt{2}\text{fs}_1(s)$, then the following two statements hold.

1. If q_0 is an input point 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 end point of s, and $\bar{q}$ is an endpoint of $\bar{s}$, then $|q - \bar{q}| \geq |s|$.

The following property follows from the inductive hypothesis. From this property, it will be clear that the inductive hypothesis is sufficient to imply the inequality in the lemma. Also, when proving the inductive hypothesis, it will be useful to apply this property at earlier steps in the algorithm, rather than use the inductive hypothesis directly.

Split Size Property. If the inductive hypothesis holds, any longest segment s such that $|s| > \sqrt{2}\text{fs}_1(s)$ is on the queue.

Assuming the inductive hypothesis, we show that the split size property holds. 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 point. Thus, there exists an 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

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

Conclude that s is not the longest segment failing the feature size bound.

From the split size property, conclude that if the inductive hypothesis holds, then upper bound on feature size of segments holds when the algorithm terminates. Thus the above inductive hypothesis is sufficient to imply the lemma. Next, we show that the inductive hypothesis holds in the base case.

Base Case. First consider initial end segments. Let q_0 be an input point contained in end segment s_e. The construction ensures that for all other points $\bar{q}$ 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. Any point in the mesh which is not an endpoint of s_n is a 1-feature size witness for s_n. If there is another point in the mesh which is of distance less than $|s_n|$ from 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, proceed to show 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 last insertion and segments that where formed during the last insertion.

Case 1. Consider any newly formed segment s and suppose s violates the inductive hypothesis. So, $|s| > \sqrt{2}\text{fs}_1(s)$, s is not on the queue, and there is an point $\bar{q}$ and end point q of s such that $|q - \bar{q}| < |s|$ and $\bar{q}$ is a feature size witness for 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 point p in $B(\overline{q\bar{q}})$ 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 point p on one of these segments at a distance of less than $|s|$ which is not an end point of some segment in $\text{Spind}(s)$.

If s is a non-end segment, consider the mesh at the step when p was inserted into the mesh. At this point, s belonged to a segment of length at least $2|s|$ which also must have failed the feature size bound. But a segment of length $2|q - p| < 2|s|$ was split to insert p. This violates the split size property.

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. To fail the inductive hypothesis, s cannot be queued, $|s| > \sqrt{2}\text{fs}_1(s)$, and there is a point $\bar{q}$ is such that $|\bar{q} - q| < |s|$ for some endpoint q of s such that $\bar{q}$ is a 1-feature size witness for s. This point $\bar{q}$ must be the most recent point added to the mesh since the inductive hypothesis held at the previous step and thus applies to s. So, $\bar{q}$ is an end point of a new segment $\bar{s}$ such that $\text{dist}(s, \bar{s}) \leq \frac{|s|}{\sqrt{2}}$.

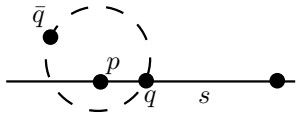


Figure 10: Diagram for case 1 in which s is a non-end segment in and p prevents q and $\bar{q}$ from being Delaunay neighbors.

Let $\hat{s}$ denote the supersegment of $\bar{s}$ with midpoint $\bar{q}$ and let q' denote the unlabeled endpoint of s. Next, we possibly relabel s and q if there is a better selection for our purposes.

- 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 point 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 $\text{dist}(q', \hat{s}) > \frac{|s|}{\sqrt{2}}$.
- Suppose q' is the nearest point 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 in $B(\overline{q\bar{q}})$, denoted p, which cannot be a 1-feature size witness for q. If p is the endpoint of some segment on the spindle of s, the Delaunay property can be applied again since p and $\bar{q}$ cannot be neighbors. This can be repeated until a point p is found which is not the endpoint of a segment in $\text{Spind}(s)$, lies in $B(\overline{q\bar{q}})$ and is not a 1-feature size witness for s. This configuration is depicted in Figure 11.

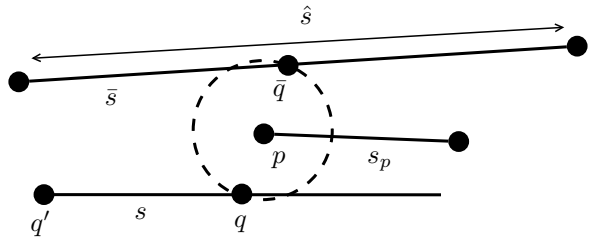


Figure 11: Segment s fails the inductive hypothesis, $\bar{q}$ is a nearby feature size witness to s and p is not a feature size witness for s.

As p cannot be a 1-feature size witness for s, p cannot be an input point and thus p belongs to some segment s_p.

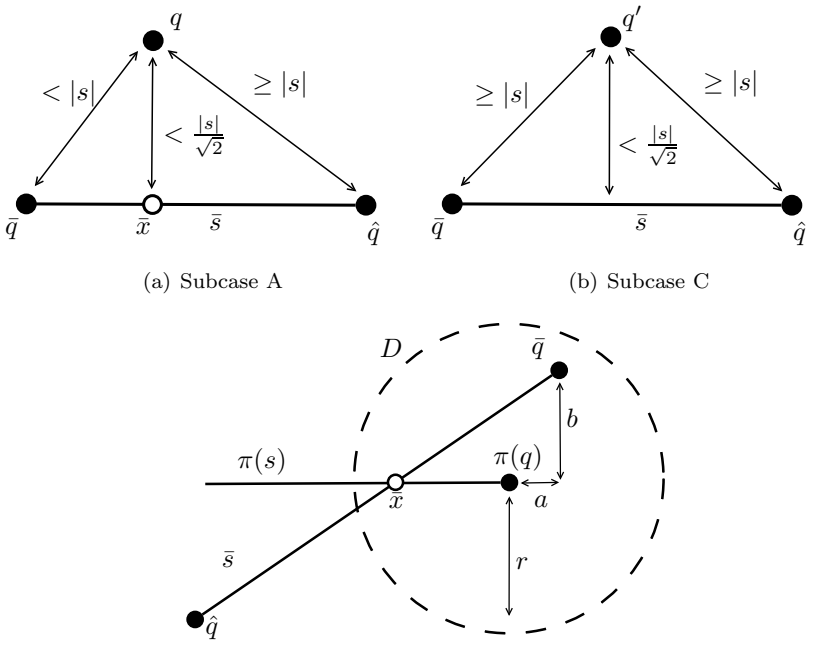
Next, we show that s is a 1-feature size witness for $\bar{s}$. If not, s must be a non-end segment on an input feature which is adjacent to $\bar{s}$ (since $\bar{s}$ is a 1-feature size witness for s). In this situation, no such p exists on the input feature containing s. Thus s is a 1-feature size 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.

- $|s| > |q - \bar{q}|$ following from the assumption that s fails the inductive hypothesis.
- $|q - \bar{q}| > |q - p|$ follows from the construction of p and the Delaunay property.
- $|q - p| > |s_p|$ is a result of Proposition 4.
- $|\bar{s}| \geq \frac{|s|}{2}$ is a result of the split size property before $\bar{q}$ is inserted in the mesh.



(a) Subcase A

(b) Subcase C

(c) Subcase B

Figure 12: Arguments in Case 2 of Lemma 7

- $|s_p| \geq |\bar{s}|$ follows from the split size property at the time when p was inserted in the mesh.

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

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

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

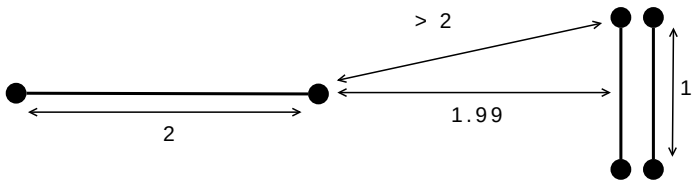
Thus, $|\bar{x} - \hat{q}| > \frac{s}{\sqrt{2}} > |q - \bar{x}|$. See Figure 12(a). Then $|\bar{s}|$ can be estimated using the Pythagorean theorem.

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

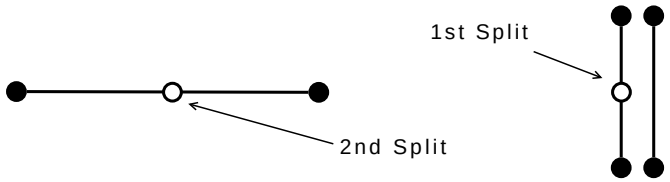
Thus $|\bar{s}| > |q - \bar{q}|$.

Subcase B. Let the nearest point on s to $\bar{s}$, denoted x, be in the 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 $\bar{s}$. This means that $|x - \bar{x}|$ is orthogonal to s and $\bar{s}$ as in Figure 12(c).

Let P be the plane containing $\bar{s}$ which is orthogonal to $|x - \bar{x}|$ and let π denote the projection of points into plane P. Let r be the radius of the disk $D = P \cap B(q, |s|)$ so $r^2 + |x - \bar{x}|^2 = |s|^2$. Observe that $\bar{q} \in D$ and $\hat{q} \notin D$. Finally, observe that for an appropriate point p to exist, $(x - q) \cdot (\bar{q} - \pi(q) < 0$. This is clear if the segment s is a non-end segment since p cannot lie in the interior of the segment s. In the case of an end segment, q' is an input point and the distance 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.



(a) Initial Mesh



(b) Refined Mesh

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

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 $x-\bar{x}$. See Figure 12(c).

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

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

Subcase C. Suppose that q' is the nearest point on s to $\bar{s}$ as depicted in Figure 12(b). By the original relabeling of q, we can conclude the distance from each of the end point of $\bar{s}$ to q' is at least $|s|$. Since the minimum distance from q' to $\bar{s}$ is less than $\frac{|s|}{\sqrt{2}}$, conclude that $|\bar{s}| > |s|$. Using a previous inequality, this implies that $|\bar{s}| > |q-\bar{q}|$. Note that $|q'-\bar{q}| > |s|$ because, if not, q' would be relabeled as q at the beginning of this Case 2.

The inequality $|\bar{s}| > |q-\bar{q}|$ holds in each of the three cases, and thus a contradiction has been reached in each case. Conclude that the inductive hypothesis does hold and the lemma follows. □

The estimates in the Lemma 11 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.

The proof of the theorem in this step relies on an inductive hypothesis which implies that the longest segment s such that $|s| > \sqrt{2}\mathrm{fs}_1(s)$ is always on the queue. This implies that if s is such that $|s| > \sqrt{2}\mathrm{fs}_1(s)$ and s is not on the queue, then at all previous steps, any segment split had length of at least $|s|$. Naturally, the proofs would be much simpler if it could be shown that the length of segments being split formed a nonincreasing 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 points 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).

Still, 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, $|s| < \sqrt{2}\mathrm{fs}_1(s)$. The desired estimate had held from the start of the argument.

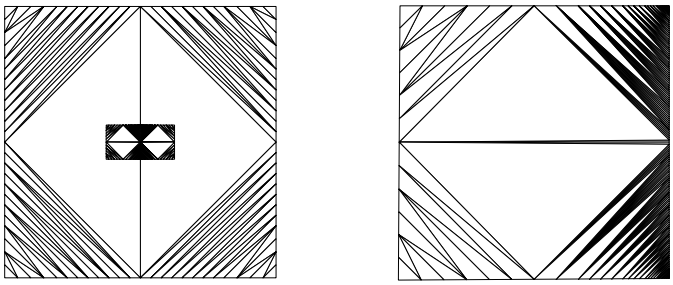


Figure 14: Example mesh following Step 2a.

Step 2a

This step is a simple operation: split all segments into fourths. This is needed as the feature size bound on the segment lengths found in the previous section are not quite strong enough for the algorithm in the next step. Figure 14 shows the result of this step on our initial example.

This split strengthens to the bound determined in Step 1b which will be needed in the analysis of Step 2b.

Lemma 8. *Following this step, for any segment in the mesh,*

$$|s| \leq \frac{1}{2\sqrt{2}}\mathrm{lfs}_1(s)$$

Proof. Following Step 1b, $|s'| \leq \sqrt{2}\mathrm{fs}_1(s') \leq \sqrt{2}\mathrm{lfs}_1(s')$, for all segments s' at the start of this stage. If s is a subsegment of s', then $\mathrm{lfs}_i(s) \leq \mathrm{lfs}_i(s')$. Now let s be one of the four segments created during this step from segment s'.

$$\begin{aligned} |s| &= \frac{1}{4}|s'| \\ &\leq \frac{\sqrt{2}}{4}\mathrm{lfs}_1(s') \\ &\leq \frac{1}{2\sqrt{2}}\mathrm{lfs}_1(s) \end{aligned}$$

□

Note that the estimate $|s| \leq \frac{1}{2\sqrt{2}}\mathrm{fs}_1(s)$ does *not* hold on segments in the mesh produced during this step. This is due to the fact that 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 .

The next lemma follows immediately from Lemma 6

Lemma 9. *Following this step, for any segment s in the mesh,*

$$\frac{1}{16}\mathrm{fs}_1(s) \leq |s|.$$

Step 2b

In this step, segments and triangles (in the current Delaunay triangulation of the faces) are split to get estimate the local feature size on the segments.

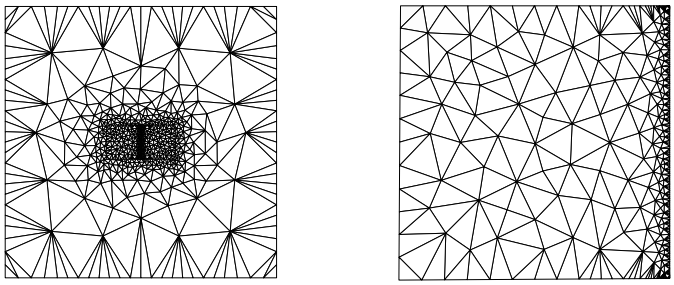


Figure 15: Example mesh following Step 2b.

Action	Insert the circumcenter of a proposed 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 end point of a segment s in face f and $\|c-q\| < \|s\|$.

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

First, the lower bound on segment length is shown.

Lemma 10. *Throughout Step 2b, the following estimate holds for any segment s.*

$$\min\left(\frac{1}{16}\mathrm{fs}_1(s), \frac{1}{4}\mathrm{lfs}(s)\right) \leq |s|$$

Proof. This lemma is shown by induction. Lemma 9 implies that $|s| \geq \frac{1}{16}\mathrm{fs}_1(s)$ and thus the base case holds. It must be shown that any new segment s which is the result of a split 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 6 imples that $\frac{1}{4}\mathrm{fs}_1(s) \leq |s|$. In the second case, a very similar argument yields $\frac{1}{4}\mathrm{lfs}(s) \leq |s|$. □

The proof that segment lengths will bound local feature size from below requires a number of geometric facts. These are stated first.

Proposition 5. *Let t be a triangle, and let $x \in t$. Then there is a vertex of t, q_t such that $|x - q_t| \leq R_t$.*

Proof. Let c_t be the circumcenter of triangle t. Observe that t is covered by the three diametral balls between each vertex and the circumcenter. See Figure 16. □

The next proposition ensures that if the nearest point on a face to a segment is in the interior of a face, then the nearest point on the segment to that face occurs as an end point of the segment.

Proposition 6. *Let s be a segment in a PLC. Then one of the following two holds.*

- *There exists $\bar{s}$ on an input feature disjoint from s such that* $\mathrm{dist}(s, \bar{s}) = \mathrm{lfs}(s)$.

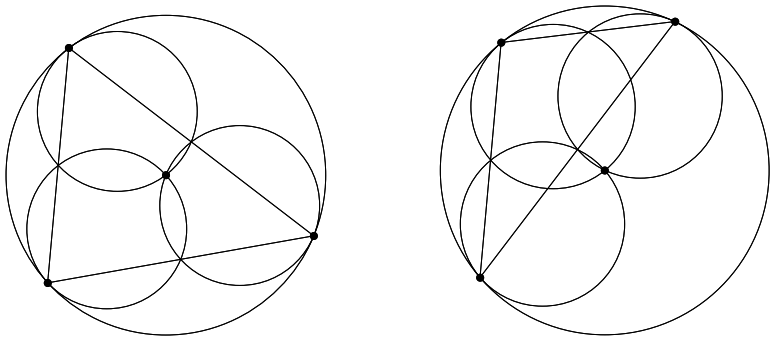
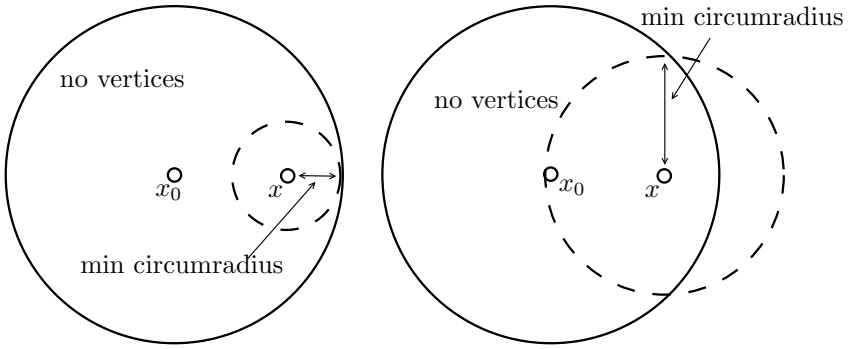


Figure 16: Given any triangle, the three diametral balls between vertices and the circumcenter cover the triangle.



(a) If point x is *contained in the circumcircle* of (Delaunay) triangle t lies in a circle of radius r which contains to vertices, then the circumradius of t is bounded below by the distance from x to the boundary of the empty ball.

(b) If point x is *contained in* (Delaunay) triangle t lies in a circle of radius R, then the lower bound on the circumradius of t corresponds to the radius of a larger circle.

Figure 17: Diagram for Proposition 7

- *There is some endpoint q of s and x in the interior of a disjoint face such that* $\text{dist}(q, x) = \text{lfs}(s)$. *Moreover, in this case $|q - x|$ is orthogonal to the face containing x.*

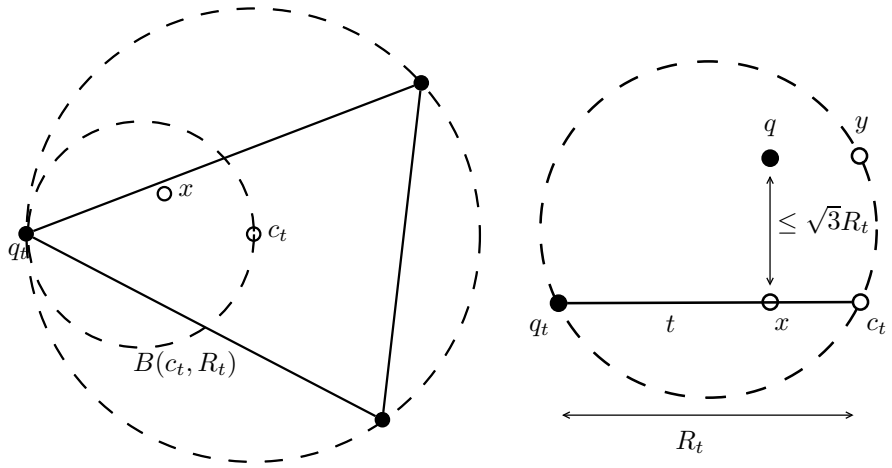
The next proposition asserts a minimum circumradius on the Delaunay triangle containing a point x given that x belongs to an empty disk in the face. See Figure 17 for the associated diagram.

Proposition 7. *Consider a set of vertices $\mathcal{P}$ in a plane. 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 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 17.

Proof. Consider $B(x, R')$ where $R' < \sqrt{R^2 - |x - x_0|^2}$. Let $\{p_1, p_2\} = \partial B(x, R') \cap \partial B(x_0, R)$. Note, if this intersection is empty or just one single point, then $B(x, R') \subset B(x_0, R)$, which means that no t exists with this $B(x, R')$ as its circumcircle (as $B(x, R)$ contains no vertices). Now, $|p_1 - p_2| \leq 2R'$. Let L be the line



(a) Top view (b) Side view

Figure 18: Triangle t in Proposition 8

through p_1 and p_2. Note that x is on the opposite side of L from all of the points of $B(x, R') \setminus B(x_0, R)$. Since triangles are convex and $B(x_0, R)$ cannot contain any vertices, this means that x does not belong to any triangle t with circumcircle $B(x, R')$. □

Suppose that a vertex q is near a face f in some sense. Letting x be the projection of q onto the plane containing f, the next lemma ensures that if the triangle t (in the Delaunay triangulation of f) containing x is large enough, then one of the vertices of t has a nearby Delaunay neighbor (in the three dimensional mesh) which is not in the face. In the algorithm, this will ensure that t was placed on the queue.

Proposition 8. *Let t be a Delaunay triangle in a face f. Let q be a point which is not in the face 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 with 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 18. Let $y = c_t + q - x$. Then observe the following properties.

- $B(\overline{q_t y}) \cap f$ 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 by the Delaunay property of t in f. □

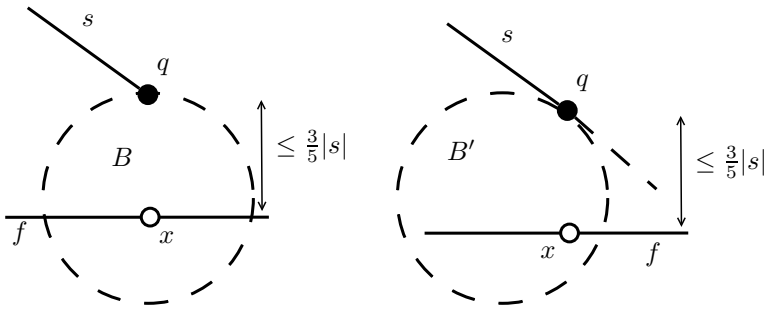
With these geometric facts in place, we seek the bounds in Theorem 5, which are given in two lemmas.

Lemma 11. *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.



(a) Case 1 (b) Case 2

Figure 19: Delaunay neighbors to point q are considered in different balls in the the two different cases.

First, it is clear that this inductive hypothesis is sufficient to ensure that the bound holds when the algorithm terminates: if the desired bound fails, the inductive hypothesis ensures that there is still a triangle on the queue which will be split and the algorithm has not reached termination.

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.

As edges have already been isolated from each other, we prove that the witness of the local feature size of s must be a face. Following Step 2a, $|s| \leq \frac{\text{lfs}_1 s}{2\sqrt{2}}$. Since splitting a segment decreases its length and cannot increase its local feature size, this bound will hold throughout the step. This ensures that no segment or input point can be the witness to the local feature size of s. Thus, there must be some face 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 π 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, ∂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 point $p \in s'$,

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

Thus $\pi(q') \in f$. This means that it is reasonable to replace s and q with s' and q'. 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 point. 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 19. 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 point and the nearest point on any segment containing q to face f, this means that B must be empty.

This means that x belongs to some triangle in f with circumradius of at least $\sqrt{\frac{2}{3}}|q - x|$ as in Figure 20. By Proposition 8, this ensures that a vertex p of t must have a Delaunay neighbor which is not in the face at distance of at most $\sqrt{\frac{5}{3}}|q - x| < 2R_t$ This ensures that t has been queued at some point.

Case 2. Suppose that q is not an input point. Let q_0 be an input point 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 an end segment which existed at the end of Step 0, then this holds because s must be produced by a sequence of

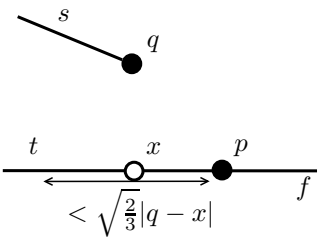
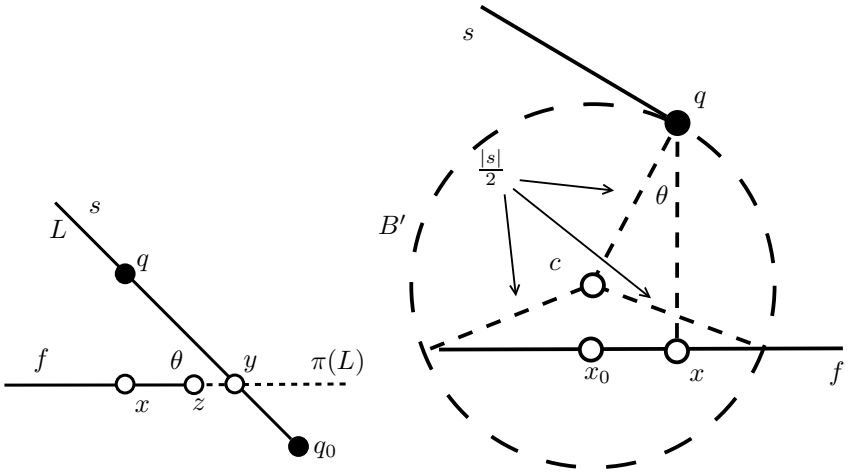


Figure 20: Diagram for Case 1.



(a) The angle between the input segment containing s and f cannot be large.

(b) Estimating $|x - x_0|$.

Figure 21: Diagrams for Case 2.

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 θ between L and $\pi(L)$ as in Figure 21(a). Since q is interior to an input segment, claim that $\sin(\theta) \leq \frac{3}{5}$. Let $y = L \cap P$. If $\sin(\theta) \geq \frac{3}{5}$ 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' the be a ball of radius $\frac{|s|}{2}$ which has a diameter with one end point at q and this diameter intersects $\pi(L)$ as in Figure 19. 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 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 point which is an endpoint of s. Observe that B' is below the cone formed by rotating L around the line containing q_0 and $\pi(q_0)$. Since q is the nearest point 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 point 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 $B' \cap P$ is empty in f. Let c be the center of B' and let $x_0 = \pi(c)$. As seen in Figure 21(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 the face containing t and thus t has been queued.

In both cases, 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 the face containing t. Since face 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. The following estimate on the distance between q and $\hat{q}$ then holds.

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

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.

By the condition following Step 2a which is maintained by the algorithm, we expect that $\text{dist}(s,\hat{s}) \geq 2\sqrt{2}|\hat{s}|$. This inequality contradicts the previous bound as $\sqrt{7} < 2\sqrt{2} = \sqrt{8}$.

Conclude that the inductive hypothesis holds and thus the upper bound on segment lengths holds. □

Lemma 12. *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 in the mesh. Any end segment must satisfy this bound as $|s| \leq \frac{1}{2\sqrt{2}}\text{fs}_1$ following Step 2a and splitting 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 7. In the base case, any segment which fails he bound must be queued since adjacent segments have the same length and thus no points 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 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 $|s|$.

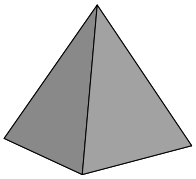


Figure 22: Initial PLC input and final refined mesh for the pyramid example.

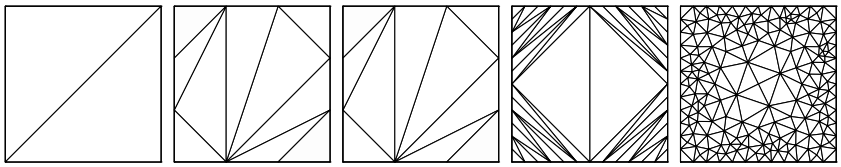


Figure 23: Base of the pyramid following steps 0, 1a, 1b, 2a, and 2b.

6 Examples

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

Example 1. The first example is a square pyramid shown 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 triangular sides is given in Figure 24.

Example 2. This example consists of a wheel of 20 faces which lies slightly above a disjoint square as depicted in Figure 25. The mesh of the square base produced following each step of the algorithm can be seen in Figure 26. Similar output for one of the rectangular "spokes" of the wheel is given in Figure 27. Note that the algorithm still terminates even in the presence of acute angles in the input. The number of vertices in the mesh 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 28. The refinement of one of these faces is shown in Figure 29.

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 4 as each segment is guaranteed to have length of at least $\frac{1}{4}\text{lfs}(s)$ or $\frac{1}{16}\text{fs}_1(s)$. A 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 11 (and thus Theorem 5) 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.

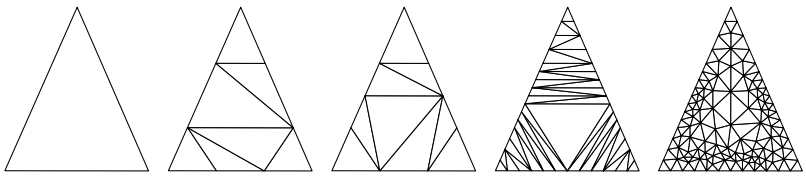


Figure 24: Side of the pyramid following steps 0, 1a, 1b, 2a, and 2b.

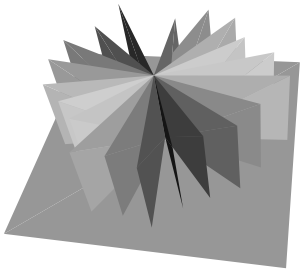


Figure 25: Wheel example input.

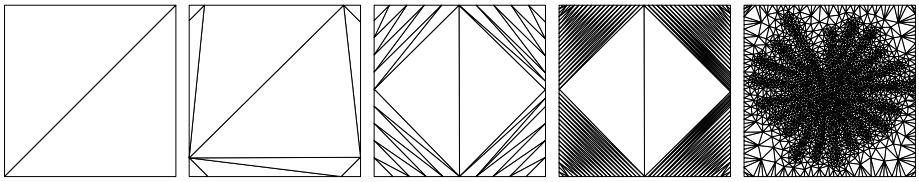


Figure 26: Base plane of the wheel example following steps 0, 1a, 1b, 2a, and 2b.

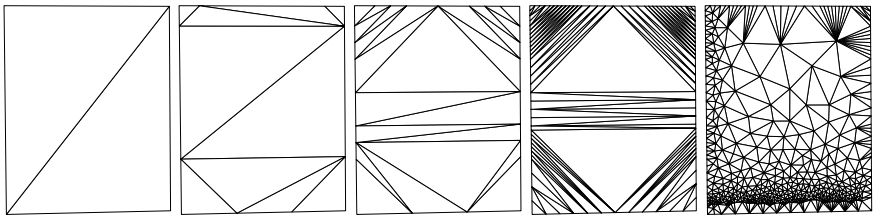


Figure 27: One "spoke" in the wheel example following steps 0, 1a, 1b, 2a, and 2b. The center of the wheel is at the bottom while the disjoint square is to the left of this face.

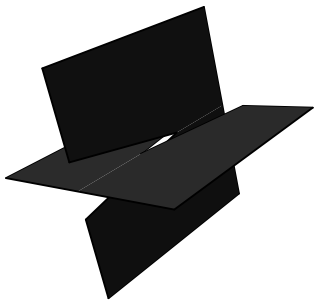


Figure 28: Example containing non-convex input faces.

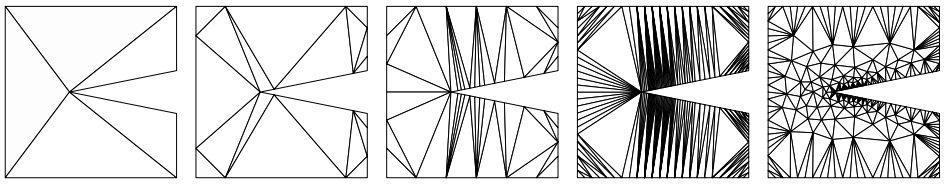


Figure 29: One of the faces in Example 3 following steps 0, 1a, 1b, 2a, and 2b.

Table 1: Number of points in the mesh following each step of the algorithm for the wheel example.

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

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.

References

[1] U. A. Acar, B. Hudson, G. L. Miller and T. Phillips, SVR: Practical engineering for a fast 3D meshing algorithm, *Proc. 16th Int. Meshing Roundtable*, 2007, pp. 45–62.

[2] A. Bowyer, Computing Dirichlet tessellations, *Comput. J.* **24** (1981) 162–166.

[3] S.-W. Cheng, T. K. Dey and J. A. Levine, A practical Delaunay meshing algorithm for a large class of domains, *Proc. 16th Int. Meshing Roundtable*, 2007, pp. 477–494.

[4] S.-W. Cheng, T. K. Dey and E. A. Ramos, Delaunay refinement for piecewise smooth complexes, *Proc. 18th ACM-SIAM Symp. Discrete algorithms*, 2007, pp. 1096–1105.

[5] S.-W. Cheng and S.-H. Poon, Three-dimensional Delaunay mesh generation, *Discrete Comput. Geom.* **36** (2006) 419–456.

[6] A. Chernikov and N. Chrisochoides, Three-dimensional semi-generalized point placement method for Delaunay mesh refinement, *Proc. 16th Int. Meshing Roundtable*, 2007, pp. 25–44.

[7] L. P. Chew, Guaranteed-quality mesh generation for curved surfaces, *Proc. 9th ACM Symp. Comput. Geom.*, 1993, pp. 274–280.

[8] D. Cohen-Steiner, and É. Colin de Verdière and M. Yvinec, Conforming Delaunay triangulations in 3D, *Comput. Geom. Theory Appl.* **28** (2004) 217–233.

[9] T. K. Dey and J. A. Levine, Delaunay meshing of piecewise smooth complexes without expensive predicates, OSU-CISRC-7108-TR40, Computer Science and Engineering Research Center, Ohio State Univ., 2008.

[10] B. Hudson, G. L. Miller and T. Phillips, Sparse Voronoi refinement, *Proc. 15th Int. Meshing Roundtable*, 2006, pp. 339-358.

[11] R. Jampani and A. Üngör, Construction of sparse well-spaced point sets for quality tetrahedralizations, *Proc. 16th Int. Meshing Roundtable*, 2007, pp. 63–80.

[12] G. L. Miller, A time efficient Delaunay refinement algorithm, it Proc. 15th ACM-SIAM Symp. Distrete Algorithms, 2004, pp. 400–409.

[13] G. L. Miller, S. E. Pav and N. J. Walkington, Fully incremental 3D Delaunay refinement mesh generation, *Proc. 11th Int. Meshing Roundtable*, 2002, pp. 75–86.

[14] G. L. Miller, S. E. Pav and N. J. Walkington, Where and why Ruppert's algorithm works, *Proc. 12th Int. Meshing Roundtable*, 2003, pp. 91–102.

[15] M. Murphy, D. M. Mount and C. W. Gable, A point-placement strategy for conforming Delaunay tetrahedralization, *Int. J. Comput. Geom. Appl.* **11** (2001) 669–682.

[16] S. E. Pav and N. J. Walkington, Robust three dimensional Delaunay refinement, *Proc. 13th Int. Meshing Roundtable*, 2004, pp. 145–156.

[17] A. Rand and N. Walkington, 3D Delaunay refinement of sharp domains without a local feature size oracle, *Proc. 17th Int. Meshing Roundtable*, 2008, pp. 37–54.

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

[19] H. Si, On refinement of constrained Delaunay tetrahedralizations. *Proc. 15th Int. Meshing Roundtable*, 2006, pp. 509–528.

[20] H. Si and K. Gartner, Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations. *Proc. 14th Int. Meshing Roundtable*, 2005, pp. 147–164.

[21] A. Üngör, Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations, *Proc. 6th Latin American Symp. Theoretical Informatics*, 2004, pp. 152–161.

[22] D. F. Watson, Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes, *Comput. J.* **24** (1981) 167–171.