

DELAUNAY REFINEMENT ALGORITHMS FOR ESTIMATING LOCAL FEATURE SIZE IN 2D AND 3D

ALEXANDER RAND

*Institute for Computational Engineering and Sciences, The University of Texas at Austin
Austin, TX 78712, USA
arand@ices.utexas.edu*

NOEL WALKINGTON

*Department of Mathematical Sciences, Carnegie Mellon University
Pittsburgh, PA 15213, USA
noelw@andrew.cmu.edu*

Received 29 October 2008

Revised 30 April 2010

Communicated by Siu-Wing Cheng

ABSTRACT

We present Delaunay refinement algorithms for estimating local feature size 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 practical to implement and several examples are given.

Keywords: Delaunay refinement; mesh generation.

1. Introduction

The prototypical Delaunay refinement algorithm given by Ruppert¹ 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. 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.

Extending 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^{2,3} and thus require the local feature size of the PLC be given to the algorithm as input, rather than be implicitly computed by the algorithm as was done by Ruppert's algorithm.⁴ The subsequent algorithm of Pav and Walkington⁵ (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. The algorithm possesses 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, unlike many algorithms for conforming Delaunay tetrahedralization, including PW3D, our algorithm has been implemented.

Two other 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 algorithm of Si and Gartner^{6,7} which computes a constrained Delaunay tetrahedralization. Local feature size is estimated at the input vertices and 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 local information but involves a user-supplied sizing parameter. Our algorithm has been implemented and used as part of a quality mesh generator;^{11,12} this paper provides 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.

Theorem 1. *For any non-acute input PLC, Algorithm 1 terminates. Following the termination of Algorithm 1, $\frac{1}{2} \text{lfs}(q_0) \leq N(q_0) \leq \sqrt{2} \text{lfs}(q_0)$ for any input vertex 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 Sec. 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, an estimate of this type is needed to form 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 specialized in the later sections is described in Sec. 3. Theorem 1 will be duplicated in Sec. 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 Sec. 5. Examples of the 3D algorithm are given in Sec. 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.

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} .
- The **spindle** of a segment s in \mathcal{S}' , denoted $\text{Spind}(s)$, is the set of segments 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. The diametral ball of the segment \overline{ab} , denoted $B(\overline{ab})$, is the smallest ball containing the interior of this segment. The circumball of the triangle with vertices a , b , and c is the smallest ball containing a , b and c on its boundary.

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}' . 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 since 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, a few conventions will be followed.

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(a) 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

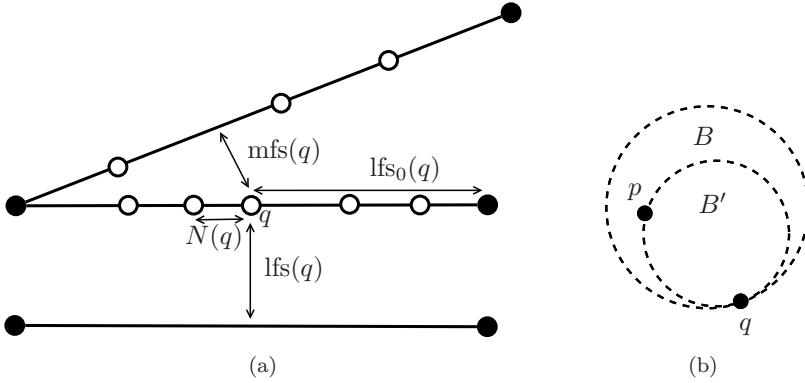


Fig. 1. Preliminary figures. (a) Example of sizing functions in Definition 4 for a 2D PLC. The black dots represent input vertices while the white dots represent vertices inserted during the refinement. (b) Diagram for the proof of Proposition 2.

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

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.

$$fs_i(s) = \begin{cases} lfs_i(s) & \text{if } s \text{ is an end segment,} \\ mfs_i(s) & \text{if } s \text{ is a non-end segment.} \end{cases}$$

Given segment *s* in *S'*, point *x* is called an *i*-feature size witness for *s* if *x* is contained in a feature of *C* of dimension at most *i* which is disjoint from *s*. Given simplex *s* in *C'*, point *x* is called a local feature size witness for *s* if *x* is contained in a feature of *C* which is disjoint from the feature of *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.¹³ 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 $fs_i(s) \leq \text{dist}(x, s)$. Proposition 1 below gives some closely related estimates for *i*-local feature size and *i*-feature size of subsegments.

Proposition 1. Let *s* and *s'* be segments with $s' \subset s$. Then $lfs_i(s) \leq lfs_i(s')$. If *s'* is a non-end segment or *s* and *s'* are both end segments containing at most one input vertex, then $fs_i(s) \leq fs_i(s')$.

Proof. Since s and s' belong to the same input feature $\text{lfs}_i(s) \leq \text{lfs}_i(s')$ follows from the definition. If s' is a non-end segment or s and s' are both end segments containing at most one input vertex then any input feature intersecting s must also intersect s' . Then $\text{fs}_i(s) \leq \text{fs}_i(s')$ immediately follows. \square

Additionally the following form of the Delaunay property will be used often.

Proposition 2. [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 in B .

Proof. 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 1(b). 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 $\overline{B'} \setminus B = q$, we conclude that $p' \in B$. \square

3. Generic Delaunay Refinement Algorithm

Each Delaunay refinement algorithm considered will have 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.

Unacceptability	Which simplices are unacceptable?
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 the queue be processed?
Safety	Which simplices are safe to split?

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

Algorithm 3 Ruppert’s Algorithm - conformity only

Unacceptability	A segment is unacceptable if its diametral disk is nonempty.
Action	Insert the midpoint of the segment.
Priority	Process segments in any order.
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 does not fit these two requirements is not.

Some of these specifications for Ruppert’s algorithm are so simple that they can be easily overlooked. However, for different purposes it is important to generalize Delaunay refinement algorithms in each of the four ways considered above. We briefly discuss the role of each of these items in improving and extending the simplest Delaunay refinement algorithms.

Unacceptability There are two typical kinds 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 altogether.^{14,15}

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.^{9,16}

Action The general action for removing an unacceptable simplex involves inserting a vertex inside the circumball of the simplex to ensure that such a simplex no longer exists in the Delaunay triangulation. Chew’s first Delaunay refinement algorithm¹⁶ used the insertion of circumcenters to remove poor quality triangles from the mesh. The circumcenter is a natural choice since it 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^{17–19} 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.¹⁴

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.²⁰

Priority The priority queue for most Delaunay refinement algorithms involves prioritizing lower dimensional simplices before higher dimensional ones.^{1,21} For time-efficient algorithms this priority queue must be modified,^{20,22,23} typically requiring simplices queued due to quality to be processed before those queued due to encroachment. Prioritizing queued simplices of equal dimension (often by circumradius) has also been used in some algorithms.^{22,24}

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.^{25,26} 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. Such estimates 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,^{25,26} this algorithm is developed as a natural predecessor to the 3D version given in Sec. 5.

Local feature size is estimated at each input vertex in terms of the distance to its nearest Delaunay neighbor in the resulting triangulation. This is divided into three steps as given in Algorithm 4. These steps are labeled according to the highest dimensional features in the input complex which are refined. The earlier steps are used in some sense to isolate input vertices from each other which eliminates difficulties and special cases in Step 1b.

Algorithm 4 Estimate Feature Size 2D

- (Step 0) Compute the Delaunay triangulation of the input vertices.
 - (Step 1a) Split end segments based on 0-local feature size.
 - (Step 1b) Estimate local feature size at all input vertices via Delaunay refinement.
-

(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.

Lemma 1. *Following Step 0, $N(q_0) = \text{lfs}_0(q_0)$ for each input vertex $q_0 \in \mathcal{P}$.*

(Step 1a) Split end segments based on 0-local feature size.

If s is an end segment and q_0 is both an endpoint of s and an input vertex such that $|s| > \text{lfs}_0(q_0)$ then insert the midpoint of s . This is repeated until no such end segments remain. The term $\text{lfs}_0(q_0)$ is known following Step 0 due to Lemma 1. The following ensures the termination of this procedure and estimates the distance to the nearest neighbor of any input vertex.

Lemma 2. *Following Step 1a, $\frac{1}{2} \text{lfs}(q_0) \leq N(q_0, \mathcal{P}')$ for each input vertex $q_0 \in \mathcal{P}$.*

Proof. The desired estimate is shown by induction. The base case results from Lemma 1 since $\text{lfs}(q_0) \leq \text{lfs}_0(q_0)$. Let q be a vertex inserted during this step and let q_0 be an input vertex. The inductive hypothesis ensures the bound for all vertices other than q , so the inductive step holds as long as $\frac{1}{2} \text{lfs}(q_0) < |q - q_0|$. This is now shown in three cases.

Case 1. q lies on an input segment disjoint from q_0 . Then $\text{lfs}(q_0) \leq |q_0 - q|$.

Case 2. q is the midpoint of an input segment s containing q_0 . Then $\text{lfs}(q_0) \leq |s|$ since the other endpoint of s provides a suitable local feature size witness. The result holds since $2|q - q_0| = |s|$.

Case 3. q is the midpoint of an end segment s containing only one input vertex which is q_0 . Then q was inserted because $|s| > \text{lfs}_0(q_0)$ and again the inductive step holds because $2|q - q_0| = |s|$. □

By design, this step ensures the following lemma.

Lemma 3. *Following Step 1a, $|s| \leq \text{lfs}_0(q_0)$ for each end segment $s \in \mathcal{S}'$ containing input vertex $q_0 \in \mathcal{P}'$.*

Further refinement of end segments preserves this estimate which applies to any end segment during the subsequent refinement.

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

Step 1b 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. The unacceptability criteria in the algorithm below has been constructed to eliminate cascading encroachment sequences which prevent the termination of Ruppert’s algorithm with small input angles; it is achieved by simply disallowing such encroachment.

First, we show that the algorithm terminates and that the distance from an input vertex to its nearest neighbor in the resulting mesh provides an appropriate upper bound on the local feature size at the input vertex. This estimate is similar to those shown in Ruppert’s analysis.

Algorithm 5 Estimate Feature Size 2D - Step 1b

Unacceptability	A segment s is unacceptable if (i) $ s \geq 2 \min_{s' \in \text{Spind}(s)} s' $ or (ii) s has an endpoint q with Delaunay neighbor p such that $ q - p < s $ and p is a 1-feature size witness for s .
Action	Insert the midpoint of a segment.
Priority	Segments may be processed in any order.
Safety	It is safe to split any simplex.

Theorem 2. *Algorithm 4 terminates. For any input vertex q_0 ,*

$$\frac{1}{4} \text{lfs}(q_0) \leq N(q_0, \mathcal{P}'). \tag{1}$$

Proof. Let $q_0 \in \mathcal{P}$ be any input vertex. Inequality (1) is shown by induction. Lemma 2 ensures the base case. Let \mathcal{P}' be the vertex set at some point during the refinement, and let vertex q be inserted as the midpoint of segment s . Now, assuming that $\frac{1}{4} \text{lfs}(q_0) \leq N(q_0, \mathcal{P}')$, the inductive step is ensured by showing that $\frac{1}{4} \text{lfs}(q_0) \leq |q_0 - q|$ in three cases.

Case 1. Segment s lies on an input segment which is not incident to q_0 . Then $\text{lfs}(q_0) \leq |q_0 - q|$.

Case 2. Segment s lies on an input segment incident to q_0 but s is not incident to q_0 . Then q cannot be the nearest neighbor to q_0 since one endpoint of s must be closer. Thus, $|q - q_0| \geq N(q_0, \mathcal{P}') \geq \frac{1}{4} \text{lfs}(q_0)$.

Case 3. Segment s is incident to q_0 . If s was queued based on (i) in the unacceptability rule, i.e., $|s|$ is at least twice the length of the shortest segment in $\text{Spind}(s)$,

$$\frac{1}{4} \text{lfs}(q_0) \leq N(q_0, \mathcal{P}') \leq \min_{s' \in \text{Spind}(s)} |s'| \leq \frac{|s|}{2} = |q_0 - q|.$$

If s was queued based on (ii) let the vertex p be the 1-feature size witness causing s to be queued and let q' be the other endpoint of s . See Fig. 2(a). Since s is an end segment, p does not lie on an input segment containing q_0 . Thus $\text{lfs}(q_0) \leq |q_0 - p| \leq |s| + \min(|q_0 - p|, |q' - p|) \leq 2|s| \leq 4|q_0 - q|$.

Inequality (1) is the key to ensuring the termination of the algorithm. This ensures that vertices do not accumulate near input vertices. Vertices inserted away from input vertices result from unacceptability rule (ii) and thus have insertion radii (the distance to the nearest neighbor at time of insertion) bounded below by 1-feature size. Since mesh feature size is only zero at input vertices, vertices cannot accumulate anywhere and the algorithm terminates. □

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

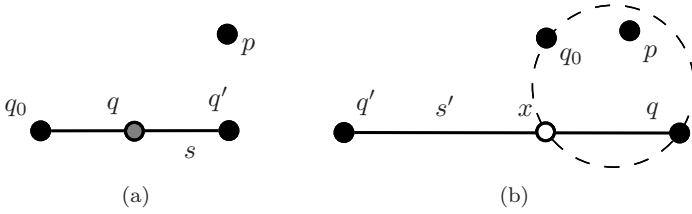


Fig. 2. Diagrams for the proofs of Theorems 2 and 3. (a) Theorem 2, Case 3. (b) Theorem 3, Case 3.

Theorem 3. *Following the termination of Algorithm 4,*

$$N(q_0, \mathcal{P}') \leq \frac{2}{\sqrt{3}} \text{lhs}(q_0) \tag{2}$$

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

Proof. If $N(q_0, \mathcal{P}) = \text{lhs}(q_0)$ (i.e. the local feature size at q_0 is realized by an input vertex), then (2) follows from $N(q_0, \mathcal{P}') \leq N(q_0, \mathcal{P}) = \text{lhs}(q_0)$.

Otherwise, $\text{lhs}(q_0) = \text{dist}(q_0, s)$ 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 upon termination of the algorithm and let q be the nearest endpoint of s' to q_0 . Also let q' denote the unlabeled endpoint of s' . This situation is depicted in Fig. 2(b).

Assume $N(q_0, \mathcal{P}') > \frac{2}{\sqrt{3}} \text{lhs}(q_0)$. Then s' will be shown to be unacceptable, which will contradict the assumption that the algorithm has terminated and thus imply (2). First, the Pythagorean theorem yields estimates on $|x - q_0|$ and $|q - q_0|$:

$$\frac{4}{3} \text{lhs}(q_0)^2 < N(q_0, \mathcal{P}')^2 \leq |q_0 - q|^2 = \text{lhs}(q_0)^2 + |x - q|^2.$$

We conclude that $|x - q_0| = \text{lhs}(q_0) < \sqrt{3}|x - q|$ and $|q - q_0| \leq 2|x - q|$.

Next, Lemma 3 is used to ensure that s' is not an end segment.

$$\begin{aligned} |q' - q_0|^2 &= |q' - x|^2 + |x - q_0|^2 < |q' - x|^2 + 3|x - q|^2 \\ &\leq |q' - x|^2 + 2|x - q||x - q'| + |x - q|^2 = |q' - q|^2 \end{aligned}$$

Then $|q - q_0| \leq |q' - q_0| < |s'|$ which contradicts Lemma 3 if s' is an end segment.

By the Delaunay property, either q and q_0 are Delaunay neighbors or q must have some Delaunay neighbor in the ball $p \in B(\overline{q_0q})$. Let p denote this neighbor of q (which is possibly q_0). Since $B(\overline{q_0q}) \cap s \subset s'$, p cannot lie on the same input feature as s' and thus p is a local feature size witness for s' . As $|p - q| \leq |q_0 - q| < |s'|$, $|s'|$ is unacceptable and thus the algorithm could not have terminated. \square

The constants in Theorems 2 and 3 are both sharp and independent of the smallest input angle. Note that the constant in Theorem 3 is sharper than that of Theorem 1 for Ruppert’s algorithm in the non-acute case since Algorithm 4 checks all Delaunay neighbors of segment endpoints, a stronger condition than encroachment of the segment.

5. Estimating Feature Size in 3D

The ideas of the previous section can be extended to 3D Delaunay refinement. 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 and Step 1a and Step 2a are simple procedures each requiring a single pass over the Delaunay triangulation. Each of these steps will be described carefully after the main results are given.

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 fs_2 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 2-feature size of that segment. Theorem 4 results from standard techniques used in the analysis of Ruppert's algorithm, while Theorem 5 requires a more in-depth analysis than that of previous algorithms.

Theorem 4. *Throughout Algorithm 6, all segments $s \in \mathcal{S}'$ satisfy*

$$\min \left(\frac{1}{16} fs_1(s), \frac{1}{4} fs_2(s) \right) \leq |s|. \quad (3)$$

Theorem 5. *Following the termination of Algorithm 6,*

$$|s| \leq \min \left(\frac{1}{2\sqrt{2}} fs_1(s), \frac{5}{3} fs_2(s) \right) \quad (4)$$

for all segments $s \in \mathcal{S}'$ in the resulting mesh.

To prove these two theorems output conditions on the PLC are determined following each step of the algorithm. Step 2b will yield a mesh satisfying precisely the conditions in the theorems.

We illustrate this algorithm by considering the results of each step on a simple PLC. The example consists of two disjoint, orthogonal squares contained inside a sufficiently large bounding box as seen in Fig. 3. The side length of each of the squares is 20 times longer than the distance between the squares. At each step of the algorithm the Delaunay triangulation of the vertices of each square will be shown as it is easier to visualize than the full tetrahedralization.

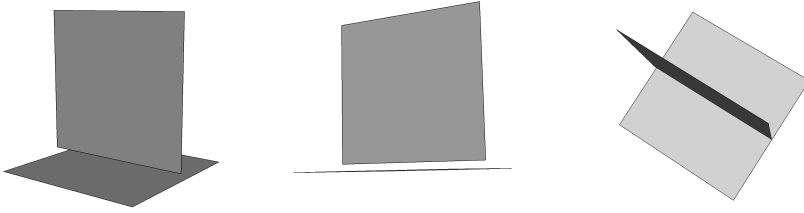


Fig. 3. Three views of an illustrative example consisting of two disjoint squares.

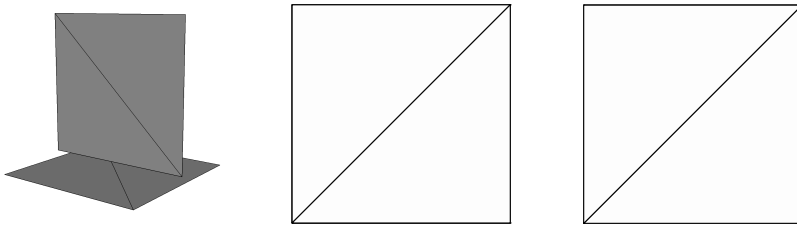


Fig. 4. Example following Step 0 (left) with horizontal square (center) and vertical square (right).

(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 Fig. 4.

Lemma 4. *Following Step 0,*

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

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

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

This step consists of a single pass over 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 Fig. 5. After completing Step 1a, a number of properties hold.

Lemma 5. *Following 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

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

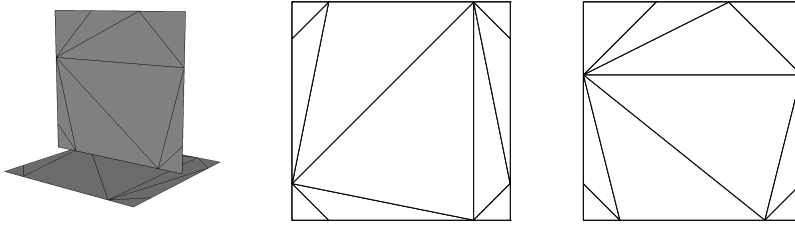


Fig. 5. Example following Step 1a (left) with horizontal square (center) and vertical square (right).

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 fs_1 on all segments via Delaunay refinement.

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

Algorithm 7 Estimate Feature Size 3D - Step 1b

Unacceptability	A segment s is unacceptable if (i) $ s \geq 2 \min_{s' \in \text{Spind}(s)} s' $ or (ii) s has an endpoint q with Delaunay neighbor p such that $ q - p < s $ and p is a 1-feature size witness for s .
Action	Insert the midpoint of a segment.
Priority	Segments are prioritized by length with longest first.
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 has been carefully maintained. Figure 6 shows the running example following Step 1b. The shortest segments occur where the boundaries of the two squares are nearest.

First, we show that the algorithm described terminates and that the length of each segment is bounded below by its 1-feature size. This argument uses the same arguments as the “usual” proofs of termination for typical Delaunay refinement algorithms.

Lemma 6. *Throughout Step 1b, all segments $s \in \mathcal{S}'$ satisfy*

$$\frac{1}{4} fs_1(s) \leq |s|. \tag{5}$$

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

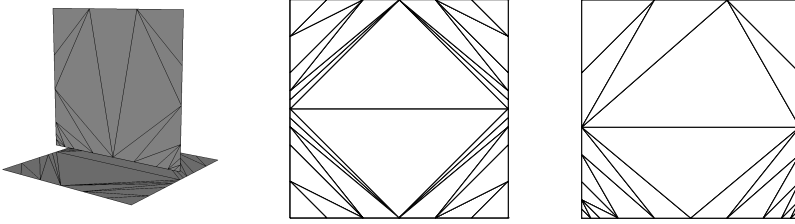


Fig. 6. Example following Step 1b (left) with horizontal square (center) and vertical square (right).

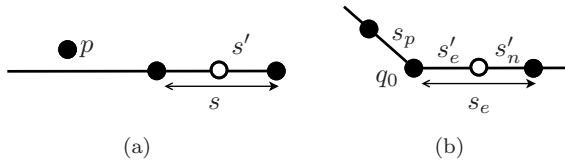


Fig. 7. Two cases in Lemma 6. (a) Case 1. (b) Case 2.

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

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

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), $|s_n| \geq |s_e| \geq \text{fs}_1(s_n)$. Thus the lower bound on segment lengths holds.

The inductive step is shown in two cases based on which unacceptability rule caused a vertex insertion.

Case 1. Consider a vertex inserted as a result of unacceptability rule (ii). Let s be the segment which is split and let p denote the 1-feature size witness causing unacceptability, i.e., $\text{dist}(s, p) \leq |s|$. See Fig. 7(a). Then for either of the new segments created, denoted s' ,

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

since p must be a 1-feature size witness for both new segments.

Case 2. Consider a vertex inserted as a result of unacceptability rule (i). Let s_e be the end segment which is split forming a new end segment s'_e and non-end segment s'_n . Since all adjacent end segments begin with equal lengths there must be some segment, denoted s_p , in $\text{Spind}(s_e)$ which was formed as a result of unacceptability rule (i) and such that $|s_e| \geq 2|s_p|$. Finally, let q_0 denote the input vertex in s_e ; see Fig. 7(b). Then,

$$\text{fs}_1(s'_e) \leq \text{-lfs}_1(q_0) \leq |s_p| + \text{fs}_1(s_p) \leq 4|s_p| \leq 4|s'_e|,$$

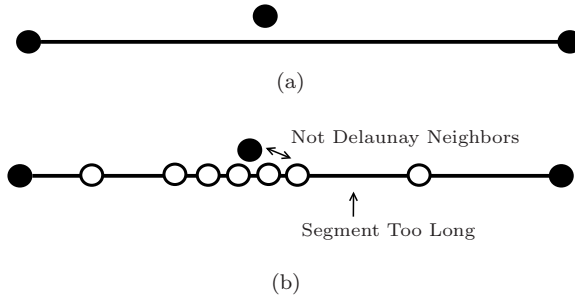


Fig. 8. Lemma 7 does not hold without specifying a refinement order. (a) Possible partial initial mesh. (b) Possible refinement.

where $fs_1(s_p) \leq 3|s_p|$ has been used since Case 1 applies to s_p . Input vertex q_0 is a 1-feature size witness for s'_n , and thus $fs_1(s'_n) \leq \text{dist}(s'_n, q_0) = |s'_n|$.

We conclude that $\frac{1}{4}fs_1(s) \leq |s|$ for all segments created during Step 1b. This estimate ensures termination of the algorithm. \square

Next we seek to bound the length of each segment from *above* in terms of its 1-feature size. In the previous lemma the ordering of the queue of segments is irrelevant. However, in order to get the upper bound, an arbitrary order does not work. To see this consider a mesh including a portion similar to Fig. 8(a). If segments to the left are refined first a situation similar to Fig. 8(b) could arise. Then there is a segment on the right side which is longer than its distance to the input vertex that is not on the segment. This segment may not “see” this nearby vertex on its Delaunay cavity. The segment endpoint necessarily has a Delaunay neighbor which is a 1-feature size witness for the segment; however, the witness may be sufficiently far away as to not cause unacceptability for the segment.

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

Lemma 7. *Following the termination of Step 1b,*

$$|s| \leq \sqrt{2}fs_1(s) \tag{6}$$

for all segments $s \in S'$ in the resulting mesh.

Proof. A segment s will be called **effectively queued** if there exists $s' \in \text{Spind}(s)$ such that $|s'| = |s|$ and s' is on the queue. Since $s \in \text{Spind}(s)$, if s is queued, then s is also effectively queued. The effectively queued definition is also designed to include end segments that will be queued by unacceptability rule (ii) when an queued adjacent end segment is split. Additionally if s is effectively queued then there exists some segment on the queue of length $|s|$.

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 effectively queued and $|s| > \sqrt{2} \text{fs}_1(s)$, 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 $\text{dist}(s, \bar{s}) < \frac{|s|}{\sqrt{2}}$, then $|q - \bar{q}| \geq |s|$ for any q and \bar{q} which are endpoints of s and \bar{s} , respectively.

Split Size Property. If s is a segment such that $|s| > \sqrt{2} \text{fs}_1(s)$ then the queue contains some segment of length at least $|s|$.

Proof of the lemma is divided into the following three claims.

- The split size property \Rightarrow (6) holds upon termination.
- The inductive hypothesis \Rightarrow the split size property.
- The inductive hypothesis holds.

The split size property \Rightarrow (6) holds upon termination. If any segment fails (6), then the split size property implies that the queue is not empty. Since the queue is empty at termination, this claim holds.

The inductive hypothesis \Rightarrow the split size property. Suppose to the contrary that s is a longest segment such that $|s| > \sqrt{2} \text{fs}_1(s)$ and that the longest segment on the queue is shorter than s . Segment s is not effectively queued because that would require a segment of equal length to be on the queue. Then by the first property of the inductive hypothesis, the 1-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 3/Remark 1 in Appendix A, \bar{s} is longer than s (since otherwise the segments would have endpoints that are nearby, which violates the inductive hypothesis). Suppose $|s|$ is a 1-feature size witness for \bar{s} . Then

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

and thus s is not the longest segment failing (6). Otherwise if s is not a 1-feature size witness for \bar{s} , then \bar{s} must be an end segment adjacent to the input feature containing non-end segment s . Let q_0 be the input vertex on \bar{s} , then applying Proposition 4 in Appendix A implies that $|\bar{s}| > \sqrt{2}|s|$ and $|\bar{s}| > \text{dist}(s, q_0)$. The end segment on the input segment containing s has length of at most $\text{dist}(s, q_0)$. Since adjacent end segments begin with equal lengths and are always split in half, the ratio of their lengths must be a power of 2 and thus

$$|\bar{s}| \geq 2 \min_{s' \in \text{Spind}(\bar{s})} |s'|.$$

So \bar{s} is unacceptable. Combined with the inequality $|\bar{s}| > \sqrt{2}|s|$, this contradicts the assumption that s is longer than any queued segment.

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 the input vertex contained in end segment s_e . Lemma 5 ensures that at the end of Step 1a for all vertices \bar{q} 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 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 of an endpoint of s_n ; this results from Proposition 2.

We proceed to the inductive step. The inductive hypothesis will be verified for all segments in two cases: segments which are newly created and segments that existed before the most recent vertex insertion.

Case 1. Consider any *newly* formed segment s and suppose s violates the inductive hypothesis. So $|s| > \sqrt{2} \text{fs}_1(s)$ and s is not on the queue. 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 there is a vertex \bar{q} and endpoint q of s such that $|q - \bar{q}| < |s|$ and \bar{q} is a 1-feature size witness for s . In either case, \bar{q} is a 1-feature size witness for s and the distance between q and \bar{q} is less than $|s|$.

Since s is not effectively queued (and thus not queued), q and \bar{q} cannot be Delaunay neighbors. Now by the Delaunay property (Proposition 2) there is some vertex p in the diametral ball between q and \bar{q} which is a Delaunay neighbor of q . Then $|p - q| < |q - \bar{q}| < |s|$ and p cannot be a 1-feature size witness for s ; if it were, s would be queued.

If s is an end segment then no such p can exist. By unacceptability rule (i) and since segments are prioritized by length, every end segment adjacent to s has length of either $|s|$ or $2|s|$. Vertex p cannot exist on an input segment for which the end segment is of length $2|s|$: it is at least a distance $|s|$ from q . For any end segment of length $|s|$ the adjacent non-end segment is also of length $|s|$ since any further refinement would violate the queue priority. Thus no p exists on an input segment adjacent to s .

If s is a non-end segment we consider the segment \hat{s} which was split forming s . If \bar{q} is a 1-feature size witness for \hat{s} then $\text{fs}_1(\hat{s}) \leq \text{dist}(\hat{s}, \bar{q}) \leq \text{dist}(s, \bar{q}) < \frac{1}{\sqrt{2}}|s| \leq \frac{1}{\sqrt{2}}|\hat{s}|$. Since vertex p is not a 1-feature size witness for s and s is a non-end segment, p must lie on the

Claim A: Let q be a vertex inserted during Step 1b and an endpoint of segments s_1 and s_2 with $|s_1| > |s_2|$. Then if p is a vertex on the input segment containing q such that $|p - q| < |s_1|$, then p was inserted after vertex q .

Proof. When q is inserted, each segment containing q has length at least $|s_1|$. So immediately after the insertion of q , there are no vertices p such that $|p - q| < |s_1|$ on the input segment containing q . □

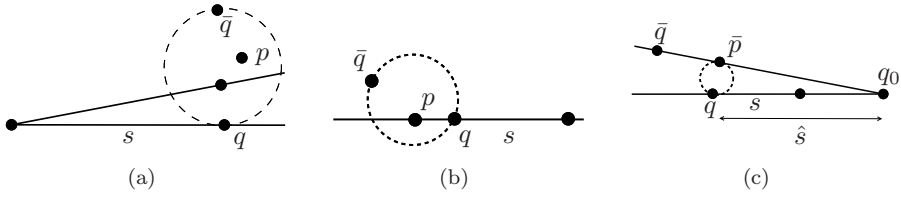


Fig. 9. Diagrams for Lemma 7, Case 1. (a) s is an end segment. (b) s is a “typical” non-end segment. (c) s is a non-end segment and \hat{s} is an end segment.

same input segment as s . Vertex p must have been inserted after q : Claim A (to the right) ensures this for any q inserted during Step 1b and it is immediate for any q inserted during Step 1a. So the segment split when inserting p has length at most $2|p - q| < 2|s| = |\hat{s}|$, violating the split size property and thus the inductive hypothesis; see Fig. 9(b). Otherwise, \bar{q} is not a 1-feature size witness for \hat{s} , which only occurs if \hat{s} is an end segment and \bar{q} lies on an input segment adjacent to \hat{s} ; see Fig. 9(c). Let q_0 be the input vertex which is an endpoint of \hat{s} . Then 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 the 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 which contradicts our assumption that s violates the inductive hypothesis.

Case 2. Consider any segment s which is not newly formed. Again we assume s fails the inductive hypothesis and seek a contradiction. The first criterion of the inductive hypothesis cannot fail as the input vertices and endpoints of s are unchanged by the the most recent vertex insertion (and thus this statement holds by the inductive hypothesis). So the second criterion must fail: s cannot be effectively queued, $|s| > \sqrt{2}fs_1(s)$, and there is a segment \bar{s} which is a 1-feature size witness for s with $\text{dist}(s, \bar{s}) \leq \frac{|s|}{\sqrt{2}}$. Moreover \bar{s} has an endpoint \bar{q} with $|\bar{q} - q| < |s|$ 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 , as follows, if there is a better selection for our purposes.

(Relabel 1) Segment \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 properties.

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

Since s is not on the queue vertices q and \bar{q} cannot be Delaunay neighbors. By the Delaunay property q must have a Delaunay neighbor p in the diametral ball

between q and \bar{q} which is not a 1-feature size witness for s . 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 relabeled segment s must have the same length as the original segment s as otherwise s would be queued. Moreover the relabeled segment s cannot be effectively queued since the original s was not effectively 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 ; see Fig. 10. As p is not a 1-feature size witness for s , p cannot be an input vertex. So p belongs to some segment s_p .

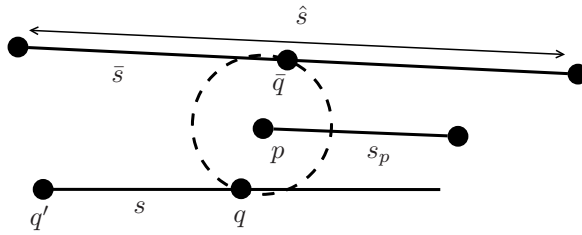


Fig. 10. Segment s fails the inductive hypothesis, \bar{q} is a nearby feature size witness to s , and p is not a 1-feature size witness for s .

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 on the end segment adjacent to q which is in $\text{Spind}(\bar{s})$; see Fig. 11. Thus s is a 1-feature size witness for \bar{s} .

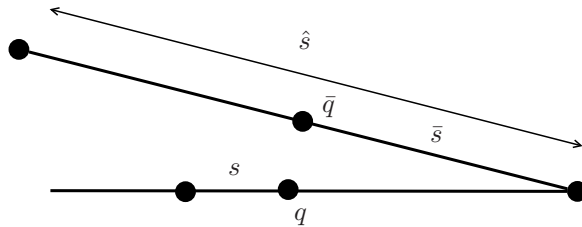


Fig. 11. If s is a non-end segment and \bar{s} is an end segment, p cannot exist as it must lie on the end segment adjacent to q .

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

$$|s| > |q - \bar{q}| > |q - p| > |s_p| \geq |\bar{s}| \geq \frac{|s|}{2}. \tag{7}$$

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.
- (iii) $|q - p| > |s_p|$ is a result of Proposition 5 in Appendix A.
- (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.

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

Subcase A. Suppose that q is the nearest point on s to \bar{s} . Let \bar{x} be the nearest point on \bar{s} to s as in Fig. 12(a). Then

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

Thus $|\bar{x} - \hat{q}| > \frac{s}{\sqrt{2}} > |q - \bar{x}|$. Then $|\bar{s}|$ can be estimated using the Pythagorean theorem: $|\bar{s}|^2 \geq |\bar{q} - \bar{x}|^2 + |\bar{x} - \hat{q}|^2 > |\bar{q} - \bar{x}|^2 + |q - \bar{x}|^2 = |q - \bar{q}|^2$. So $|\bar{s}| \geq |q - \bar{q}|$.

Subcase B. Let the nearest point on s to \bar{s} be in the relative interior of s . Then 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} , respectively, and conclude that $x - \bar{x}$ is orthogonal to both s and \bar{s} . 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 ; see Figs. 12(c) and 12(d). Define disk $D := P \cap B(q, |s|)$ and let r denote the radius of D . Since s is orthogonal to $x - \bar{x}$, $r^2 + |x - \bar{x}|^2 = |s|^2$.

First, q cannot be an input vertex: since s is not queued any vertex which is not a 1-feature size witness for q would be at least a distance $|s|$ away.

Second, $\bar{q} \notin B(q', \frac{3}{2}|s|)$. From (7), $|s_p| > \frac{|s|}{2}$ so $p \notin B(q', \frac{3}{2}|s|)$. If $\bar{q} \in B(q', \frac{3}{2}|s|)$ and s is a non-end segment then p lies on the input segment s_0 containing s . Let B^* denote the diametral ball of q and \bar{q} and it follows that $B^* \cap s_0 \subset B(q', \frac{3}{2}|s|)$; see Fig. 12(e). If s is an end segment the same construction yields an ball empty ball when considering the nearest segment s' to \bar{q} such that $s' \in \text{Spind}(s)$ and $|s| = |s'|$. Since s would then be queued we conclude that $\bar{q} \notin B(q', \frac{3}{2}|s|)$.

Third, let $D_1 := \{x \in D : (\bar{x} - \pi(q)) \cdot (\bar{q} - \pi(q)) < 0\}$. D_1 corresponds to the shaded region in Fig. 12(d). We now verify that $\bar{q} \in D_1$. Define r' such that $P \cap B(q', \frac{3}{2}|s|) = P \cap B(\pi(q'), r')$ and let $\theta = \angle \pi(q')\pi(q)\bar{q}$; see Fig. 12(f). By the law of cosines $|\pi(q') - \bar{q}|^2 = |s|^2 + |\pi(q) - \bar{q}|^2 - 2|s||\pi(q) - \bar{q}| \cos \theta$. Substituting the relations $|\pi(q') - \bar{q}| \geq r'$, $|\pi(q) - \bar{q}| \leq |s|$, $r^2 + |x - \bar{x}|^2 = |s|^2$, and $(r')^2 + |x - \bar{x}|^2 = \frac{9}{4}|s|^2$ yields $2|s||\pi(q) - \bar{q}| \cos \theta \leq -\frac{1}{4}|s|$. So θ is larger than 90 degrees and thus $\bar{q} \in D_1$.

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

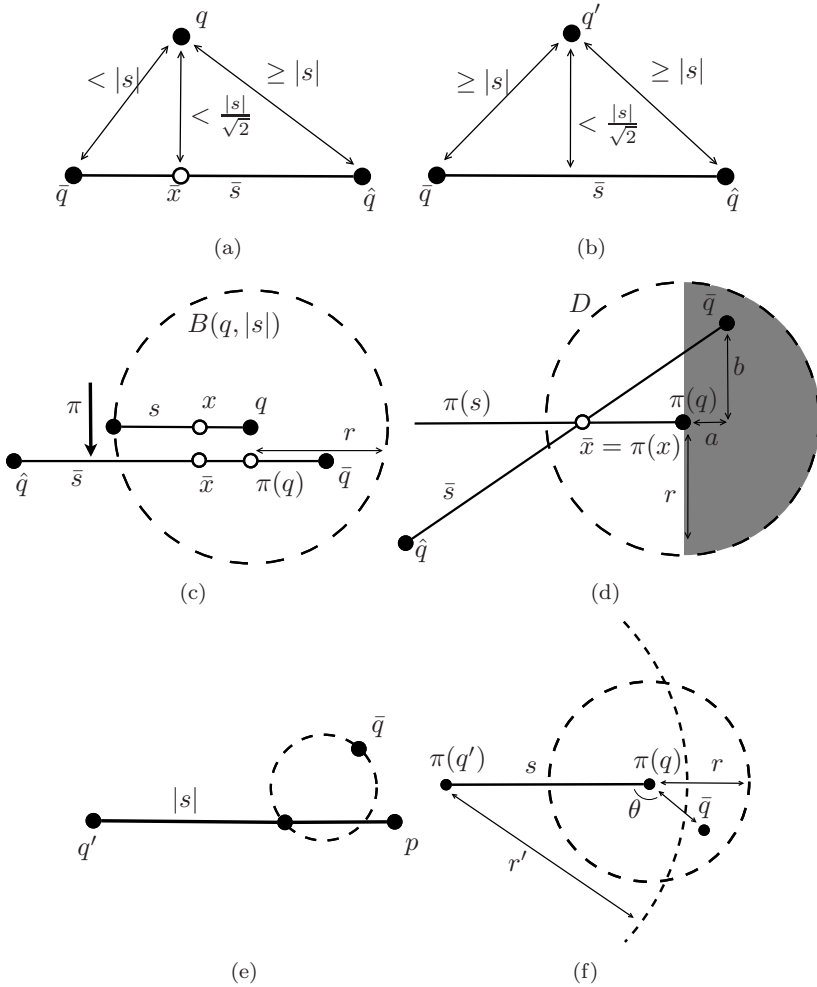


Fig. 12. Diagram for Lemma 7, Case 2. (a) Subcase A. (b) Subcase C. (c) Subcase B: orthogonal to plane P . (d) Subcase B: plane P . (e) Subcase B: $\bar{q} \notin B(q', \frac{3}{2}|s|)$. (f) Subcase B: definitions of r , r' and θ .

seen in Fig. 12(d). The length $|s|$ is bounded by

$$\begin{aligned}
 |\bar{s}|^2 &= |\bar{q} - \hat{q}|^2 = (|\bar{q} - \bar{x}| + |\bar{x} - \hat{q}|)^2 \geq (|\bar{q} - \bar{x}| + r - |\bar{x} - \pi(q)|)^2 \\
 &= |\bar{q} - \bar{x}|^2 + (r - |\bar{x} - \pi(q)|)^2 + 2|\bar{q} - \bar{x}|(r - |\bar{x} - \pi(q)|) \\
 &\geq (a + |\bar{x} - \pi(q)|)^2 + b^2 + (r - |\bar{x} - \pi(q)|)^2 + 2|\bar{x} - \pi(q)|(r - |\bar{x} - \pi(q)|). \quad (8)
 \end{aligned}$$

The final inequality uses $|\bar{x} - \bar{q}| \geq |\bar{x} - \pi(q)|$ since $\pi(q)$ is the nearest point in D_1 to \bar{x} . Expanding all terms of (8) gives

$$|\bar{s}|^2 \geq a^2 + b^2 + r^2 + 2a|\bar{x} - \pi(q)| \geq a^2 + b^2 + r^2.$$

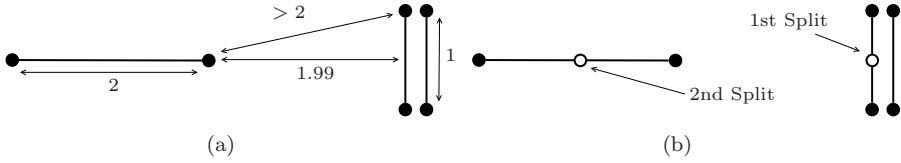


Fig. 13. Sequence of split segment lengths is not monotone. (a) Initial mesh. (b) Refined mesh.

Moreover $r^2 > \frac{|s|^2}{2}$ because $|x - \bar{x}| < \frac{1}{\sqrt{2}}|s|$ and $r^2 = |s|^2 - |x - \bar{x}|^2$. Thus

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

Subcase C. Suppose that q' is the nearest point on s to \bar{s} as depicted in Fig. 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 $|\bar{s}| > |s|$. Combined with inequality (i), this implies that $|\bar{s}| > |q - \bar{q}|$.

The inequality $|\bar{s}| > |q - \bar{q}|$ holds in each of the three subcases, and thus a contradiction has been reached in each subcase which completes the proof.

The estimates in the Lemma 7 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 7 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 Fig. 13(a). Notice that the length-two segment is not queued initially since all vertices are sufficiently far from its endpoints. When the leftmost of the length-one segments is split, its midpoint causes the longer length-two segment to be queued. See Fig. 13(b).

While the lengths of segments which are split increase, this example does not break the inductive hypothesis! It is important to notice that in this case 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.

The goal of this step is to refine the mesh in a way that reduces the constant in (6). All non-end segments will simply be split into fourths. End segments will also be split into fourths but the resulting non-end segments will be refined further according to the following rule.

End Segment Split Rule 1. Split any end segment s into fourths. Then repeatedly insert the midpoint of any newly created non-end segment s' until $|s| \leq \frac{1}{2\sqrt{2}} \text{dist}(s', s_0)$ for any input segment s_0 adjacent to s .

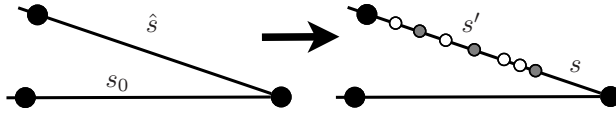


Fig. 14. End segment split operation.

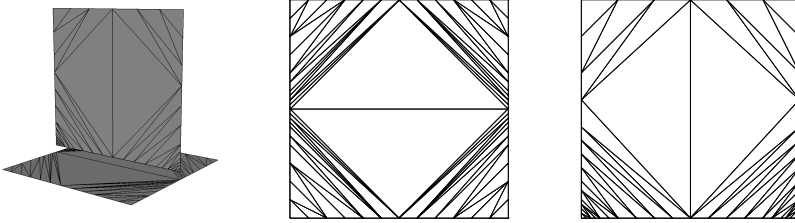


Fig. 15. Example following Step 2a (left) with horizontal square (center) and vertical square (right). For display purposes segments have only been split in half.

This operation is depicted in Fig. 14. We emphasize that this operation is local in the current Delaunay triangulation of the PLC: $\min_{s' \cap s_0 \neq \emptyset, s' \not\subset s_0} \text{dist}(s', s_0)$ can be computed without considering any disjoint features.

Figure 15 shows the result of Step 2a on the example input. For display purposes segments have only been split in half rather than fourths and the extra refinement prescribed by End Segment Split Rule 1 has not been performed. These heuristics have been observed to work in practice and will be further discussed in Sec. 6.

The resulting changes in inequalities (5) and (6) are now given in the following two lemmas.

Lemma 8. *Following Step 2a, all segments $s \in \mathcal{S}'$ satisfy*

$$\frac{1}{16} \text{fs}_1(s) \leq |s|. \tag{9}$$

Proof. Let \hat{s} be a segment existing at the end of Step 1b and let s be a resulting segment after splitting \hat{s} into fourths. (5) implies $\text{fs}_1(s) \leq \text{fs}_1(\hat{s}) \leq 4|\hat{s}| = 16|s|$. If $|s| \geq \frac{1}{2\sqrt{2}} \text{fs}_1(s)$ then let s' be a segment created by splitting s according to End Segment Split Rule 1. Then $\text{fs}_1(s') \leq \text{fs}_1(s) + |s'| \leq (2\sqrt{2} + 1)|s'|$. This argument applies for any further split performed by the End Segment Split Rule 1. \square

Lemma 9. *Following Step 2a, all segments $s \in \mathcal{S}'$ satisfy*

$$|s| \leq \frac{1}{2\sqrt{2}} \text{fs}_1(s). \tag{10}$$

Proof. Let s be a subsegment of \hat{s} resulting from a split into fourths.

Case 1. s and \hat{s} are either both end segments or both non-end segments. Then $fs_1(\hat{s}) \leq fs_1(s)$ since any 1-feature size witness for s is also a 1-feature size witness for \hat{s} . Then $|s| = \frac{1}{4}|\hat{s}| \leq \frac{\sqrt{2}}{4} fs_1(\hat{s}) \leq \frac{1}{2\sqrt{2}} lfs_1(s)$.

Case 2. s is a non-end segment and \hat{s} is an end segment. End Segment Split Rule 1 is constructed so that (10) holds for all these segments. □

(Step 2b) Estimate fs_2 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 2-feature size on the segments. This is performed via a Delaunay refinement algorithm given in Algorithm 8.

Algorithm 8 Estimate Feature Size 3D - Step 2b

Unacceptability	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 in \mathcal{F} containing t . A segment s is unacceptable if (i) $ s \geq 2 \min_{s' \in \text{Spind}(s)} s' $ or (ii) s has an endpoint q with Delaunay neighbor p such that $ q - p < s $ and p is a 2-feature size witness for s .
Action	For an end segment, insert vertices according to End Segment Split Rule 2 below. Otherwise, insert the circumcenter of a segment or triangle.
Priority	Triangles are given highest priority, in any order. Segments are then prioritized by length.
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 $.

End Segment Split Rule 2. Split any end segment s into fourths. Then repeatedly split any newly created non-end segment s' until $|s| \leq \frac{1}{2\sqrt{2}} \text{dist}(s', s_0)$ for any input segment s_0 adjacent to s and $|s| \leq \frac{5}{3} \text{dist}(s', f_0)$ for any input face f_0 intersecting but not containing s .

As in Step 2a non-end segments produced when splitting end segments are immediately refined based on local information in the PLC. The priority rule in Algorithm 8 is atypical: lower 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 Fig. 16. Step 2b is the only step in which vertices are added in faces.

Theorem 4 is now shown using standard ideas from the analysis of Delaunay refinement algorithms.

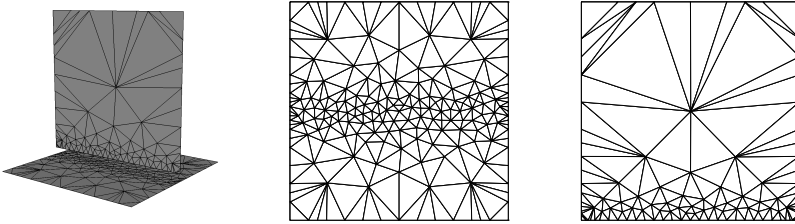


Fig. 16. Example following Step 2b (left) with horizontal square (center) and vertical square (right).

Proof of Theorem 4. The estimate is shown inductively by an argument combining the techniques used in Lemma 6 and Lemma 8.

Base Case. Lemma 8 implies $|s| \geq \frac{1}{16} fs_1(s)$ for all initial segments.

In the inductive step, let s be a segment created during Step 2b after processing queued segment \hat{s} .

Case 1. s and \hat{s} are either both end segments or both non-end segments and \hat{s} was queued based upon unacceptability rule (ii). Unacceptability rule (ii) ensures $fs_2(\hat{s}) \leq |\hat{s}|$ so

$$fs_2(s) \leq fs_2(\hat{s}) + |s| \leq |\hat{s}| + |s| \leq 3|s|.$$

Case 2. s is an end segment and \hat{s} was queued based upon unacceptability rule (i). Some end segment s' adjacent to \hat{s} with $|s'| = |\hat{s}|$ must have been queued previously due to unacceptability rule (ii). Then $fs_2(s) \leq fs_2(s') + |\hat{s}| \leq |s'| + |\hat{s}| \leq 4|s|$.

Case 3. s is a non-end segment and \hat{s} is an end segment. Then s is produced by End Segment Split Rule 2 and thus must result from splitting some non-end segment s' where either $|s'| > \frac{1}{2\sqrt{2}} fs_1(s')$ or $|s'| > \frac{5}{3} fs_2(s')$. In the former case $fs_1(s) \leq fs_1(s') + |s| (\leq 2\sqrt{2} + 1)|s|$ and in the latter case $fs_2(s) \leq fs_2(s') + |s| \leq (\frac{5}{3} + 1)|s|$. Since $2\sqrt{2} + 1 \leq 16$ and $\frac{5}{3} + 1 \leq 4$ the theorem holds. \square

Theorem 5 is an immediate result of the next two lemmas. First, estimate (10) is shown to hold for all segments created during Step 2b.

Lemma 10. *Following the termination of Step 2b,*

$$|s| \leq \frac{1}{2\sqrt{2}} fs_1(s) \tag{11}$$

for all segments $s \in S'$ in the resulting mesh.

Proof. Lemma 9 ensures that (11) holds for all segments which exist at the beginning of Step 2a. When a non-end segment is split or when considering a newly

formed end segment, this estimate is preserved. Non-end segments formed by splitting end segments may have a very small 1-feature size but End Segment Split Rule 2 ensures (11) holds by construction. \square

Lemma 11. *Following the termination of Step 2b,*

$$|s| \leq \frac{5}{3} \text{fs}_2(s) \tag{12}$$

for all segments $s \in \mathcal{S}'$ in the resulting mesh.

Proof. As in Lemma 7 a segment s will be called **effectively queued** if there exists $s' \in \text{Spind}(s)$ such that $|s'| = |s|$ and s' is on the queue. Inequality (12) is shown by induction using the following inductive hypothesis.

Inductive Hypothesis Let s be a segment such that $|s| > \frac{5}{3} \text{fs}_2(s)$. If s effectively queued then there is some triangle t which is on the queue.

If the inductive hypothesis holds then the desired bound holds at termination since whenever the desired bound fails, the queue is not empty. Supposing that s is some segment such that $|s| > \frac{5}{3} \text{fs}_2(s)$ and s is not effectively queued, we will show this implies some triangle must be on the queue and this vertex proposed for insertion when processing this triangle will not be rejected by the safety rule.

Lemma 10 ensures that $|s| \leq \frac{\text{fs}_1(s)}{2\sqrt{2}}$ for all segments s of the mesh during Step 2b. So no segment or input vertex can be the witness to the 2-feature size of s . Thus there must be an input face f such that $\text{dist}(s, f) = \text{fs}_2(s)$. Using Proposition 6 in Appendix A, we conclude that there is some x in a face f and an endpoint q of s such that $\text{fs}_2(s) = |x - q|$, 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 and that $|s'| = |s|$. Note: no $s' \in \text{Spind}(s)$ with $|s'| < |s|$ exists because otherwise s would be queued. The distance from x to the boundary ∂f of f is estimated by

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

So $\text{dist}(x, \partial f) > 2|s|$. Letting q' denote an endpoint of s' ,

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

Thus $\pi(q') \in f$. So s and q can be replaced with s' and q' and the local feature size bound still fails. Thus without loss of generality, s is the nearest segment to P in the set $\{\bar{s} : |\bar{s}| = \min_{s' \in \text{Spind}(s)} |s'|\}$; see Fig. 17(a).

In either of the two possible cases the triangle t in f which contains x has been placed on the queue.

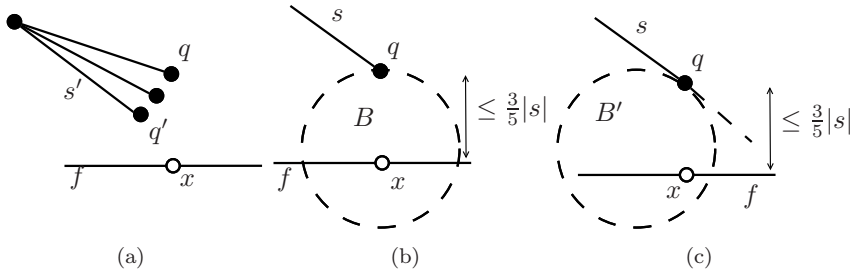


Fig. 17. Selecting q and an associated empty ball B . (a) Relabeling to select the nearest end segment to the face. (b) Empty ball B in Case 1. (c) Empty ball B in Case 2.

Case 1. Suppose that q is an input vertex. Let B be the ball of radius $\frac{|s|}{2}$ with q on the boundary and x on its diameter containing q as in Fig. 17(b). The segment s is not on the queue, so q cannot have any Delaunay neighbors in B which witness the 2-feature size of s . Since q is an input vertex and no end segments containing q have length less than $|s|$, B must be empty.

Proposition 7 in Appendix A implies that x belongs to some triangle in f with circumradius of at least $\sqrt{\frac{2}{3}}|q - x|$ as in Fig. 18(a). Proposition 8 in Appendix A ensures that a vertex p of t must have a Delaunay neighbor $p' \notin f$ such that $|p' - p| < \sqrt{\frac{5}{3}}|q - x| < 2R_t$. Thus t has been queued.

Case 2. Suppose that q is not an input vertex. Let q_0 be an input vertex on the segment containing s . First, $|q_0 - q| \geq |s|$: either s is the segment between q_0 and q or q_0 is a 1-feature size witness for s and (11) holds.

Let θ denote the angle between L and $\pi(L)$; see Fig. 18(b). 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 $|q - y| = \frac{|q-x|}{\sin \theta} \leq \frac{5}{3}fs_2(s) < |s|$ which means that y is contained in the input segment containing s and thus cannot be contained in f . Hence 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 $fs_1(s) < |s|$ violating the bound (11).

Let B' be the ball of radius $\frac{|s|}{2}$ and diameter which is orthogonal to L , has q as an endpoint, and intersects $\pi(L)$; see Fig. 17(c). 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, the assertion 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 . Since $\sin \theta > \frac{3}{5}$ the ball B' is below the cone formed by rotating L around the line containing q_0 and $\pi(q_0)$. Since s was selected to be the nearest segment to P in the set $\{\bar{s} : |\bar{s}| = \min_{s' \in \text{Spind}(s)} |s'|\}$ B cannot contain any vertex on an adjacent segment.

Since s is not queued and any vertex in B' would serve as a 2-feature size witness to cause s to be queued, B' must be empty. Next we seek to apply Proposition 7

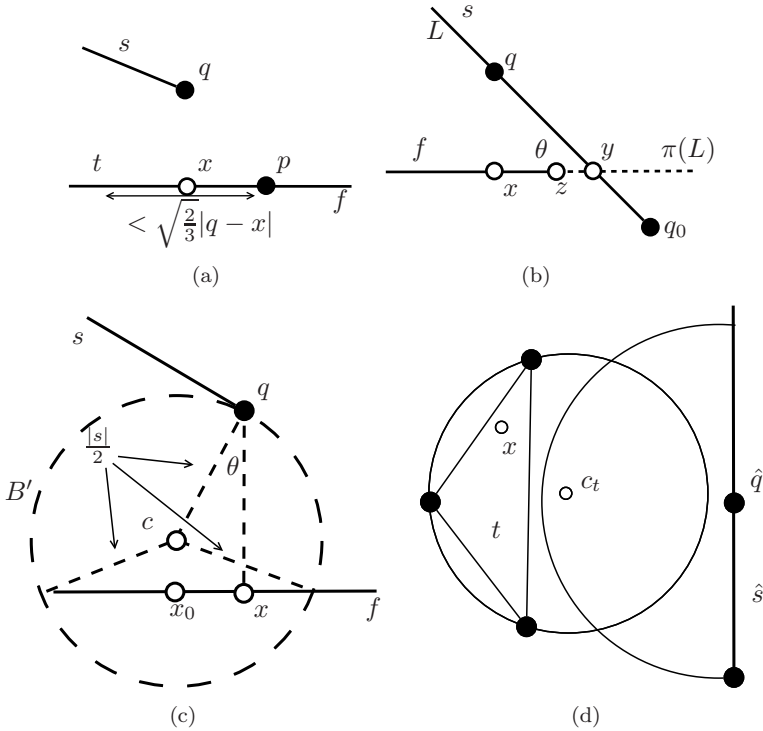


Fig. 18. Diagrams for the proof of Lemma 11. (a) Case 1: Application of Proposition 7 in Appendix A. (b) Case 2: The angle between the input segment containing s and f cannot be large. (c) Case 2: Estimating $|x - x_0|$. (d) The circumcenter of t is rejected by the safety rule.

in Appendix A based on the fact that B' contains no vertex of \mathcal{P} . Let c be the center of B' and let $x_0 = \pi(c)$. As seen in Fig. 18(c), $|x - x_0| = \frac{|s|\sin\theta}{2}$ 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 in Appendix A, we 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 $|q-x| = fs_2(s) < \frac{3}{5}|s|$ and $\cos\theta \geq \frac{4}{5}$,

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

Now, by Proposition 8 in Appendix A, there is a vertex v of triangle t which has a Delaunay neighbor p such that $p \notin f$ and $|p-v| < 2R_t$ and thus t has been queued.

In Cases 1 and 2 it was shown that triangle t must have been queued. If t is in the queue then the inductive hypothesis holds. Alternatively t could have been processed and its circumcenter rejected by the safety rule. We will demonstrate that causes a contradiction.

The circumcenter of t can only be rejected if there was some segment \hat{s} with

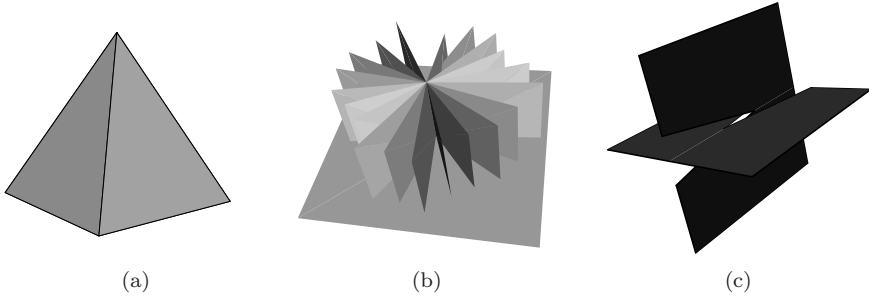


Fig. 19. Three example PLCs. (a) Example 1. (b) Example 2. (c) Example 3.

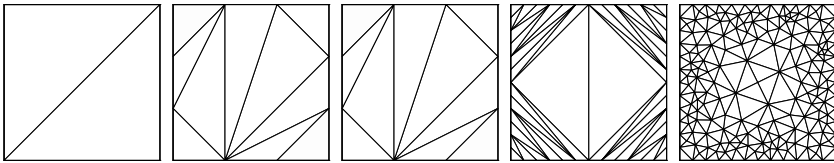


Fig. 20. Base of the pyramid following steps 0, 1a, 1b, 2a, and 2b.

endpoint \hat{q} such that $|c_t - \hat{q}| < |\hat{s}|$ and \hat{s} lies in f ; see Fig. 18(d). Since f is disjoint from s , all points of \hat{s} must be a 1-feature size witnesses for s . Then,

$$\begin{aligned}
 |q - \hat{q}|^2 &= |q - x|^2 + |x - \hat{q}|^2 \leq 3R_t^2 + (|c_t - \hat{q}| + |x - c_t|)^2 \\
 &\leq 3R_t^2 + (|\hat{s}| + R_t)^2 \leq 7|\hat{s}|^2.
 \end{aligned}
 \tag{13}$$

The second inequality holds since the circumdisk of t must be empty by the Delaunay property. The final inequality uses the estimate $|\hat{s}| > |c_t - \hat{q}| \geq R_t$. Recalling Lemma 10 inequality (11) is maintained throughout this step, so $\text{dist}(s, \hat{s}) \geq 2\sqrt{2}|\hat{s}|$. This inequality contradicts (13) since $\sqrt{7} < 2\sqrt{2} = \sqrt{8}$. \square

6. Examples

The following three examples are given to demonstrate Algorithm 6. For display purposes the end segment split rules have been relaxed. The precise details of this modification and a few other heuristics which are useful in practice are discussed following the examples.

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

Example 2. This example consists of a wheel of 20 faces which lies slightly above a disjoint square as depicted in Fig. 19(b). The mesh of the square base produced

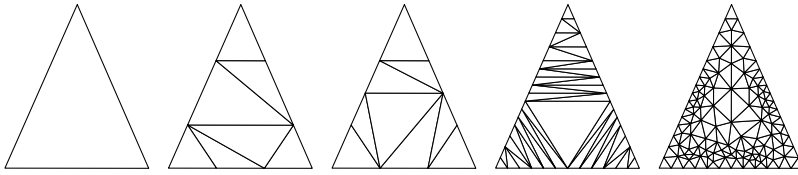


Fig. 21. Side of the pyramid following steps 0, 1a, 1b, 2a, and 2b.

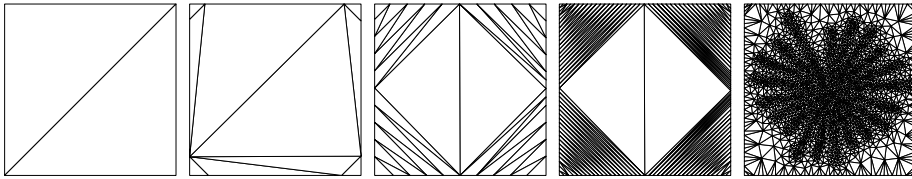


Fig. 22. Base plane of the wheel example following steps 0, 1a, 1b, 2a, and 2b.

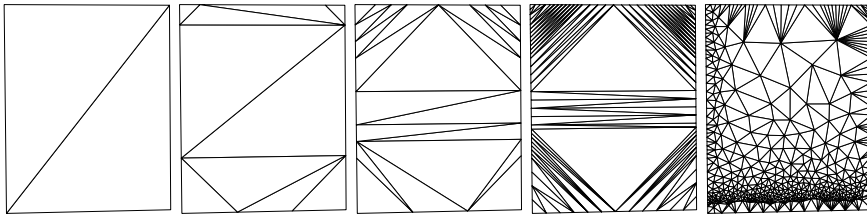


Fig. 23. One “spoke” in the wheel example following steps 0, 1a, 1b, 2a, and 2b. The center of the wheel is at the bottom and the disjoint square is to the left of this face.

following each step of the algorithm can be seen in Fig. 22. Similar output for one of the rectangular “spokes” of the wheel is given in Fig. 23. 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.

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

		Step				
		0	1a	1b	2a	2b
	4	72	202	518	2,051	11,351
Step 2a Factor	2	72	202	518	1,029	7,449
	0	72	202	518	518	5,348

Note: Following Step 0, the mesh contains the 46 input vertices plus 26 vertices for the initial bounding box.

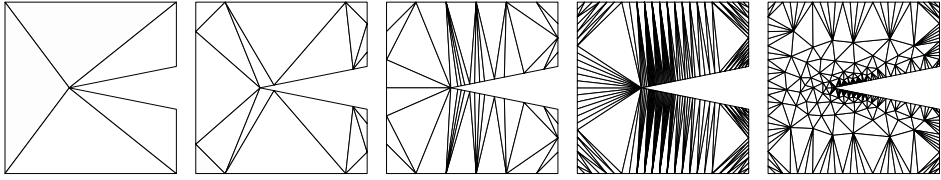


Fig. 24. One of the faces in Example 3 following steps 0, 1a, 1b, 2a, and 2b.

Example 3. In the final example, we consider a PLC containing two non-convex faces shown in Fig. 19(c). The refinement of one face is shown in Fig. 24.

In these examples nearby edges typically cause more refinement than nearby faces. Step 2a causes segments to be split in fourths *after* they have been refined to realize 1-feature size. The results are then consistent with the different bounds with respect to 1- and 2-feature size in Theorems 4 and 5.

In these examples additional refinement by the end segment split rules has not been performed. In practice non-end segments created when splitting end segments have sufficiently close 2-feature size witnesses to ensure further refinement. No end segment split rule was required in the proof of Lemma 7 and bounds on the segment length could still be ensured. Without the end segment split rules, the estimate in Lemma 7 likely holds following Step 2b; i.e., $|s| \leq \sqrt{2} fs_1(s)$.

In practice the algorithm has been seen to terminate even after changing Step 2a to only split segments in half (instead of fourths). This often significantly reduces the output size. The 1-feature size bound of Theorem 5 no longer holds but a bound with a weakened constant (i.e., $\frac{1}{\sqrt{2}}$ rather than $\frac{1}{2\sqrt{2}}$) is seen. This typically does not impact subsequent conforming Delaunay refinement algorithms for which constructions are affected only by the larger constant $\frac{5}{3}$ associated with the 2-feature size estimate. Table 1 contains the number of vertices in Example 2 for three different variants of Step 2a. In further studies we seek to prove correctness of the modified we have considered or demonstrate a counterexamples in which the algorithm can fail.

Acknowledgments

This work was supported in part by National Science Foundation Grant DMS-0811029. Additional support from the NSF was provided through the Center for Nonlinear Analysis. Much of this work was performed while the first author was at Carnegie Mellon University.

Appendix A. Auxiliary Geometric Results

Proposition 3. *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 $|q - \bar{q}| < |s|$.*

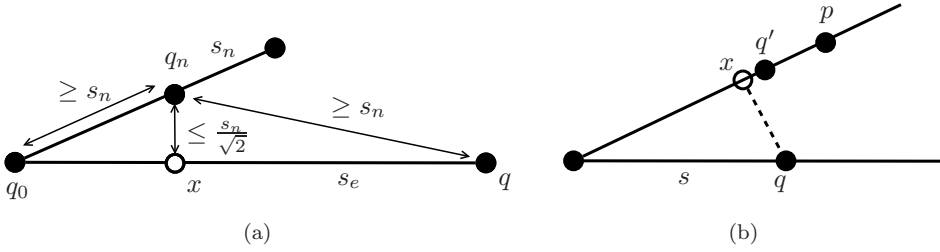


Fig. 25. Diagrams for the proofs of two propositions. (a) Proposition 4. (b) Proposition 5.

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

Remark 1. Proposition 3 will be used in a contrapositive form: if $\text{dist}(s, \bar{s}) < \frac{|s|}{\sqrt{2}}$, q and \bar{q} are endpoints of s and \bar{s} , respectively, and $|q - \bar{q}| \geq |s|$ then $|\bar{s}| \geq |s|$.

The constant in Proposition 3 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.

Proposition 4. Let s_e be an end segment and let s_n be a non-end segment on an input segment adjacent to s_e . Let q_0 be the input vertex on s_e . If $\text{dist}(s_n, s_e) < \frac{s_n}{\sqrt{2}}$ and $|q_n - q_e| \geq |s_n|$ for any pair of endpoints of s_e and s_n then $|s_e| > \sqrt{2}|s_n|$ and $|s_e| > \text{dist}(s_n, q_0)$.

Proof. Let x be the nearest point on s_e to s_n , let q be the unlabeled vertex of s_e , and let q_n denote the endpoint of s_n nearest to q_0 as in Fig. 25(a). The Pythagorean theorem ensures that $|q_0 - x| > \frac{|s_n|}{\sqrt{2}}$ and $|q - x| > \frac{|s_n|}{\sqrt{2}}$. The second inequality also follows from the Pythagorean theorem:

$$\begin{aligned}
 |s_e|^2 &> |q_0 - x|^2 + |q - x|^2 \\
 &= \text{dist}(s_n, q_0)^2 + |q_n - q_0|^2 - 2 \text{dist}(s_n, s_e)^2 > \text{dist}(s_n, q_0)^2.
 \end{aligned}
 \quad \square$$

Proposition 5. 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 $|s_p| \leq |q - p|$.

Proof. If p lies on the same input segment as q , then there 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, denoted s_0 as in Fig. 25(b). Let x be the nearest point on this adjacent input segment to q . Then $|q - p| > |x - p| > |q' - p| \geq |s_p|$. \square

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

- $fs_2(s) = fs_1(s)$.
- *There exists an endpoint q of s and a point x in the interior of a face f such that $s \cap f = \emptyset$, $|q - x| = fs_2(s)$, and \overline{qx} is orthogonal to f .*

Proof. If $fs_2(s) \neq fs_1(s)$ there exists a face f and point $x \in f$ such that f is disjoint from s and $fs_2(s) = |x - y|$ for some $y \in s$. Then $x \notin \partial f$ 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 $fs_2(s) = fs_1(s)$. 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 case 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 $y \in s$, we 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 . \square

Proposition 7. *Consider a set of coplanar vertices \mathcal{P} . Suppose ball $B(x_0, R)$ contains no vertices of \mathcal{P} . Consider $x \in B(x_0, R)$ where x lies 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 ; see Fig. 26.

Proof. Let B_t be the circumball of t . 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 $\{p_1, p_2\} = \partial B_t \cap \partial B(x_0, R)$ and let $s = \overline{p_1 p_2}$. We consider two cases; see Fig. 27.

Case 1. s lies between x and x_0 . Then $R^2 = \text{dist}(x_0, s)^2 + \frac{|s|^2}{4}$ by the Pythagorean theorem. Applying $\text{dist}(x_0, s) \leq |x - x_0|$ yields $R_t \geq \frac{|s|}{2} \geq \sqrt{R^2 - |x - x_0|^2}$.

Case 2. s does not lie between x and x_0 . Let L be the line parallel to s which passes through x_0 . Then $L \cap B(x_0, R) \subset B_t$ and 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 ,*

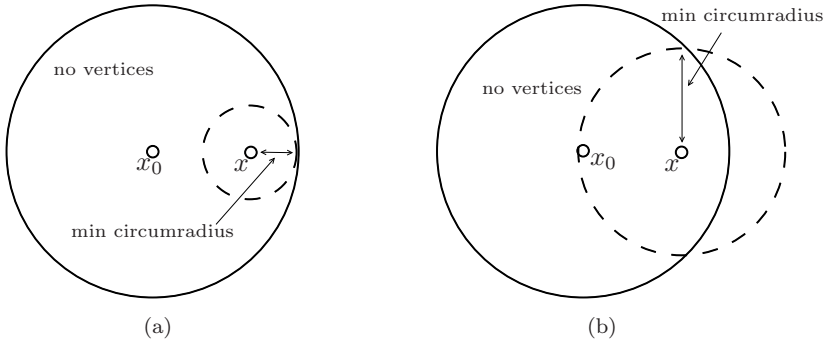


Fig. 26. Diagrams for Proposition 7. (a) If point x is contained in the circumcircle of (Delaunay) triangle t and lies in an empty disk, then the circumradius of t is at least the distance from x to the boundary of the empty disk. (b) If point x is contained in (Delaunay) triangle t and lies in an empty disk, then the the circumradius of t is larger than the indicated distance.

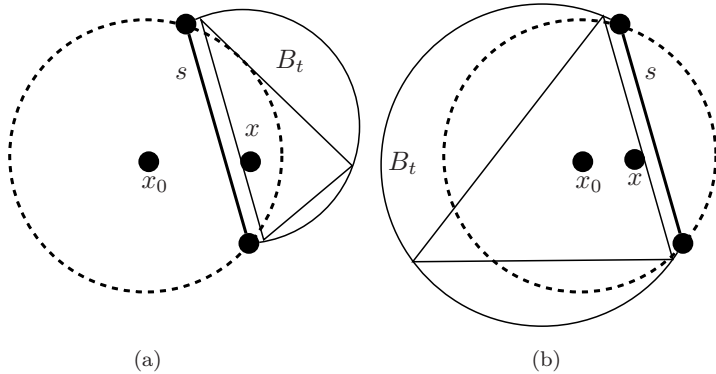


Fig. 27. Two cases in the proof of Proposition 7. (a) Case 1. (b) Case 2.

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 c_t be the circumcenter of t . Since t is covered by the three diametral balls between each vertex of t and c_t , there is a vertex q_t of t such that $|x - q_t| \leq R_t$; see Fig. 28. Since x lies in t , there is a vertex of t , denoted q_t , such that $|x - q_t| \leq R_t$. Let $y = c_t + q - x$; see Fig. 29. Let $B(\overline{q_t y})$ denote ball with diameter $\overline{q_t y}$. Then the following properties hold:

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

By Proposition 2, 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 disk of c_t and q_t must be empty since t is a Delaunay triangle in the face. □

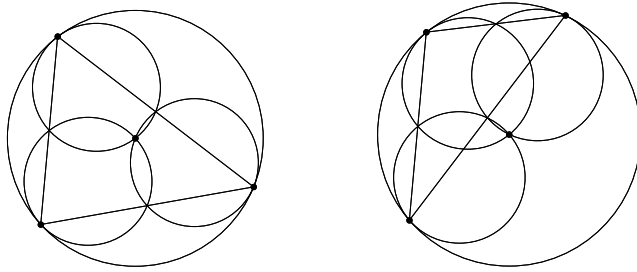


Fig. 28. Given any triangle, the three diametral balls between vertices and the circumcenter cover the triangle.

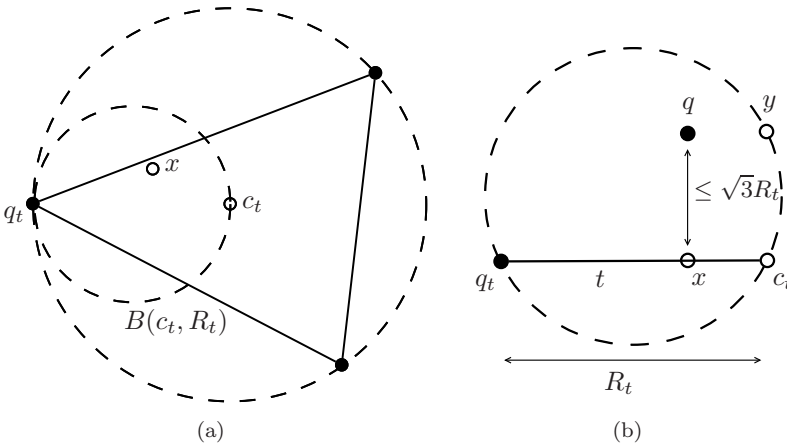


Fig. 29. Triangle t in Proposition 8. (a) Top view. (b) Side view.

References

1. J. Ruppert, A Delaunay refinement algorithm for quality 2-dimensional mesh generation, *J. Algorithms* **18**(3) (1995) 548–585.
2. M. Murphy, D. M. Mount and C. W. Gable, A point-placement strategy for conforming Delaunay tetrahedralization, *Int. J. Comput. Geom. Appl.* **11**(6) (2001) 669–682.
3. D. Cohen-Steiner, É. Colin de Verdière and M. Yvinec, Conforming Delaunay triangulations in 3D, *Comput. Geom.* **28**(2-3) (2004) 217–233.
4. S.-W. Cheng and S.-H. Poon, Three-dimensional Delaunay mesh generation, *Proc. 14th Symp. Discrete Algorithms* (2003), pp. 295–304.
5. S. E. Pav and N. J. Walkington, Robust three dimensional Delaunay refinement, *Proc. 13th Int. Meshing Roundtable* (2004), pp. 145–156.
6. H. Si and K. Gartner, Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations, *Proc. 14th Int. Meshing Roundtable* (2005) pp. 147–163.
7. H. Si, On refinement of constrained Delaunay tetrahedralizations, *Proc. 15th Int. Meshing Roundtable* (2006) pp. 510–528.
8. S.-W. Cheng, T. K. Dey and E. A. Ramos, Delaunay refinement for piecewise smooth complexes, *Proc. 18th Symp. Discrete Algorithms* (2007), pp. 1096–1105.

9. 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.
10. T. K. Dey and J. A. Levine, Delaunay meshing of piecewise smooth complexes without expensive predicates, *Algorithms* **2**(4) (2009) 1327–1349.
11. 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.
12. A. Rand and N. Walkington, Collars and intestines: Practical conforming Delaunay refinement, *Proc. 18th Int. Meshing Roundtable* (2009) pp. 481–497.
13. S.-W. Cheng and S.-H. Poon, Three-dimensional Delaunay mesh generation, *Discr. Comput. Geom.* **36**(3) (2006) 419–456.
14. L. P. Chew, Guaranteed-quality mesh generation for curved surfaces, *Proc. 9th Symp. Comput. Geom.* (1993), pp. 274–280.
15. C. Boivin and C. Ollivier-Gooch, Guaranteed-quality triangular mesh generation for domains with curved boundaries, *Int. J. Numer. Meth. Engrg.* **55**(10) (2002) 1185–1213.
16. L. P. Chew, Guaranteed-quality triangular meshes, Technical Report TR-89-983, Computer Science Department, Cornell University, 1989.
17. A. Üngör, Off-centers: a new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations, *Proc. 6th Latin Amer. Symp. Theor. Inform.* (2004), pp. 152–161.
18. 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.
19. R. Jampani and A. Üngör, Construction of sparse well-spaced point sets for quality tetrahedralizations, *Proc. 16th Int. Meshing Roundtable* (2007), pp. 63–80.
20. B. Hudson, G. Miller and T. Phillips, Sparse Voronoi refinement, *Proc. 15th Int. Meshing Roundtable* (2006), pp. 339–356.
21. 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.
22. G. L. Miller, A time efficient Delaunay refinement algorithm, *Proc. 15th Symp. Discrete Algorithms* (2004), pp. 400–409.
23. 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.
24. A. Rand, Reordering Ruppert’s algorithm, *Proc. 18th Fall Workshop Comput. Geom.* (2008).
25. J. R. Shewchuk, Mesh generation for domains with small angles, *Proc. 16th Symp. Comput. Geom.* (2000), pp. 1–10.
26. G. L. Miller, S. E. Pav and N. J. Walkington, When and why Ruppert’s algorithm works, *Proc. 12th Int. Meshing Roundtable* (2003), pp. 91–102.