Chapter 6

Delaunay refinement in the plane

Delaunay refinement algorithms generate high-quality meshes by inserting vertices into a Delaunay or constrained Delaunay triangulation. The vertices are placed to ensure domain conformity and to eliminate elements that are badly shaped or too large. Delaunay refinement has many virtues: the optimality of Delaunay triangulations with respect to criteria related to interpolation and the smallest angles; our rich inheritance of theory and algorithms for Delaunay triangulations; the fast, local nature of vertex insertion; and most importantly, the guidance that the Delaunay triangulation provides in finding locations to place new vertices that are far from the other vertices, so that short edges do not appear.

This chapter describes an enormously influential Delaunay refinement algorithm for triangular mesh generation invented by Jim Ruppert—arguably the first provably good mesh generation algorithm to be truly satisfying in practice. Ruppert's algorithm was inspired by Chew's algorithm, discussed in Section 1.2, but it has the advantage that it constructs graded meshes and it offers mathematical guarantees on triangle grading and size optimality. Figure 6.1 shows a mesh produced by Ruppert's algorithm.

We use Ruppert's algorithm to introduce ideas that recur throughout the book. We begin by introducing a generic template that summarizes most Delaunay refinement algorithms. Then we describe Ruppert's algorithm and show that, under the right conditions, it is mathematically guaranteed to produce high-quality triangular meshes. Ruppert's work established the theoretical strength of the generic algorithm, which we extend to tetrahedral meshing in subsequent chapters.

One guarantee is that the triangles in these meshes have high quality. Another guarantee is that the edges are not unduly short: there is a lower bound on the edge lengths proportional to a space-varying function called the *local feature size*, which roughly quantifies the longest edge lengths possible in any high-quality mesh of a specified piecewise linear complex. A highlight of the theory of mesh generation is that one can estimate both the maximum edge lengths and the minimum number of triangles in a high-quality mesh. Ruppert's algorithm comes within a constant factor of optimizing both these quantities, and so we say that the meshes it produces are *size-optimal*, a notion we define formally in Section 6.5.

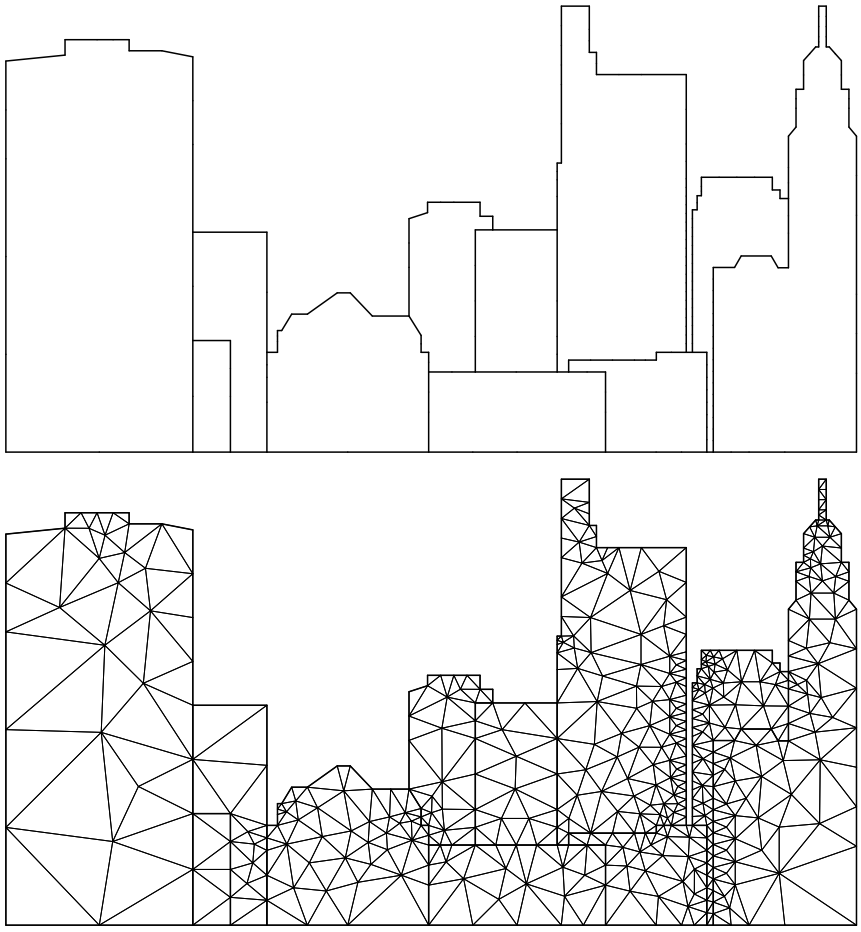


Figure 6.1: Meshing the Columbus skyline: a PLC and a Delaunay mesh thereof with no angle less than 28°, produced by Ruppert's Delaunay refinement algorithm.

6.1 A generic Delaunay refinement algorithm

The following generic Delaunay refinement method meshes a complex $\mathcal{P}$.

D R ($\mathcal{P}$)

1. Choose a vertex set $S \subset |\mathcal{P}|$.
2. Compute Del S.
3. If Del S fails to satisfy a property guaranteeing its geometric or topological conformity to $\mathcal{P}$, choose a point $c \in |\mathcal{P}|$ at or near the violation, insert c into S, update Del S, and repeat step 3.
4. If an element $\tau \in$ Del S is poorly shaped or too large, choose a point $c \in |\mathcal{P}|$ in or near τ's circumball, insert c into S, update Del S, and go to step 3.

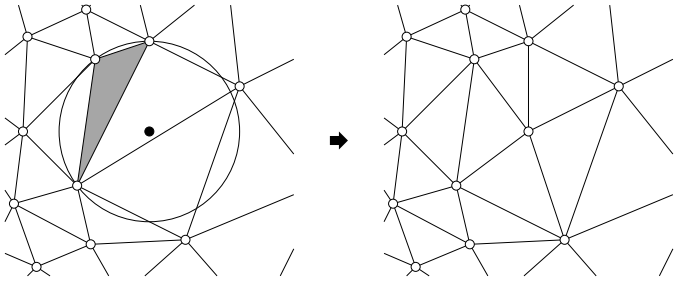


Figure 6.2: Inserting the circumcenter of a triangle whose radius-edge ratio exceeds 1. Every new edge adjoins the circumcenter and is at least as long as the circumradius of the skinny triangle, which is at least as long as the shortest edge of the skinny triangle.

5. Return the mesh $\{\sigma \in \operatorname{Del} S : \sigma \subseteq |\mathcal{P}|\}$.

The first step depends on the type of complex. For a piecewise linear domain, S is the set of vertices in $\mathcal{P}$. For a smooth or piecewise smooth domain, points are chosen carefully from the surfaces to form an initial vertex set. Step 2 constructs an initial Delaunay triangulation. Some algorithms maintain a constrained Delaunay triangulation instead; see Section 6.7. Steps 3 and 4 refine the triangulation by calling a vertex insertion algorithm such as the Bowyer–Watson algorithm. The properties enforced by Step 3 can encompass conforming to the domain boundary, capturing the domain topology, and approximating curved domain geometry with sufficient accuracy. Step 5 obtains a mesh of $\mathcal{P}$ by deleting the extraneous simplices that are not included in $|\mathcal{P}|$.

D R terminates only when the mesh has the desired geometric properties and element shapes and sizes. The algorithm designer's burden is to choose the properties to enforce, choose vertices to insert, and prove that the mesh will eventually satisfy the properties, so the algorithm will terminate. Counterintuitive as it may seem, the proof usually proceeds by first showing that the algorithm must terminate, and hence the properties must be satisfied and the elements must have high quality.

The main insight behind all Delaunay refinement algorithms is that they constrain how close together two vertices can be, and thus constrain how short an edge can be. Consider inserting a new vertex at the circumcenter of an element whose radius-edge ratio is one or greater, as illustrated in Figure 6.2. Because a Delaunay simplex has an empty circumball, the distance from the new vertex to any other vertex is at least as great as the circumball's radius, which is at least as great as the simplex's shortest edge. Therefore, *the new vertex cannot participate in an edge shorter than the shortest edge already existing*. This explains why circumcenters are excellent places to put new vertices, why the radius-edge ratio is the quality measure naturally optimized by Delaunay refinement algorithms, and why these algorithms must terminate: they eventually run out of places to put new vertices. The following lemma formalizes the last idea.

Lemma 6.1 (Packing Lemma)**.** *Let $D \subset \mathbb{R}^d$ be a bounded domain. Let $S \subset D$ be a point set and $\lambda > 0$ a scalar constant such that for every two distinct points u and v in S, $d(u, v) \geq \lambda$. Then there is a constant ξ depending solely on D and λ such that $|S| \leq \xi$.*

P . Consider the point set $D_{\lambda/2} = \{x \in \mathbb{R}^d : d(x, D) \leq \lambda/2\}$, known as the *Minkowski sum* of D with a ball of radius $\lambda/2$. $D_{\lambda/2}$ is bounded, as D is bounded. Every Euclidean d-ball having a center in S and radius $\lambda/2$ is included in $D_{\lambda/2}$. Because every pair of points in S is separated by a distance of at least λ, these balls have disjoint interiors, and we call this set of d-balls a *packing*. Therefore, $|S| \leq \text{volume}(D_{\lambda/2})/((\lambda/2)^d V_d)$, where V_d is the volume of a unit d-ball. □

The Delaunay refinement algorithms we study in this book place a lower bound λ on inter-vertex distances that is proportional to the aforementioned *local feature size* function, which locally estimates the longest edge lengths possible in a high-quality mesh. For a polyhedral domain, the local feature size is related to the distances between disjoint linear cells. For a domain whose boundary is a smooth surface, it is related to the distance from that boundary to its medial axis. We give formal definitions for the local feature size in the sections where they are first used.

6.2 Ruppert's Delaunay refinement algorithm

The input to Ruppert's algorithm is a piecewise linear complex $\mathcal{P}$ (recall Definition 2.8) and a positive constant $\bar{\rho}$ that specifies the maximum permitted radius-edge ratio for triangles in the output mesh. Recall from Section 1.7 that this equates to a minimum permitted angle of $\theta_{\min} = \arcsin \frac{1}{2\bar{\rho}}$ and a maximum permitted angle of $\theta_{\max} = 180^\circ - 2\theta_{\min}$. The output is a mesh of $\mathcal{P}$—that is, a Steiner triangulation of $\mathcal{P}$ (recall Definition 2.12)—whose triangles all meet the standard of quality. The algorithm is guaranteed to terminate if $\bar{\rho} \geq \sqrt{2}$ and no two edges in $\mathcal{P}$ meet at an acute angle. (We remove the latter restriction in Section 6.6).

Ruppert's algorithm begins by setting S to be the set of vertices in $\mathcal{P}$ and constructing $\text{Del}\, S$. Let $\rho(\tau)$ denote the radius-edge ratio of a triangle τ. Suppose that step 4 of D R is elaborated as:

> If there is a triangle $\tau \in \text{Del}\, S$ with $\rho(\tau) > \bar{\rho}$, insert the circumcenter c of τ into S, update $\text{Del}\, S$, and go to step 3.

Let λ be the shortest distance between any two points in S before c is inserted. Let r be τ's circumradius, and let ℓ be the length of τ's shortest edge. Recall the main insight from the previous section: because τ's open circumdisk is empty, $d(c, S) \geq r = \rho(\tau)\ell > \bar{\rho}\lambda$. If we choose $\bar{\rho} \geq 1$, then step 4 maintains a lower bound of λ on inter-point distances and we can apply the Packing Lemma (Lemma 6.1). If we choose $\bar{\rho} = 1$, every triangle whose smallest angle is less than 30° will be split by the algorithm. Therefore, if the algorithm terminates, every triangle has all its angles between 30° and 120°.

Unfortunately, this simple refinement rule has a serious flaw, because some of the new vertices might lie outside the domain $|\mathcal{P}|$. Moreover, recall from Figure 2.12 that $\text{Del}\, S$ does not always respect the domain boundaries. We fix both problems by incorporating special treatment of the domain boundaries into step 3 of the algorithm.

A goal of the algorithm is to force each *segment*—each edge in $\mathcal{P}$—to be represented by a sequence of edges in $\text{Del}\, S$. During refinement, the algorithm maintains a set E of *subsegments*, edges that should ideally appear in the mesh. Initially, E is the set of segments in $\mathcal{P}$; but as the algorithm adds new vertices to S, these vertices subdivide segments into subsegments, and

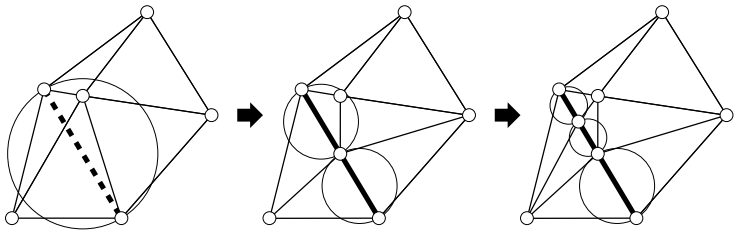


Figure 6.3: Segments (bold) are split until no subsegment is encroached.

subsegments into shorter subsegments. The algorithm explicitly maintains the identity of the subsegments, whether or not they appear as edges in Del S.

Consider a subsegment e that is absent from Del S, as at upper right in Figure 2.12. By Proposition 2.10, Del S contains every strongly Delaunay edge, so e is not strongly Delaunay, and some vertex $v \in S$ must lie in e's closed diametric disk besides e's vertices.

Definition 6.1 (encroachment)**.** A vertex v that lies in the closed diametric ball of a subsegment e but is not a vertex of e is said to *encroach upon* e.

The following procedure S S treats an encroached subsegment by inserting a new vertex at its midpoint, thereby dividing it into two subsegments of half the length. All encroached subsegments are treated this way, whether they are present in Del S or not, as illustrated in Figure 6.3. The recovery of a boundary segment by repeatedly inserting vertices on its missing subsegments is sometimes called *stitching*.

S S (e, S, E)

1. Insert the midpoint of e into S.
2. Remove e from E and add its two halves to E.

We will see at the end of this section that if no subsegment is encroached, every triangle's circumcenter lies in the domain $|\mathcal{P}|$, so we can remove any skinny triangle by inserting a vertex at its circumcenter. However, this triangle refinement rule still has a serious flaw: a vertex inserted at a circumcenter can lie arbitrarily close to a segment, thereby forcing the Delaunay refinement algorithm to generate extremely small triangles in the region between the vertex and segment. We sidestep this hazard with a more complicated rule for treating skinny triangles that prevents vertices from getting dangerously close to segments, at the cost of weakening the angle bounds from 30° and 120° to approximately 20.7° and 138.6°. If a triangle τ has a radius-edge ratio greater than $\bar{\rho}$, we consider inserting its circumcenter c, as before. But if c encroaches upon a subsegment e, we reject c and split e instead.

S T (τ, S, E)

1. Let c be the circumcenter of τ.
2. If c encroaches upon some subsegment $e \in E$, call S S (e, S, E). Otherwise, insert c into S.

If S T elects not to insert c and instead calls S S , we say that c is *rejected*. Although rejected vertices do not appear in the final mesh, they play a significant role in the algorithm's analysis. The new vertex inserted at the midpoint of e by S S might or might not lie in τ's open circumdisk. If not, τ survives the call to S T , but the Delaunay refinement algorithm can subsequently attempt to split τ again.

The following pseudocode implements Ruppert's Delaunay refinement algorithm.

D T PLC($\mathcal{P}, \overline{\rho}$)

1. Let S be the set of vertices in $\mathcal{P}$. Let E be the set of edges in $\mathcal{P}$.
2. Compute Del S.
3. While some subsegment $e \in E$ is encroached upon by a vertex in S, call S - S (e, S, E), update Del S, and repeat step 3.
4. If Del S contains a triangle $\tau \subseteq |\mathcal{P}|$ for which $\rho(\tau) > \overline{\rho}$, call S T ($\tau, S, E$), update Del S, and go to step 3.
5. Return the mesh $\{\sigma \in \text{Del } S : \sigma \subseteq |\mathcal{P}|\}$.

Figure 6.4 illustrates the algorithm D T PLC with a sequence of snapshots of the triangulation. The theoretical minimum angle of 20.7° is quite pessimistic compared to the algorithm's observed performance; it is almost always possible to obtain a minimum angle of 33° in practice.

Observe that step 4 ignores skinny triangles that lie outside the domain. D T PLC gives priority to encroached subsegments, and splits a triangle only if no subsegment is encroached. This policy guarantees that every new vertex lies in the domain $|\mathcal{P}|$.

Proposition 6.2. *Let $\mathcal{T}$ be a Steiner triangulation of a PLC $\mathcal{P}$ in the plane. If no subsegment in $\mathcal{T}$ (i.e. edge in $\mathcal{T}$ included in a segment in $\mathcal{P}$) is encroached, then every triangle in $\mathcal{T}$ has its circumcenter in $|\mathcal{P}|$.*

P . Suppose for the sake of contradiction that some triangle $\tau \in \mathcal{T}$ has a circumcenter $c \notin |\mathcal{P}|$. Let p be a point in the interior of τ. As Figure 6.5 shows, the line segment pc crosses from the interior of $|\mathcal{P}|$ to its exterior, and therefore must cross a subsegment e on the boundary of $|\mathcal{P}|$. We will show that e is encroached.

Let B_τ be the closed circumdisk of τ, let B_e be the closed diametric disk of e, and let H be the closed halfplane containing p whose boundary line is e's affine hull, so $B_e \cap H$ is a half-disk. The interior of B_τ must intersect e, as $pc \subset B_\tau$, but not e's vertices, as τ is Delaunay. Because the center c of B_τ lies outside H, it follows that $B_e \supset B_\tau \cap H \supset \tau$. Therefore, B_e contains all three vertices of τ. Two of τ's vertices might be vertices of e, but the third vertex encroaches on e. This contradicts the assumption that no subsegment is encroached. □

In Chapter 7, we describe a generalization of Proposition 6.2 called the Orthocenter Containment Lemma (Lemma 7.7), which provides an an alternative proof of the proposition.

6.3 Implementation and running time

We offer some recommendations on how to implement Ruppert's algorithm efficiently.

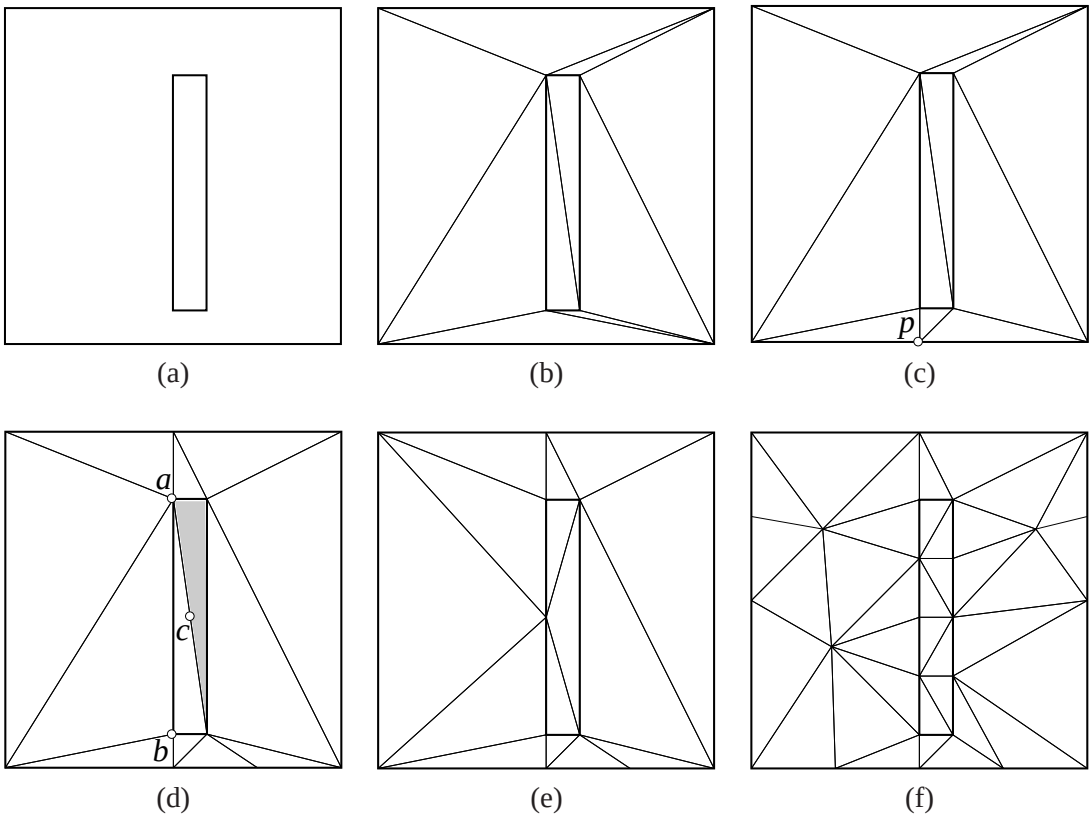


Figure 6.4: The algorithm D T PLC in action. (a) The input PLC. (b) The Delaunay triangulation of the input vertices. (c) The bottom edge is encroached, so it is split at its midpoint p. (d) After all the encroached subsegments are split, the shaded triangle has poor quality, so the algorithm considers inserting its circumcenter c. (e) Because c encroaches upon the segment ab, the algorithm does not insert c; instead it splits ab. (f) The final mesh.

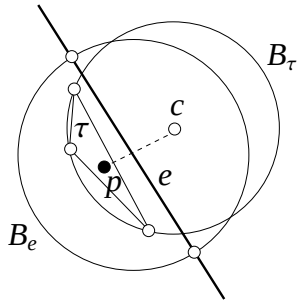


Figure 6.5: B_e includes the portion of τ's circumdisk B_τ on the same side of e as p.

First, an implementation should maintain a queue of skinny and oversized triangles and a queue of encroached subsegments, which help steps 3 and 4 to run quickly. Newly created triangles and subsegments are added to these queues if they ought to be split, but deleted triangles and subsegments are not removed from the queues until they reach the front of a queue and are discovered to be no longer in the mesh. Second, there is an easy, local, constant-time test for whether a subsegment e is encroached: either $e \notin \operatorname{Del} S$, or $\operatorname{Del} S$ contains a triangle that has e for an edge and an angle of 90° or greater opposite e—see Exercise 1. Therefore, after the initial Delaunay triangulation is computed, the queues can be initialized in time linear in the number of triangles.

Third, when a new vertex is inserted, only the newly created triangles need be checked to see if they are skinny, and only the edges of those triangles need be checked to see if they are encroached subsegments. Recall that the dictionary data structure of Section 3.2 maps each edge of the mesh to the two triangles that have it for an edge in expected constant time; the same data structure can record which edges of the mesh are subsegments. If the new vertex splits a subsegment, the two new subsegments must be tested; recall Figure 6.3.

Fourth, the easiest way to discover whether a triangle circumcenter encroaches upon a subsegment and should be rejected is to insert it, then check the edges opposite it. It is wise to store the deleted triangles so they can be rapidly restored if the circumcenter is rejected. A triangle that is deleted then restored should be in the queue of bad triangles if and only if it was in the queue before the vertex insertion, *including the triangle that the circumcenter was meant to split*—the rejected circumcenter might be successfully inserted later.

Fifth, if the bound $\bar{\rho}$ on triangle quality is strict, prioritizing the skinny triangles so that those with the largest radius-edge ratios (the smallest angles) are split first usually reduces the number of triangles in the final mesh by as much as 35%. (If the bound is not strict, ordering seems to make little difference.) Circumcenters of very skinny triangles tend to eliminate more skinny triangles than circumcenters of mildly skinny triangles. Although one can use a binary heap to prioritize triangles, experiments show that it is much faster, and equally effective, to maintain multiple queues representing different quality ranges.

Sixth, if the domain $|\mathcal{P}|$ is not convex, the triangulation $\operatorname{Del} S$ contains triangles that lie outside the domain but in its convex hull. If a tiny angle forms between the boundary of $|\mathcal{P}|$ and the boundary of $\operatorname{conv}|\mathcal{P}|$, D T PLC could perpetually add new vertices, trying to get rid of the skinny triangle in vain. To avoid this fate, step 4 of the algorithm declines to split triangles that are not in $|\mathcal{P}|$. Unfortunately, identifying these triangles can be costly. A simple alternative—in fact, Ruppert's original algorithm—is to enclose $\mathcal{P}$ in a large square bounding box and mesh the entire box. The advantage is that no small angle can form between $\mathcal{P}$ and the bounding box. The disadvantages are that time is spent creating triangles that will be discarded in the end, and that small *exterior* angles of $|\mathcal{P}|$ can still put the algorithm into an infinite loop of refinement. Section 6.7 discusses another alternative that works better but requires an implementation of constrained Delaunay triangulations.

There is a notable discrepancy between the running time of Ruppert's algorithm in practice and its running time in the theoretical worst case. Let n be the number of vertices in the PLC $\mathcal{P}$, and let $N \geq n$ be the number of vertices in the final mesh. With a careful implementation, Ruppert's algorithm is observed to consistently take $O(n \log n + N)$ time in practice. The term $n \log n$ covers the construction of the initial triangulation. The term N covers the cost of refinement, be-

cause most vertex insertions during Delaunay refinement take constant time. There is little need for point location during the refinement stage, as almost every new vertex the Delaunay refinement algorithm generates is associated with a known subsegment that contains it or a known triangle whose open circumball contains it. The exception is when the algorithm inserts the midpoint of a subsegment that is missing from the mesh, in which case the algorithm must search the triangles adjoining a vertex of the subsegment to find one whose open circumball contains the midpoint. The use of a CDT eliminates this small cost.

However, recall from Section 3.5 that an unlucky vertex insertion can delete and create a linear number of triangles. PLCs are known for which refinement takes $\Theta(N^2)$ time, but such examples are contrived and do not arise in practice. Delaunay refinement tends to rapidly even out the distribution of vertices so that most new vertices have constant degree and take constant time to insert.

6.4 A proof of termination

D T PLC has two possible outcomes: either it will eventually delete the last skinny triangle or encroached subsegment and succeed, or it will run forever, creating new skinny triangles and encroached subsegments as fast as it deletes old ones. If we prove it does not run forever, we prove it succeeds.

If $\bar{\rho} > \sqrt{2}$ and no two edges in $\mathcal{P}$ meet at an acute angle, we can prove that D T PLC terminates by showing that it maintains a lower bound on the distances between mesh vertices, then invoking the Packing Lemma. The lengths of the edges in a mesh produced by Ruppert's algorithm are roughly proportional to a space-varying function that characterizes the local distances between the geometric features of a PLC.

Definition 6.2 (local feature size)**.** The *local feature size* (LFS) of a PLC $\mathcal{P}$, where $|\mathcal{P}| \subset \mathbb{R}^d$, is a function $f : \mathbb{R}^d \to \mathbb{R}$ such that $f(x)$ is the radius of the smallest ball centered at x that intersects two disjoint linear cells in $\mathcal{P}$.

Figure 6.6 shows local feature sizes for a PLC at three points p, q, and r. A useful intuition is to imagine an inflating ball centered at a point; the ball stops growing when it first intersects two mutually disjoint features. For example, the ball growing outward from p in the figure does not stop growing when it first intersects two segments, because they adjoin each other; it continues to grow until it touches a vertex that is disjoint from the upper segment, as shown. Then $f(p)$ is the radius of the ball.

For every point $p \in \mathbb{R}^d$, there is a linear cell at a distance $f(p)$ from p. Moreover, there is a linear cell with dimension less than d at a distance $f(p)$, because the boundary of every d-cell in $\mathcal{P}$ is a union of lower-dimensional cells in $\mathcal{P}$. Omitting the d-cells from a PLC does not change f.

An important property of the local feature size is that it is 1-Lipschitz.

Definition 6.3 (k-Lipschitz)**.** A real-valued function φ is k-Lipschitz if for any two points x and y,

$$\varphi(x) \leq \varphi(y) + k\, d(x, y).$$

The 1-Lipschitz property implies that f is continuous and its gradient, where it exists, has magnitude no greater than 1; but f is not differentiable everywhere.

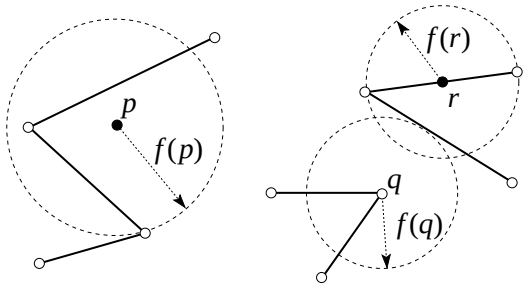


Figure 6.6: Examples of local features sizes in $\mathbb{R}^2$. At each point p, q, and r, the local feature size $f(p)$, $f(q)$, or $f(r)$ is the radius of the smallest disk that intersects two mutually disjoint linear cells.

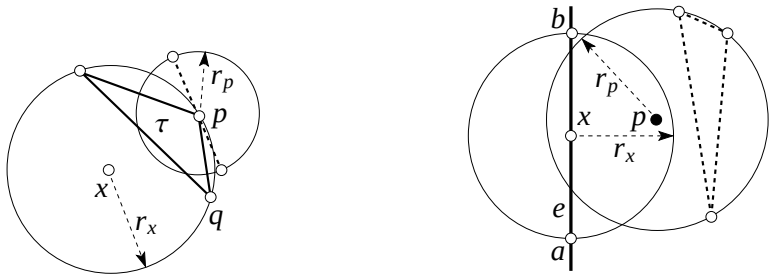
Proposition 6.3. *The local feature size f is* 1*-Lipschitz—that is, $f(x) \leq f(y) + d(x,y)$ for any two points $x, y \in \mathbb{R}^2$.*

P . By the definition of local feature size, $B(y, f(y))$ intersects two mutually disjoint cells in $\mathcal{P}$. The ball $B(x, f(y) + d(x,y))$ includes $B(y, f(y))$, and therefore intersects the same two cells. It follows that $f(x) \leq f(y) + d(x,y)$. □

We assign every mesh vertex x an *insertion radius* $r_x = d(x, S)$, the distance from x to the nearest distinct vertex in the set S of mesh vertices the instant before x is added to S. Observe that immediately after x is added to S, r_x is the length of the shortest edge adjoining x in Del S. Every vertex that S T rejects for encroaching upon a subsegment has an insertion radius too, equal to its distance to the nearest vertex in S at the moment it is considered and rejected. To show that D T PLC does not run forever, we will prove that the insertion radii do not become much smaller than the local feature size.

We say that a vertex is of *type i* if it lies on a linear i-cell in $\mathcal{P}$ but not on a lower-dimensional cell. Every vertex inserted or rejected by D T PLC, except vertices of type 0, has a *parent* vertex that, loosely speaking, is blamed for generating the child vertex. The parent might be in S, or it might be a rejected vertex.

- *Type 0 vertices* are input vertices in $\mathcal{P}$. A type 0 vertex has no parent. The insertion radius r_x of a type 0 vertex x is the distance from x to the nearest distinct vertex in $\mathcal{P}$.
- *Type 1 vertices* are those inserted at the midpoints of encroached subsegments by S - S . For a type 1 vertex x generated on an encroached subsegment e, there are two possibilities. If the point encroaching upon e is a vertex in S, define x's parent p to be the vertex in S nearest x; then the insertion radius of x is $r_x = d(x, p)$. Otherwise, the encroaching point p is a rejected circumcenter, which we define to be x's parent, and x's insertion radius r_x is the radius of e's diametric ball. Because p is in that ball, $d(x, p) \leq r_x$.
- *Type 2 vertices* are those inserted or rejected at triangle circumcenters by S T . For a type 2 vertex x generated at the circumcenter of a triangle τ, define x's parent p to be the most recently inserted vertex of the shortest edge of τ; if both vertices are in $\mathcal{P}$, choose one arbitrarily. Because τ's circumdisk is empty, the insertion radius of x is $r_x = d(x, p)$, the circumradius of τ.



(a) (b)

Figure 6.7: (a) The insertion radius r_x of the circumcenter x of a triangle τ is at least $\rho(\tau)$ times the insertion radius r_p of the most recently inserted vertex p of τ's shortest edge. (b) The insertion radius r_p of an encroaching circumcenter p of a Delaunay triangle is at most $\sqrt{2}$ times the insertion radius r_x of the midpoint x of the encroached subsegment.

The following proposition proves a lower bound on the insertion radius of each vertex in terms of either its parent's insertion radius or the local feature size. These bounds will lead to lower bounds on the edge lengths in the final mesh.

Proposition 6.4. *Let $\mathcal{P}$ be a PLC in which no two edges meet at an acute angle. Let x be a vertex inserted into S or rejected by* D T PLC. *Let p be the parent of x, if one exists.*

(i) *If x is of type 0, then $r_x \geq f(x)$.*

(ii) *If x is of type 1, and its parent p is of type 0 or 1, then $r_x \geq f(x)$.*

(iii) *If x is of type 1, and its parent p is of type 2, then $r_x \geq r_p/\sqrt{2}$.*

(iv) *If x is of type 2, then $r_x > \bar{\rho} r_p$.*

P . If x is of type 0, then the disk $B(x, r_x)$ contains $x \in \mathcal{P}$ and another vertex in $\mathcal{P}$, hence $f(x) \leq r_x$ by the definition of local feature size.

If x is of type 2, then x is the circumcenter of a Delaunay triangle τ with $\rho(\tau) > \bar{\rho}$, as illustrated in Figure 6.7(a). The parent p is the most recently inserted vertex of τ's shortest edge pq, or both p and q are vertices in $\mathcal{P}$; it follows from the definition of insertion radius that $r_p \leq d(p, q)$. The insertion radius of x (and the circumradius of τ) is $r_x = d(x, p) = \rho(\tau)\, d(p, q) > \bar{\rho} r_p$ as claimed.

If x is of type 1, it is the midpoint of a subsegment e of a segment $s \in \mathcal{P}$, and x's parent p encroaches upon e. There are three cases to consider, depending on the type of p.

- If p is of type 0, then $p \in \mathcal{P}$, and the disk $B(x, d(x, p))$ intersects two disjoint linear cells in $\mathcal{P}$, namely p and s. By the definition of local feature size, $f(x) \leq d(x, p) = r_x$.

- If p is of type 1, then p was previously inserted on some segment $s' \in \mathcal{P}$. The segments s and s' cannot share a vertex, because the presence of p in e's diametric disk would imply that s meets s' at an angle less than 90°. Thus $B(x, d(x, p))$ intersects two disjoint segments and $f(x) \leq d(x, p) = r_x$.

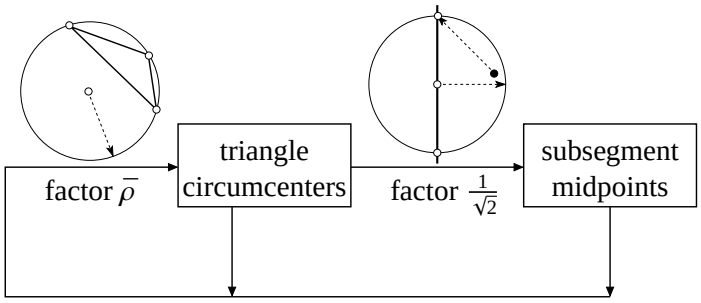


Figure 6.8: Flow graph illustrating the worst-case relation between a vertex's insertion radius and the insertion radii of the children it begets. If no cycle has a product smaller than one, Ruppert's algorithm will terminate.

- If p is of type 2, then p is a circumcenter rejected for encroaching upon e, so it lies in the diametric disk of e, whose center is x, as illustrated in Figure 6.7(b). Let b be the vertex of e nearest p; then $\angle pxb \leq 90°$ and $d(p,b) \leq \sqrt{2}d(x,b)$. D T PLC calls S T only when no subsegment is encroached, so $r_x = d(x,b)$. The parent p is the center of the circumdisk of a Delaunay triangle, so r_p is the radius of the circumdisk. The triangle's open circumdisk does not contain b, so $r_p \leq d(p,b) \leq \sqrt{2}d(x,b) = \sqrt{2}r_x$, as claimed.

□

The flow graph in Figure 6.8 depicts the relationship between the insertion radius of a vertex and the smallest possible insertion radii of its children, from Proposition 6.4. The boxes represent type 2 and type 1 vertices, respectively. Type 0 vertices are omitted because they cannot contribute to cycles in the flow graph. We can prove that D T PLC terminates by showing that it cannot produce sequences of vertices with ever-diminishing insertion radii—that is, there is no cycle in the flow graph whose product is less than one. This is true if we choose $\bar{\rho}$ to be at least $\sqrt{2}$. When a rejected circumcenter splits a subsegment, the newly created edges can be a factor of $\sqrt{2}$ shorter than the circumradius of the skinny triangle; we compensate for that by trying to split a triangle only if its circumradius is at least a factor of $\sqrt{2}$ greater than the length of its shortest edge. If $\bar{\rho} = \sqrt{2}$, the final mesh has no angle less than $\arcsin \frac{1}{2\sqrt{2}} \doteq 20.7°$.

From this reasoning, it follows that if $\bar{\rho} \geq \sqrt{2}$, D T PLC creates no edge shorter than the shortest distance between two disjoint linear cells in $\mathcal{P}$. Proposition 6.5 below makes a stronger statement: D T PLC spaces vertices proportionally to the local feature size. If a user chooses $\bar{\rho} < \sqrt{2}$, the algorithm will try to obtain the quality requested, but it might fail to terminate, or it might generate a mesh that is not properly graded.

Proposition 6.5. *Suppose that $\bar{\rho} > \sqrt{2}$ and no two segments in $\mathcal{P}$ meet at an angle less than* 90°. *Define the constants*

$$C_S = \frac{\left(\sqrt{2}+1\right)\bar{\rho}}{\bar{\rho} - \sqrt{2}}, \qquad C_T = \frac{\bar{\rho}+1}{\bar{\rho} - \sqrt{2}}.$$

Let x be a vertex inserted or rejected by D T PLC($\mathcal{P}, \bar{\rho}$).

(i) *If x is of type 1, then $r_x > f(x)/C_S$.*

(ii) *If x is of type 2, then $r_x > f(x)/C_T$.*

P . The expressions for C_S and C_T above arise as the solution of the equations $C_S = \sqrt{2}C_T + 1$ and $C_T = 1 + C_S/\bar{\rho}$, which are both used below. Observe that $C_S > C_T > 1$. The proof proceeds by induction on the sequence of points that are inserted or rejected by D T PLC. Let x be one such point, and suppose for the sake of induction that the claim holds for every previously inserted or rejected point.

If x is of type 1 and its parent p is of type 0 or 1, then $r_x \geq f(x) > f(x)/C_S$ by Proposition 6.4, confirming property (i). If x is of type 1 and p is of type 2, then $r_x \geq r_p/\sqrt{2}$ by Proposition 6.4 and $d(x, p) \leq r_x$. Inductive application of property (ii) gives $r_p > f(p)/C_T$. By the Lipschitz property of f (Proposition 6.3),

$$f(x) \leq f(p) + d(x, p) < C_T r_p + r_x \leq \left(\sqrt{2}C_T + 1\right) r_x = C_S r_x,$$

confirming property (i).

If x is of type 2, then by the inductive hypothesis its parent p satisfies $r_p > f(p)/C_S$. By Proposition 6.4, $r_x > \bar{\rho} r_p$. The Lipschitz property implies $f(x) \leq f(p) + d(x, p) = f(p) + r_x$, so

$$r_x > \bar{\rho} r_p > \frac{\bar{\rho}}{C_S} f(p) \geq \frac{\bar{\rho}}{C_S}(f(x) - r_x).$$

Rearranging terms gives

$$r_x > \frac{f(x)}{1 + C_S/\bar{\rho}} = \frac{f(x)}{C_T},$$

confirming property (ii). □

Proposition 6.5 gives a lower bound on the distance between a newly inserted vertex and all the preceding vertices. We want a more general bound on the distance between a vertex and all the other vertices, including those that are inserted later. The Lipschitz property of f allows us to derive the latter bound from the former.

Proposition 6.6. *Suppose that $\bar{\rho} > \sqrt{2}$ and no two segments in $\mathcal{P}$ meet at an angle less than* 90°. *For any two vertices p and q that* D T PLC *inserts, $d(p, q) \geq f(p)/(C_S + 1)$.*

P . If p is inserted after q, Proposition 6.5 states that $d(p, q) \geq f(p)/C_S$. If q is inserted after p, $d(p, q) \geq f(q)/C_S$. Because f is 1-Lipschitz, $f(p) \leq f(q) + d(p, q) \leq (C_S + 1)\, d(p, q)$. Either way, the result follows. □

Proposition 6.6 establishes a lower bound on the distances between vertices proportional to the local feature size, which implies that D T PLC is guaranteed to produce graded meshes if the local feature size function is strongly graded, as Figure 6.9 illustrates. We call this guarantee *provably good grading*, because it implies that small geometric features of the domain do not cause the algorithm to produce unduly short edges far from those features.

To make the proposition concrete, consider choosing $\bar{\rho} = 1.93$ to guarantee that no angle is smaller than roughly 15°. Then $C_S \doteq 9.01$, so the spacing of vertices is at worst about ten times

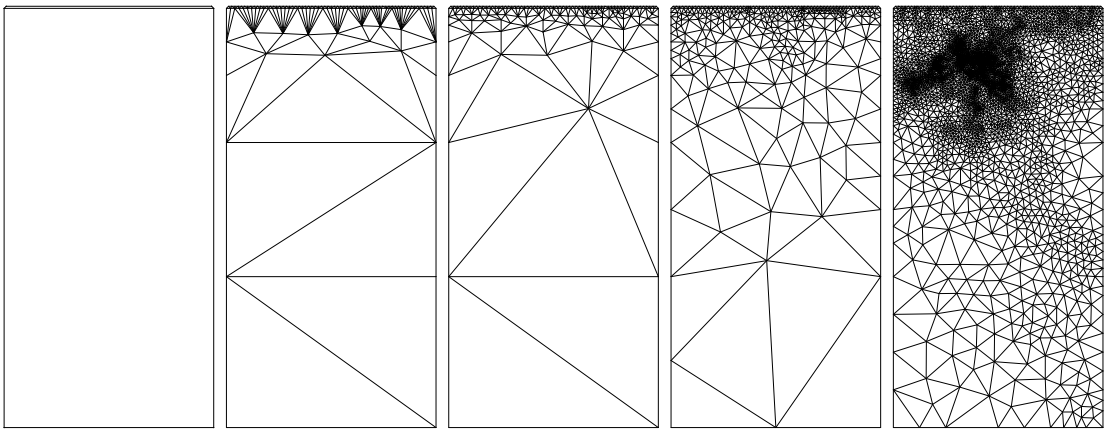


Figure 6.9: A domain with two polygons, the top one being extremely thin compared to the bottom one. Four meshes generated by Ruppert's algorithm, with no angle smaller than 5°, 20°, 30°, and 34.2°, respectively.

smaller than the local feature size. This worst theoretical outcome never occurs in practice; for example, the edges of the 20° mesh in Figure 6.9 are at least one third as long as their local feature sizes. The bound permits us to apply the Packing Lemma to prove that the algorithm does not run forever.

Theorem 6.7. *Suppose that* $\bar{\rho} > \sqrt{2}$ *and no two segments in* $\mathcal{P}$ *meet at an angle less than* 90°. *Then* D T PLC($\mathcal{P}, \bar{\rho}$) *terminates and returns a Steiner Delaunay triangulation of* $\mathcal{P}$ *whose triangles have radius-edge ratios at most* $\bar{\rho}$.

P . Let $f_{\min} = \min_{x \in |\mathcal{P}|} f(x)$. Because $\mathcal{P}$ is finite and any two disjoint linear cells in $\mathcal{P}$ are separated by a positive distance, $f_{\min} > 0$. By Proposition 6.6, D T PLC maintains an inter-vertex distance of least $f_{\min}/(C_S + 1) > 0$. By the Packing Lemma (Lemma 6.1), there is an upper bound on the number of vertices in the triangulation, so D T PLC must terminate. The algorithm terminates only if no subsegment is encroached and no skinny triangle lies in the domain, so it returns a high-quality mesh of $\mathcal{P}$ as stated. □

Figure 6.9 shows that the algorithm often succeeds for angle bounds well in excess of 20.7°, failing to terminate on the depicted domain only for angle bounds over 34.2°. The meshes illustrate the expected trade-off between mesh size and quality, albeit with longer edges than the lower bounds suggest.

For simplicity, we have not discussed the effects of refining triangles for being too large. In practice, it is usual for a user to specify a *size field* $\lambda : \mathbb{R}^2 \to \mathbb{R}$ that dictates space-varying upper bounds on the edge lengths or triangle circumradii in the mesh. Triangles that violate these bounds are split, just like skinny triangles. See Section 14.4 for an example of an analysis method that can extend the termination guarantee and derive lower bounds on the edge lengths in the mesh when refinement is driven by both a size field and the geometry of the domain.

6.5 A proof of size optimality and optimal grading

An algorithm is said to generate *size-optimal* meshes if the number of triangles in every mesh it produces is within a constant factor of the minimum possible number.

Definition 6.4 (size optimality)**.** Let P be a class of piecewise linear complexes—that is, a set containing all PLCs that satisfy some criterion. For every PLC $\mathcal{P} \in P$, let $\mathcal{T}(\mathcal{P}, \bar{\rho})$ be the triangulation with the fewest triangles among all possible Steiner triangulations of $\mathcal{P}$ whose triangles' radius-edge ratios do not exceed $\bar{\rho}$. Let $\mathcal{M}(\mathcal{P}, \bar{\rho})$ be the Steiner triangulation of $\mathcal{P}$ generated by an algorithm that guarantees that no triangle in the mesh has a radius-edge ratio greater than $\bar{\rho}$. The triangulations this algorithm generates are *size-optimal* if for every $\mathcal{P} \in P$, the number of triangles in $\mathcal{M}(\mathcal{P}, \bar{\rho})$ is at most c times the number of triangles in $\mathcal{T}(\mathcal{P}, \bar{\rho})$, where c is a constant that depends solely on $\bar{\rho}$.

Ruppert's algorithm generates size-optimal meshes of the class of PLCs whose underlying spaces are convex and in which no two segments meet at an angle less than 90°, for $\bar{\rho} \in (\sqrt{2}, \infty)$. The inclusion is strict: the constant c approaches infinity as $\bar{\rho}$ approaches $\sqrt{2}$ from above or infinity from below, so the guarantee is most meaningful for moderate demands on triangle quality.

Size optimality does not mean that we can find the perfectly optimal mesh $\mathcal{T}$—likely a futile quest—but we can still reason about its size and prove that the size of $\mathcal{M}$ is asymptotically optimal. The reader might ask, asymptotic in relation to what? One of the most interesting theoretical discoveries about mesh generation is that there is a natural measure of how many elements are required in any high-quality simplicial mesh: the integral over the domain of the inverse squared local feature size.

The following proposition shows that this integral is an asymptotic upper bound on the number of triangles in a mesh $\mathcal{M}$ generated by Ruppert's algorithm. Subsequent propositions show that it is an asymptotic lower bound on the number of triangles in any high-quality mesh, including $\mathcal{T}$, so $\mathcal{M}$ is size-optimal. Unfortunately, the lower bound is contingent on $|\mathcal{P}|$ being convex; in its original form, Ruppert's algorithm does not offer a size-optimality guarantee for nonconvex domains. In Section 6.7, we discuss how a variant of Ruppert's algorithm that uses constrained Delaunay triangulations does offer size-optimality for nonconvex domains, with the insight that the analysis method must redefine the local feature size function to use shortest distances in $|\mathcal{P}|$.

Proposition 6.8. *Let $\mathcal{P}$ be a PLC in the plane in which no two segments meet at an angle less than* 90°. *Let $\mathcal{M}$ be a mesh of $\mathcal{P}$ generated by* D T PLC$(\mathcal{P}, \bar{\rho})$ *with $\bar{\rho} > \sqrt{2}$. Then the number of triangles in $\mathcal{M}$ is less than*

$$\frac{8(3 + 2C_S)^2}{\pi} \cdot \int_{|\mathcal{P}|} \frac{dx}{f(x)^2},$$

where C_S is a constant that depends solely on $\bar{\rho}$, defined in Proposition 6.5, and dx represents an infinitesimal measure of area in the plane.

P . Let S be the set of vertices in $\mathcal{M}$, and let $|S|$ denote the number of vertices in $\mathcal{M}$. For each vertex $v \in S$, consider the Euclidean disk $B_v = B(v, r_v)$ where $r_v = f(v)/(2 + 2C_S)$. The interiors of these disks are pairwise disjoint by Proposition 6.6. As no two segments in $\mathcal{P}$ meet each other

at an acute angle, at least one quarter of each disk is included in $|\mathcal{P}|$. Therefore,

$$\begin{aligned}
\int_{|\mathcal{P}|} \frac{dx}{f(x)^2} &> \sum_{v \in S} \int_{B_v \cap |\mathcal{P}|} \frac{dx}{f(x)^2} \\
&> \sum_{v \in S} \int_{B_v \cap |\mathcal{P}|} \frac{dx}{(f(v) + r_v)^2} \\
&\geq \frac{1}{4} \sum_{v \in S} \int_{B_v} \frac{dx}{(f(v) + r_v)^2} \\
&= \frac{1}{4} \sum_{v \in S} \frac{\pi r_v^2}{(3 + 2C_S)^2 r_v^2} \\
&= \frac{\pi}{4(3 + 2C_S)^2} |S|.
\end{aligned}$$

Recall from Section 2.1 that an $|S|$-vertex triangulation has at most $2|S| - 5$ triangles. The result follows. □

The matching lower bound on the number of triangles in a high-quality mesh depends on several observations that seem unsurprising, but require care to prove: small domain features are surrounded by proportionally small triangles; triangles that adjoin each other cannot have arbitrarily different sizes; and triangles that are distant from each other cannot have a size difference far greater than the distance between them. Together, these observations imply that the local feature size function places an upper bound on the local edge lengths in a good mesh. None of these observations is true if arbitrarily skinny triangles are allowed; all of them depend on having an upper bound on the radius-edge ratio, or equivalently, a lower bound on the smallest angle. They also require that $|\mathcal{P}|$ be convex.

The following series of propositions formalizes these observations to prepare for the lower bound proof. A triangle τ has three altitudes—the distance from a vertex of τ to the affine hull of the opposite edge; let $h(\tau)$ denote its shortest altitude. The following proposition shows that between any two disjoint simplices in a triangulation, there is a triangle whose shortest altitude does not exceed the distance between the simplices. The proposition holds for any triangulation, of good quality or not. But note that good quality implies that the triangle with a bounded altitude also has bounded edge lengths.

Proposition 6.9. *Let $\mathcal{T}$ be a triangulation in the plane. Let p and q be two points such that $pq \in |\mathcal{T}|$. Let σ_p and σ_q be the unique simplices in $\mathcal{T}$ whose relative interiors contain p and q, respectively. If σ_p and σ_q are disjoint, there is a triangle $\tau \in \mathcal{T}$ that intersects both pq and σ_p such that $h(\tau) \leq d(p, q)$.*

P . Consider three cases: σ_p is a vertex, an edge, or a triangle.

If σ_p is a vertex, namely the point p, let $\tau \in \mathcal{T}$ be a triangle adjoining p that intersects $pq \setminus \{p\}$—there are either one or two such triangles. Because p and σ_q are disjoint, the edge of τ opposite p intersects pq. Therefore, $h(\tau) \leq d(p, q)$ and the proposition holds.

If σ_p is an edge, consider several possibilities. If q is collinear with σ_p, replace p with the vertex of σ_p nearest q and apply the reasoning above. Otherwise, let $\tau_p \in \mathcal{T}$ be the triangle that has σ_p for an edge and intersects $pq \setminus \{p\}$. If $h(\tau_p) \leq d(p, q)$, the proposition holds. Otherwise,

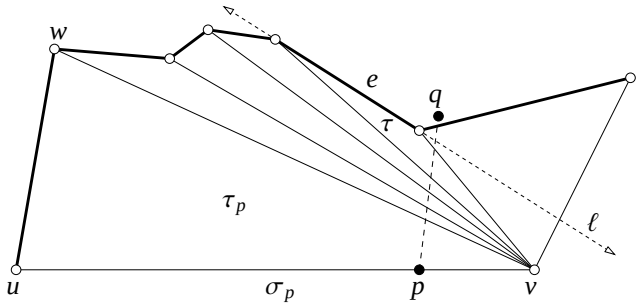


Figure 6.10: Between p and q, at least one triangle τ adjoining uv has an altitude of $d(p,q)$ or less.

let u, v, and w be the vertices of τ_p, with u and v being the vertices of σ_p. Assume without loss of generality that the plane is rotated so that σ_p is horizontal with τ_p above it, that the plane is reflected so that w lies to the left of pq, and that the vertices are labeled so u lies to the left of v, as illustrated in Figure 6.10.

Consider the fan of triangles in $\mathcal{T}$ that adjoin v and have interiors that intersect pq, starting with τ_p and proceeding in clockwise order, as illustrated. None of these triangles' interiors can contain q, because then σ_q would not be disjoint from σ_p. Therefore, the chain of edges opposite v in the fan starts with the edge uw and ends with an edge that intersects pq; these edges are bold in Figure 6.10. Because $h(\tau_p) > d(p,q)$, the apex w is higher than the point q—that is, it is further above the affine hull of σ_p. But the last edge in the chain intersects pq and so must have at least one vertex as low as q or lower. Therefore, the chain includes at least one *down edge* whose clockwise vertex is lower than its counterclockwise vertex. Let e be the most clockwise down edge in the chain, let ℓ be e's affine hull, and let $\tau \in \mathcal{T}$ be the triangle joining edge e with vertex v. Because e is the last down edge, all subsequent edges and the point q must lie above or on ℓ. Because e is a down edge, p is further from ℓ than v. It follows that $h(\tau) \le d(v,\ell) < d(p,\ell) \le d(p,q)$.

In the third and final case, σ_p is a triangle. Let p' be the point where pq intersects the boundary of σ_p, and let σ'_p be the face of σ_p whose relative interior contains p'. Replace p with p', replace σ_p with σ'_p, and apply the reasoning above. □

Any bound on the smallest angle of a triangulation imposes a limit on the grading of triangle sizes. The next proposition bounds the difference in sizes between two triangles that share a vertex. In the following propositions, let $\ell_{\max}(\tau)$ denote the length of τ's longest edge.

Proposition 6.10. *Let $\mathcal{T}$ be a triangulation in the plane with $|\mathcal{T}|$ convex. Let $\bar{\rho} = \max_{\sigma\in\mathcal{T}} \rho(\sigma)$ be the maximum radius-edge ratio among the triangles in $\mathcal{T}$; thus, $\theta_{\min} = \arcsin \frac{1}{2\bar{\rho}}$ is the minimum angle. Let τ and τ' be two triangles in $\mathcal{T}$ that share a vertex v. Then $\ell_{\max}(\tau) \le \eta\, h(\tau')$, where $\eta = (2\cos\theta_{\min})^{1+180°/\theta_{\min}} / \sin\theta_{\min} = 2\bar{\rho}(4 - 1/\bar{\rho}^2)^{0.5+90°/\arcsin 1/(2\bar{\rho})}$.*

P . Let a be the length of the longest edge adjoining the vertex v, let b be the length of the shortest, and let $\phi \le 180°$ be the angle separating the two edges. We claim that the ratio a/b cannot exceed $(2\cos\theta_{\min})^{\phi/\theta_{\min}}$. This bound is tight if $\phi/\theta_{\min}$ is an integer; Figure 6.11 offers an example where the bound is obtained.

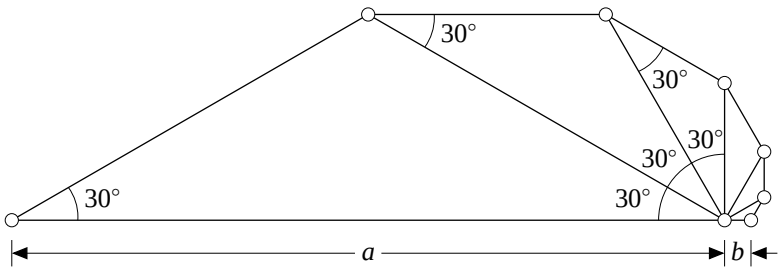


Figure 6.11: In a triangulation with no angle smaller than 30°, the ratio a/b cannot exceed 27.

We prove this claim by induction on the sequence of edges around v from the longest edge to the shortest. For the base case, suppose the longest and shortest edges belong to a common triangle. Let α and β be the angles opposite the edges of lengths a and b, respectively; then $\alpha + \beta + \phi = 180°$ and $\sin\alpha = \sin(\beta + \phi) = \sin\beta\cos\phi + \sin\phi\cos\beta$. By the Law of Sines, $a/b = \sin\alpha/\sin\beta = \cos\phi + \sin\phi/\tan\beta$, so a/b is maximized when $\beta = \theta_{\min}$. Observe that if ϕ also equals $\theta_{\min}$, then $a/b = 2\cos\theta_{\min}$. It is straightforward to verify that if $\phi > \theta_{\min}$, then $a/b < (2\cos\theta_{\min})^{\phi/\theta_{\min}}$, because the former grows more slowly than the latter as ϕ increases above $\theta_{\min}$. This establishes the base case.

If the longest and shortest edges adjoining v are not edges of a common triangle, let c be the length of an intermediate edge adjoining v. Then by the inductive hypothesis, $a/b = (a/c)(c/b) \leq (2\cos\theta_{\min})^{\angle ac/\theta_{\min}}(2\cos\theta_{\min})^{\angle bc/\theta_{\min}} = (2\cos\theta_{\min})^{\phi/\theta_{\min}}$, and the claim holds.

Because τ has two edges no longer than a and no angle smaller than $\theta_{\min}$, its longest edge satisfies $\ell_{\max}(\tau) \leq 2a\cos\theta_{\min}$. Because τ' has two edges no shorter than b and no angle smaller than $\theta_{\min}$, its shortest altitude satisfies $h(\tau') \geq b\sin\theta_{\min}$. The result follows by combining inequalities. □

The constant η in Proposition 6.10 can be improved; see Exercise 8.

The next proposition shows that in a high-quality triangulation, every triangle's longest edge has an upper bound proportional to the local feature size at any point in the triangle. Recall that Proposition 6.6 gives a proportional lower bound for the edges produced by D T PLC. Together, the two propositions show that Ruppert's algorithm generates meshes whose edge lengths are within a constant factor of the longest possible. We call this guarantee *optimal grading*.

Proposition 6.11. *Let $\mathcal{P}$ be a PLC in the plane with $|\mathcal{P}|$ convex. Let $\mathcal{T}$ be a Steiner triangulation of $\mathcal{P}$. Let $\bar{\rho} = \max_{\sigma\in\mathcal{T}}\rho(\sigma)$. Let τ be a triangle in $\mathcal{T}$. Let x be a point in τ. Then $\ell_{\max}(\tau) \leq 2\eta\, f(x)$, where η is a constant that depends solely on $\bar{\rho}$, specified in Proposition 6.10.*

P . By the definition of local feature size, the disk $B(x, f(x))$ intersects two disjoint linear cells in $\mathcal{P}$, each a vertex or edge, at two points p and q, respectively. Because p and q lie on disjoint edges or vertices in $\mathcal{P}$, they lie on disjoint edges or vertices in $\mathcal{T}$. Because $|\mathcal{P}|$ is convex, $pq \in |\mathcal{P}|$ and we can apply Proposition 6.9 to show there is a triangle $\tau' \in \mathcal{T}$ that intersects pq such that $h(\tau') \leq d(p, q)$.

If τ adjoins τ', then $\ell_{\max}(\tau) \leq \eta\, h(\tau')$ by Proposition 6.10. It follows that $\ell_{\max}(\tau) \leq \eta\, d(p, q) \leq 2\eta\, f(x)$, and the claim holds.

Otherwise, let u be a point in $\tau' \cap pq$. By a second application of Proposition 6.9 to the points x and u, there is a triangle $\tau'' \in \mathcal{T}$ that adjoins τ and satisfies $h(\tau'') \leq d(x,u)$. Because u lies on pq, it lies in $B(x, f(x))$ and $d(x,u) \leq f(x)$. Therefore, $\ell_{\max}(\tau) \leq \eta\, h(\tau'') \leq \eta\, d(x,u) \leq \eta\, f(x)$, and the claim holds. □

We can now prove a lower bound on the size of a high-quality mesh.

Proposition 6.12. *Let $\mathcal{P}$ be a PLC in the plane with $|\mathcal{P}|$ convex. Let $\mathcal{T}$ be a Steiner triangulation of $\mathcal{P}$. Let $\bar{\rho} = \max_{\sigma\in\mathcal{T}} \rho(\sigma)$. The number of triangles in $\mathcal{T}$ is at least*

$$\frac{1}{\sqrt{3}\eta^2} \cdot \int_{|\mathcal{P}|} \frac{dx}{f(x)^2},$$

where η is a constant that depends solely on $\bar{\rho}$, specified in Proposition 6.10.

P . Let $\ell_{\max}(x)$ be a function that maps each point $x \in |\mathcal{P}|$ to the length of the longest edge of the triangle in $\mathcal{T}$ that contains x, taking the greatest value if more than one triangle contains x. By Proposition 6.11, $\ell_{\max}(x) \leq 2\eta\, f(x)$, so

$$\begin{aligned}
\int_{|\mathcal{P}|} \frac{dx}{f(x)^2} &\leq 4\eta^2 \int_{|\mathcal{P}|} \frac{dx}{\ell_{\max}(x)^2} \\
&= 4\eta^2 \sum_{\tau\in\mathcal{T}} \int_{\tau} \frac{dx}{\ell_{\max}(\tau)^2} \\
&= 4\eta^2 \sum_{\tau\in\mathcal{T}} \frac{\text{area}(\tau)}{\ell_{\max}(\tau)^2} \\
&\leq \sqrt{3}\eta^2 \sum_{\tau\in\mathcal{T}} 1,
\end{aligned}$$

because $\text{area}(\tau)/\ell_{\max}(\tau)^2 = h(\tau)/(2\ell_{\max}(\tau))$ attains its maximum possible value of $\sqrt{3}/4$ for an equilateral triangle. The summation is the number of triangles in $\mathcal{T}$, so the claim follows. □

Propositions 6.8 and 6.12 together establish the size optimality of meshes produced by D - T PLC, formally stated in the following theorem.

Theorem 6.13. *Let $\mathcal{P}$ be a PLC in the plane such that $|\mathcal{P}|$ is convex and no two segments in $\mathcal{P}$ meet at an angle less than 90°. Let $\bar{\rho}$ be a real number greater than $\sqrt{2}$.* D T PLC *produces a mesh $\mathcal{M}$ of $\mathcal{P}$ whose triangles' radius-edge ratios do not exceed $\bar{\rho}$, such that the number of triangles in $\mathcal{M}$ is at most a constant factor greater than the number of triangles in any other Steiner triangulation of $\mathcal{P}$ whose radius-edge ratios do not exceed $\bar{\rho}$.*

6.6 Meshing domains with small angles

Ruppert's algorithm requires that no two segments meet at an acute angle. This is a severe restriction. In practice, the algorithm often succeeds despite acute angles, but as domain angles drop below about 45°, it becomes increasingly likely to fail to terminate.

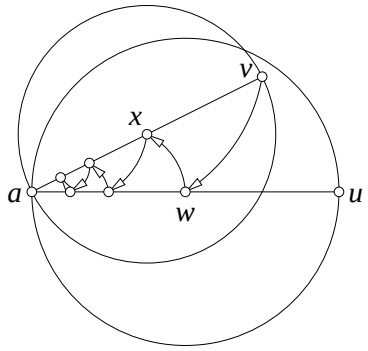


Figure 6.12: Ping-pong encroachment caused by a small input angle. Vertex *v* encroaches upon *au*, which is split at *w*. Vertex *w* encroaches upon *av*, which is split at *x*, which encroaches upon *aw*, and so on.

Figure 6.12 demonstrates one difficulty caused by small input angles. If two adjoining segments have unequal lengths, an endless cycle of mutual encroachment may produce ever-shorter subsegments incident to the apex of the small angle. This phenomenon, sometimes called *ping-pong encroachment*, is observed only with angles of 45° or less.

Sometimes it is impossible to obtain good element quality. If two segments of a domain adjoin each other at a 1° angle, some triangle of the final mesh will have an angle of 1° or less. Moreover, a small domain angle sometimes necessitates generating elements with new small angles that are not inherited from the domain (recall Figure 1.5). Given a domain with small angles, a mesh generator must diagnose where it is necessary to give up and accept some poor-quality elements.

This section discusses two modifications to Ruppert's algorithm that extend it so it works remarkably well with domains that have small angles. The first modification was proposed by Ruppert himself; he calls it "modified segment splitting using concentric circular shells." The second modification is a simple observation about which skinny triangles the mesh generator should not try to split. Together, these two modifications yield a variant of Ruppert's algorithm that always terminates and has some impressive properties. Most importantly, it can guarantee that no triangle has an angle greater than 138.6°. It also guarantees that skinny triangles appear only between segments separated by small angles. Recall that for many applications, bounding the largest angles is more important than bounding the smallest angles, because the former are related to the discretization and interpolation errors.

The first modification is to split some encroached subsegments off-center, rather than at their midpoints. Imagine that each input vertex is enclosed by concentric circles whose radii are all the powers of two—that is, 2^i for all integers i, as illustrated in Figure 6.13. When an encroached subsegment adjoins another segment at an angle less than 90°, split the subsegment not at its midpoint, but at one of the circular shells centered at the shared vertex, so that one of new subsegments has a power-of-two length. Choose the shell that gives the best-balanced split, so the two new subsegments produced by the split are between one-third and two-thirds the length of the split subsegment.

If both vertices of a segment adjoin other segments, the segment may undergo up to two unbalanced splits—one for each end. Choose one vertex arbitrarily, and split the segment so the

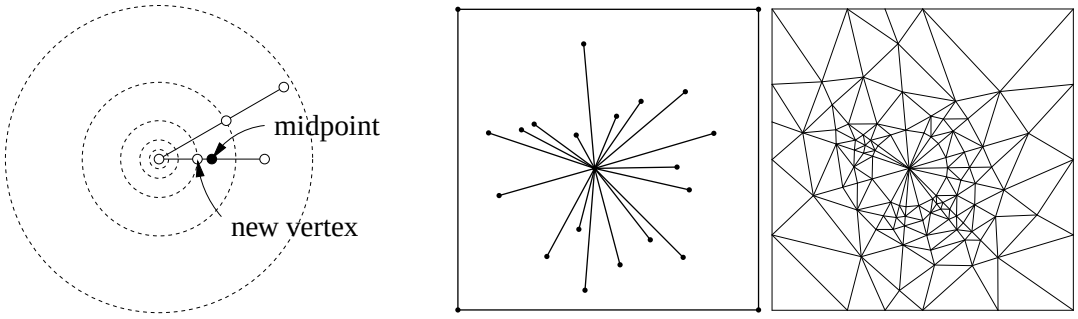


Figure 6.13: Concentric circular shells appear at left. If an encroached subsegment meets another segment at an acute angle, the subsegment is split at its intersection with a circular shell whose radius is 2^i for some integer i. The illustrations at right are a sample input and output of Ruppert's algorithm with concentric shell segment splitting.

subsegment adjoining that vertex has a power-of-two length between one-quarter and one-half the length of the split subsegment. The other subsegment produced by this split might undergo a subsequent off-center split, in which case all three subsegments will be at least one-fifth the length of the original segment. All subsequent subsegment splits are bisections.

Concentric shell segment splitting prevents the runaway cycle of ever-shorter subsegments portrayed in Figure 6.12, because adjoining subsegments of equal length do not encroach upon each other. Ruppert also suggests changing his algorithm so that it does not attempt to split a skinny triangle nestled in the corner of a small input angle. These changes are often effective, as the mesh at right in Figure 6.13 shows, and they always suffice for simple polygons with no internal boundaries.

However, Figure 6.14 illustrates a more treacherous way by which small input angles and internal boundaries can cause Delaunay refinement to fail to terminate. Recall the key idea that Delaunay refinement should create no new edge that is shorter than the shortest edge previously existing. If two subsegments that adjoin each other at a very small angle are bisected, the new edge connecting their two midpoints can starkly violate this rule. The new, shorter edge can cause subsequent refinement as the algorithm removes skinny triangles, as illustrated, which can cause the subsegments to be split again, creating a yet shorter edge, and the cycle may continue forever.

An idea that breaks this cycle is to deny these new, unduly short edges the privilege of causing further refinement. Specifically, call an edge *seditious* if its vertices lie on two distinct segments that meet each other at an angle less than 60°, the two vertices lie on the same concentric shell, and the two vertices are true midpoints (not off-center splits), as illustrated in Figure 6.14.

The second modification is to simply decline to try to split any skinny triangle whose shortest edge is seditious. This precaution prevents the short lengths of seditious edges from propagating through the mesh. Triangles with small angles can survive, but only between segments adjoining each other at small angles. Figure 6.14 depicts a mesh generated by the modified algorithm for a PLC that requires both modifications to stop the algorithm from refining forever.

The observation behind why this modified algorithm terminates is that unduly short edges—edges shorter than those predicted by Proposition 6.4—can be created in only two circumstances. Off-center subsegment splits can create them, but only twice per PLC segment. Unduly short

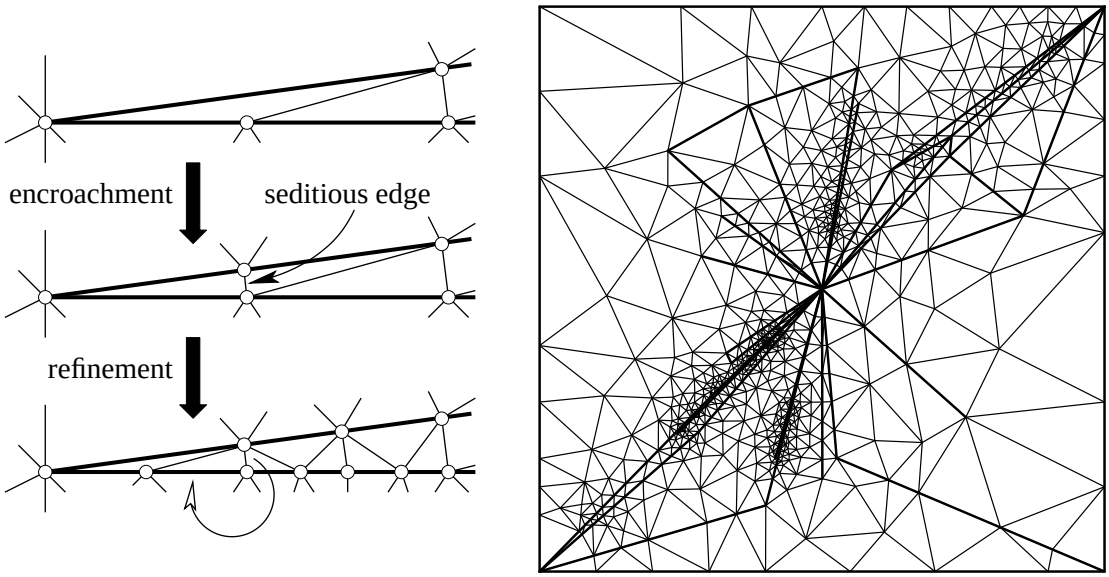


Figure 6.14: At left, a demonstration of how segments separated by small angles create short, seditious edges as they are split; the refinement of skinny triangles can cause the subsegments to be split again. At right, a mesh generated by Ruppert's algorithm with concentric shells when it declines to split triangles whose shortest edges are seditious. No angle in this mesh is greater than 127.1°, and no triangle has an angle less than 26.45° unless its shortest edge is seditious.

edges are also created by cascading bisections of adjoining segments, as illustrated in Figure 6.14, but these edges are all seditious, and are prevented from causing further refinement.

Proposition 6.14. *Let $\bar{\rho} > \sqrt{2}$ be the maximum permitted radius-edge ratio of a triangle whose shortest edge is not seditious. If Ruppert's algorithm is modified to use concentric shells for segment splitting and to decline to try to split any triangle whose shortest edge is seditious, it is guaranteed to terminate for any two-dimensional PLC $\mathcal{P}$, with no restrictions on the angles at which segments meet. Moreover, no triangle of the final mesh has an angle greater than* $180° - 2\arcsin\frac{1}{2\bar{\rho}}$ *nor an angle less than* $\sin\phi_{\min}/\sqrt{5 - 4\cos\phi_{\min}}$*, where $\phi_{\min}$ is the smallest angle separating two adjoining segments in $\mathcal{P}$.*

P . Let $\mathcal{P}'$ be a copy of the PLC $\mathcal{P}$ modified to include every off-center vertex the algorithm inserts on a segment; i.e. each vertex that is not the true midpoint of the subsegment being split. The segments in $\mathcal{P}'$ are subdivided accordingly. At most two off-center splits occur for each segment in $\mathcal{P}$, so $\mathcal{P}'$ has only finitely many extra vertices. Let $f(\cdot)$ denote the local feature size with respect to $\mathcal{P}'$, and let $f_{\min} = \min_{x\in|\mathcal{P}'|} f(x)$.

Consider a group of segments that meet at a common vertex z in $\mathcal{P}'$, with consecutive segments in the group separated by angles less than 60°. These segments have power-of-two lengths (possibly excepting some segments that will never be split, which we can ignore). When they are refined, they are split at their true midpoints, so their subsegments have power-of-two lengths. At any time during refinement, if the shortest subsegment of the segments in the group has length 2^i, then all the vertices on the segments lie on circular shells centered at z of radii $j \cdot 2^i$ for positive

integers j, and two vertices can be separated by a distance less than 2^i only if they lie on the same shell. If a vertex v on one segment encroaches upon a subsegment e of another subsegment in the group, then e crosses the shell that v lies on, and the two subsegments created when e is split cannot be shorter than the two subsegments adjoining v. It follows that a cascading chain of mutual encroachments solely within the group does not create a subsegment shorter than the shortest subsegment already in the group.

Say that a pair of vertices (v, x) is *seditious* if they lie on two distinct segments in $\mathcal{P}'$ that meet each other at an angle less than 60° at some vertex z, and they lie on the same shell; that is, $d(z, v) = d(z, x)$. Thus vx is a seditious edge if the mesh contains it. We claim that the modified algorithm never generates two vertices separated by a distance less than $f_{\min}$ unless they are a seditious pair. Suppose for the sake of contradiction that v is the first vertex inserted that breaks this invariant. Then there is a vertex x such that (v, x) is not seditious but $d(v, x) < f_{\min}$. Let w be the vertex nearest v at the moment v is inserted. Then $d(v, w) \leq d(v, x) < f_{\min}$. It is possible that w and x are the same vertex.

We claim that v is neither a circumcenter nor a type 1 vertex whose parent is a rejected circumcenter. If v is the circumcenter of a skinny triangle τ, then τ's shortest edge has length at least $f_{\min}$ because the algorithm does not split a triangle whose shortest edge is seditious, and by the inductive hypothesis, all the nonseditious edges had length $f_{\min}$ or greater before v was inserted. But τ's radius-edge ratio exceeds $\bar{\rho}$, so its circumradius is greater than $\bar{\rho} f_{\min} > \sqrt{2} f_{\min}$ and thus $d(v, w) > \sqrt{2} f_{\min}$, a contradiction. If v is a type 1 vertex whose parent p is a rejected circumcenter, then $r_p > \sqrt{2} f_{\min}$ by the same reasoning, so Proposition 6.4(iii) implies that $r_v \geq r_p/\sqrt{2} > f_{\min}$. Then $d(v, w) \geq r_v > f_{\min}$, a contradiction.

Therefore, v is a type 1 vertex inserted on an encroached subsegment e of a segment s, and the diametric disk of e contains some encroaching vertex, and hence contains w. Thus w is not a circumcenter (which would have been rejected). The fact that $d(v, w) < f_{\min}$ implies that w is not a vertex in $\mathcal{P}'$ and does not lie on a segment disjoint from s. The same is true for x. Therefore, w lies on a segment in $\mathcal{P}'$ that adjoins s at a shared vertex z. By our reasoning above, the two subsegments created when e is split are not shorter than the two subsegments adjoining w, which have lengths of at least $f_{\min}$ by the inductive hypothesis. But $d(v, x) < f_{\min}$, so x is in e's diametric disk. Thus x, like w, lies on a segment s' that adjoins s at z, and adjoins two subsegments whose lengths are at least $f_{\min}$. The fact that $d(v, x) < f_{\min}$ implies that v and x lie on a common circular shell. The radius of that shell is at least $f_{\min}$, so s and s' meet at an angle less than 60°. This contradicts our assumption that (v, x) is not seditious. It follows that only seditious pairs can be separated by a distance less than $f_{\min}$.

Because every subsegment has a length of at least $f_{\min}$, every seditious edge has a length of at least $2 f_{\min} \sin \frac{\phi_{\min}}{2}$. It follows from the Packing Lemma (Lemma 6.1) that the modified algorithm terminates.

When the algorithm terminates, every triangle whose shortest edge is not seditious has no angle less than $\arcsin \frac{1}{2\bar{\rho}}$, and thus no angle greater than $180° - 2 \arcsin \frac{1}{2\bar{\rho}}$. To bound the angles of the other triangles, consider the seditious edge wx in Figure 6.15. Its vertices lie on two distinct segments that meet at a vertex v at an angle $\phi < 60°$, and the vertex x is a true midpoint of vy. If a triangle whose shortest edge is wx respects the segments, its largest angle cannot exceed $\angle wxy = 90° + \phi/2 < 120°$, which establishes our claim about the largest angles.

Let $\psi = \angle xyw$, and observe that $\psi < \phi$. No Delaunay triangle with shortest edge wx can

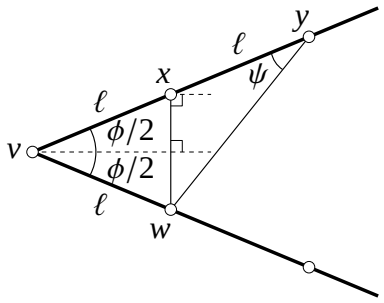


Figure 6.15: If a triangle's shortest edge wx is seditious and subtends an input angle ϕ, the triangle has no angle greater than $90° + \phi/2$ nor less than ψ.

have an angle less than ψ, because by the Inscribed Angle Theorem, any such triangle would have either y or v inside its circumdisk. By the Law of Sines, $\sin\psi/d(v,w) = \sin\angle vwy/d(v,y)$, hence

$$2\sin\psi = \sin\angle vwy = \sin(180° - \phi - \psi) = \sin(\phi + \psi) = \sin\phi\cos\psi + \cos\phi\sin\psi.$$

Therefore, $(2 - \cos\phi)^2\sin^2\psi = \sin^2\phi\cos^2\psi = \sin^2\phi(1 - \sin^2\psi)$. Rearranging terms gives $\sin\psi = \sin\phi/\sqrt{5 - 4\cos\phi}$, which establishes our claim about the smallest angles. □

Proposition 6.14 guarantees termination, but not good grading. It is possible to salvage a weakened proof of good grading; see the biographical notes for details.

In a practical implementation, it is wise to use an inter-segment angle smaller than 60° to define seditious edges, so that Delaunay refinement is less tolerant about leaving skinny triangles behind. This change breaks the termination proof, but in practice it threatens termination only if the angle threshold for seditious edges is substantially smaller than 20°. (The algorithm must still decline to try to split triangles that are right in the corners of small domain angles, of course, as these cannot be improved.)

6.7 Constrained Delaunay refinement

If software for constructing and updating constrained Delaunay triangulations is available, Ruppert's algorithm is easily modified to construct and maintain a CDT instead of $\mathrm{Del}\,S$, and it enjoys several advantages by doing so. First, the algorithm stores no triangles outside the domain, even if $|\mathcal{P}|$ is not convex, and therefore saves the costs of maintaining them and checking which triangles are in the domain. Second, every subsegment is an edge of the CDT; whereas Ruppert's original algorithm sometimes must pay for point location to insert the midpoint of a subsegment that is absent from $\mathrm{Del}\,S$, constrained Delaunay refinement requires no point location, because every newly inserted vertex is associated with a mesh edge or triangle. Third, and most important, CDTs prevent overrefinement that can occur where geometric features are separated by small distances exterior to the domain, as illustrated in Figure 6.16. As the mesh at left shows, Ruppert's original algorithm with a bounding box can refine a mesh much more than necessary, because of encroachments and skinny triangles exterior to the domain. A CDT prevents this overrefinement, as the mesh at right illustrates.

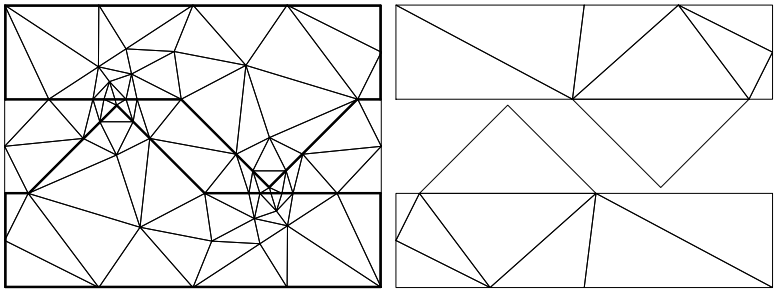


Figure 6.16: Two variations of Ruppert's Delaunay refinement algorithm with a 20° minimum angle. Left: Overrefinement with a Delaunay triangulation in a box. Right: Refinement with a constrained Delaunay triangulation.

A nuisance in Section 6.5 is that the proof of size optimality holds only if $|\mathcal{P}|$ is convex. Figure 6.16 shows that this is not merely a technical flaw in the proofs; Ruppert's algorithm does not always generate size-optimal meshes of nonconvex domains. The local feature size does not distinguish exterior distances from interior distances, so it correctly predicts the behavior of Ruppert's algorithm, but it is not an accurate estimate of the longest possible edge lengths.

Once modified to maintain a CDT, Ruppert's algorithm is size-optimal even for nonconvex domains. We can prove this by replacing the Euclidean distance with the *intrinsic distance* between two points—the length of the shortest path connecting the points that lies entirely in $|\mathcal{P}|$. An intrinsic path must go around holes and concavities. Redefine the local feature size $f(x)$ at a point x to be the smallest value such that there are two disjoint linear cells in $\mathcal{P}$ within an intrinsic distance of $f(x)$ from x. The edge lengths in the mesh at right in Figure 6.16 are locally proportional to this modified local feature size. It is a tedious but straightforward exercise to show that the proofs in Sections 6.4 and 6.5 all hold for Ruppert's algorithm with a CDT and the intrinsic local feature size, without the assumption that $|\mathcal{P}|$ is convex.

Consider a different meshing problem: to produce a triangular mesh of a piecewise linear complex in three-dimensional space with no 3-cells, composed of polygons meeting at shared segments. The polygon triangulations must conform to each other—that is, match triangle edge to triangle edge—along their shared boundaries. This problem arises in boundary element methods for solving partial differential equations and in global illumination methods for computer graphics.

The constrained Delaunay refinement algorithm can solve this problem, with no conceptual changes, by meshing all the surfaces simultaneously. Again, the key is to define the local feature size in terms of intrinsic distances in the underlying space of the PLC. Where polygons meet at shared segments, features in one polygon may affect the local feature size in another, reflecting the fact that the refinement of one polygon can propagate into an adjoining polygon by splitting their shared segments.

6.8 Notes and exercises

For the sake of establishing precedence, we note that Ruppert's 1995 article [180] is his fullest presentation of his algorithm and its analysis, but earlier versions appeared in 1992 and 1993 [178, 179]. In 1993, Chew [61] independently discovered a very similar Delaunay refinement algorithm that guarantees a minimum angle of 30°. Unlike his 1989 algorithm described in Section 1.2, his 1993 algorithm offers optimal grading and size optimality for any angle bound less than 26.5°—compared to 20.7° for Ruppert's algorithm—although this property was proven not by Chew, but subsequently by Shewchuk [201]. The improved angle bounds are obtained by using a constrained Delaunay triangulation and a more conservative procedure for treating encroached subsegments. Miller, Pav, and Walkington [147] reanalyze Ruppert's original algorithm and extend its angle guarantee to 26.4°, with size optimality and optimal grading intact. The same techniques show that Chew's algorithm guarantees size optimality and optimal grading up to an angle guarantee of 28.6°. For angle bounds between 28.6° and 30°, Chew's algorithm is guaranteed to terminate but is not guaranteed to produce a graded or size-optimal mesh.

Most of the analysis in this chapter is taken from Ruppert's article, but the proof of Proposition 6.10 is adapted from Mitchell [150] and the proof of Proposition 6.9 is new. The idea to analyze Delaunay meshing algorithms in terms of the radius-edge ratio comes from Miller, Talmor, Teng, and Walkington [148]. The first size-optimality proof for a mesh generation algorithm was given by Bern, Eppstein, and Gilbert [18] for their provably good quadtree mesher, and their quadtree box sizes are a forerunner of Ruppert's local feature size. Mitchell [150] gives a stronger lower bound on the number of triangles in a high-quality triangulation that shrinks proportionally to $\theta_{\min}$ as $\theta_{\min}$ approaches zero, whereas the bound given here shrinks proportionally to $2^{-O(1/\theta_{\min})}$.

The idea to place new vertices at triangle circumcenters originates in a 1987 paper by William Frey [100], who appears to be the first to suggest using the Delaunay triangulation to guide vertex placement, rather than generating all the vertices before triangulating them. Circumcenters are not always the optimal locations to place new vertices. If a skinny triangle's circumcircle is substantially larger than its shortest edge, it is often better to place the new vertex closer to the short edge, so they form an acceptable new triangle. The effect is to make Delaunay refinement behave like an advancing front method. Frey [100] and Üngör [219] report that these *off-centers* give an excellent compromise between the quality and the number of triangles, and Üngör also shows that Ruppert's theoretical results remain true for properly chosen off-centers. A more aggressive algorithm of Erten and Üngör [94] optimizes the placement of a new vertex in a circumcircle; it often generates triangular meshes that have no angle smaller than 41°.

The idea to recover a boundary in a Delaunay triangulation by repeatedly inserting vertices at missing portions of the boundary originates in a 1988 paper by Schroeder and Shephard [188], who named this process *stitching*.

The suggestion to use concentric circular shells for segment splitting comes from Ruppert's original paper. The idea to decline to split triangles with seditious edges, as described in Section 6.6, is a slight variation of an algorithm of Miller, Pav, and Walkington [147]. Pav [167] proves that their algorithm offers good grading. The software T [1] implements Ruppert's algorithm, Chew's 1993 algorithm, Üngör's off-centers, and the modifications for domains with small angles discussed in Section 6.6. Chapters 9 and 15 address the difficulty of meshing three-

[1] http://www.cs.cmu.edu/~quake/triangle.html

dimensional domains with small angles.

Another important extension of Ruppert's algorithm is to domains with curved boundaries. The first such extension, by Boivin and Ollivier-Gooch [32], uses CDTs to aid the recovery of curved ridges in triangular meshes. A more recent algorithm by Pav and Walkington [166] can handle cusps—curved ridges that meet at an angle of zero. This algorithm maintains and returns a true Delaunay triangulation.

Yet another important extension is to generate anisotropic meshes. Practical Delaunay mesh generators adapt easily to anisotropy; for instance, George and Borouchaki [103] modify the Bowyer–Watson algorithm to use circumellipses instead of circumspheres. However, if the anisotropy field varies over space, the newly created elements might not have empty circumellipses, and the quality of the mesh cannot be guaranteed. Labelle and Shewchuk [127] propose a provably good algorithm for anisotropic triangular mesh generation that introduces anisotropic Voronoi diagrams to provide a foundation for the mathematical guarantees.

The fact that Ruppert's algorithm might perform work quadratic in the size of the final mesh was first noted by Ruppert in an unpublished manuscript. Barbič and Miller [15] work out an example in detail. Har-Peled and Üngör [109] describe a Delaunay refinement algorithm, essentially Ruppert's algorithm with off-centers, that uses a quadtree to help it run in optimal $O(n \log n + N)$ time, where n is the number of input vertices and N is the number of output vertices. See also the discussion of *sparse Voronoi refinement* by Hudson, Miller, and Phillips [115] in the Chapter 8 notes, which achieves virtually the same running time without the need for a quadtree.

Exercises

1. Show that an edge $e \in \operatorname{Del} S$ is encroached if and only if $\operatorname{Del} S$ contains a triangle that has e for an edge and a nonacute angle ($\geq 90°$) opposite e. Therefore, a subsegment's encroachment can be diagnosed in $O(1)$ time.

2. Suppose D T PLC takes as input a PLC in which some segments meet each other at angles less than 90°, but never less than 60°. Then Proposition 6.4 no longer suffices, because a vertex inserted on one segment might encroach upon a subsegment on an adjoining segment. Show that nonetheless, D T PLC must terminate.

3. Suppose D T PLC enforces a stricter standard of quality for triangles that do not intersect the relative interior of a segment: a triangle that does not intersect a segment interior is split if its radius-edge ratio exceeds 1. A triangle that intersects a segment interior is split if its radius-edge ratio exceeds $\sqrt{2}$. Given a PLC in which no two segments meet at an angle less than 90°, show that D T PLC still must terminate.

4. Ruppert's algorithm splits encroached subsegments at their midpoints. We sometimes achieve smaller meshes if we use off-center splits when the encroaching vertex is not a rejected circumcenter. For example, if an input vertex v encroaches upon a subsegment e, and v is very close to e but not to e's midpoint, then splitting e off-center might reduce the number of triangles in the final mesh. One idea is to project v orthogonally onto e. Unfortunately, this idea might create an unreasonably short edge, as Figure 6.17 illustrates. Explain

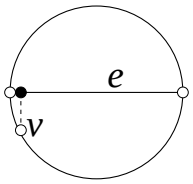


Figure 6.17: Projecting an encroaching vertex onto an encroached subsegment can create a dangerously short edge.

how to modify this idea so that Proposition 6.4 still holds, while still splitting subsegments as close to the projected point as possible.

5. If two segments meet each other at an angle of 45° or less, D T PLC may fail to terminate because of ping-pong encroachment, illustrated in Figure 6.12. However, suppose we modify D T PLC by eliminating step 4 so skinny triangles are ignored and changing step 3 so that only subsegments missing from Del S are considered encroached:

   3. While some subsegment $e \in E$ is missing from Del S, call S S (e, S, E), update Del S, and repeat step 3.

   Let $\mathcal{P}$ be a PLC in which no four segments meet at a single vertex. Show that this modified D T PLC always produces a Steiner Delaunay triangulation of $\mathcal{P}$, no matter how many small angles it has.

6. Suppose that we modify step 3 of D T PLC as described in the previous exercise, but instead of eliminating step 4, we replace it with the following.

   4. While Del S contains a triangle τ for which $\rho(\tau) > \bar{\rho}$ and the circumcenter c of τ does not encroach upon any subsegment in E, insert c into S, update Del S, and go to step 3.

   In this modified algorithm, skinny triangles may survive, but only near the domain boundaries. Quantify the minimum quality of the triangles relative to their proximity to the domain segments in terms of their edge lengths and $\bar{\rho}$.

7. Show that if the local feature size function is modified to use intrinsic distances, it is still 1-Lipschitz, i.e. it still satisfies Proposition 6.3.

8. Show that if $\theta_{\min} \leq 30°$, the inequality in Proposition 6.10 can be improved so that $\eta = (2\cos\theta_{\min})^{180°/\theta_{\min}} / \sin\theta_{\min}$. Hint: This expression follows immediately if the longest edge of τ adjoins v. If τ's longest edge does not adjoin v, what is the angle separating the two edges of τ that do adjoin v?

9. Consider the problem of meshing the PLC $\mathcal{P}$ illustrated in Figure 1.5, which includes two adjoining segments that are separated by a very small angle ϕ. Clearly, it is impossible to avoid placing a triangle with angle ϕ or less at the small domain angle. But ideally, the mesh would have no other angle less than 30°. With help from Proposition 6.10, show that for sufficiently small ϕ, no such Steiner triangulation of $\mathcal{P}$ exists.

10. Prove that Ruppert's algorithm with "modified segment splitting using concentric circular shells" always terminates for domains that are simple polygons with no internal boundaries, even if it splits skinny triangles whose shortest edges are seditious, so long as it declines to split triangles nestled right in the corners of small domain angles.

11. Prove that D T PLC maintains a quarantined complex (see Chapter 7) when it inserts a circumcenter of a triangle.