Safe Steiner Points for Delaunay Refinement

Benoît Hudson[1]

Toyota Technological Institute at Chicago
bhudson@tti-c.org

Summary. In mesh refinement for scientific computing or graphics, the input is a description of an input geometry, and the problem is to produce a set of additional "Steiner" points whose Delaunay triangulation respects the input geometry, and whose points are well-spaced. Ideally, we would like the minimum output size – as few Steiner points as possible. Ruppert showed how to come within a constant factor of optimal in two dimensions, and a number of techniques have been proposed to extend his work to various settings (higher dimension, curved features, fast runtime, parallelism, etc). Each extension needs a different choice of Steiner point, and thus its own tedious proof of the sizing guarantee. In this article, I show how to classify Steiner points as safe or potentially unsafe; any algorithm that chooses only safe points achieves the sizing guarantee. Furthermore, I show that it is easy to classify as safe the Steiner points chosen in most prior work. This work frees future meshing researchers to more easily consider varying the choice of Steiner points to achieve important new properties.

1 Introduction

The Delaunay refinement method [Che89, Rup95, She96, She98] sketched in Figure 1 is one of the main approaches for mesh refinement, because it sits at a happy confluence of theoretically provable qualities and good behaviour in practice. The method starts by computing a Delaunay triangulation, and incrementally adds vertices and retriangulates until the mesh being maintained has the desired properties. According to Shewchuk [She98]: *The central question of any Delaunay refinement algorithm is "where should the next vertex be inserted?"* He gives what he calls a "reasonable" answer, but it is not the only one: indeed, many prior and subsequent approaches describe different reasonable answers, and provide provable guarantees that they work. One proof in particular, pioneered by Ruppert, shows that the algorithm terminates with a small mesh. This proof recurs in nearly all Delaunay refinement articles, with only minor adjustments. It is typically two pages long. My goal here is to allow future meshing researchers to reproduce the termination proof very quickly — in a few sentences. This will allow the meshing community to more easily develop new algorithms, and leave us more room to describe their novel features without surrendering any mathematical rigor.

DelaunayRefinement($\mathcal{X}$, sf, ρ, $\mathcal{K}$)
1: Initialize a Delaunay triangulation T
2: **while** $\exists t \in T$ that is "bad" **do**
3: Choose a point p
4: Add p to the mesh and retriangulate
5: **end while**
6: **return** T

Fig. 1: A sketch of the Delaunay refinement technique: Given an input geometry $\mathcal{X}$, a spacing function sf, and a quality requirement ρ, and perhaps some other constants $\mathcal{K}$, iteratively add points until the Delaunay triangulation respects the geometry, conforms to the spacing function, and contains only good-quality elements.

Ruppert's algorithm [Rup95], extended to three dimensions by Shewchuk [She98], was stated with very specific choices for the main steps: (1) the initial triangulation is the Delaunay triangulation of the input points; (2) a triangle or tetrahedron is skinny if its *radius-edge ratio* — the ratio of the radius of its circumscribing ball to the length of its shortest edge — is larger than a user-specified constant ρ; (3) when a triangle or tetrahedron is skinny, we must insert the center of its circumscribing ball (its *circumcenter*), unless that point is close to a facet of the input geometry, in which case we must follow a specific procedure to snap the point to that facet. See Section 9.1 for details. This allowed them to produce a proof that the loop is guaranteed to terminate with a good quality mesh in which each input feature appears as the union of a set of facets in the output mesh. Furthermore, it outputs a mesh of bounded size: no more than a constant factor larger than is optimal.

Much subsequent work has largely kept this framework in mind, but changed the definition of each step in order to prove ever-better qualities or handle more general inputs. Boivin and Ollivier-Gooch show how to adapt Ruppert's algorithm for some curved inputs [BOG02]; there have recently been several results on meshing piecewise smooth inputs in higher dimension [ORY05, RY07, CDL07]. Üngör recommends using the *off-center* [Üng04], a point other than the circumcenter: in practice, this nearly halves the output size without affecting the provable guarantees. Har-Peled and Üngör [HPÜ05] show how to achieve fast asymptotic runtime when meshing point clouds by carefully using off-centers. Hudson, Miller, and Phillips [HMP06] achieve fast runtime on a broader class of inputs by starting with a very coarse mesh, then alternating between inserting input and Steiner points. Refinement can also be done in parallel: the question becomes to find a *set* of points to insert simultaneously [STÜ07, STÜ04, HMP07].

Each article cited above must prove that the bound on the output size still holds. There are three main approaches: showing that someone else's prior proof applies, spending approximately two tedious pages on the proof, or eliding the proof for lack of space. The first approach is, of course, the ideal. Chernikov and Chrisochoides [CC06, CC07] have studied one generalization. The present work provides a much broader generalization, in the hopes of allowing far more radical departures from tradition.

Contributions

I describe simple rules for categorizing a Steiner point as safe. The rules allow a meshing algorithm to produce a mesh that has good *quality* (see Section 2), and that *respects* the input geometry (Section 3) and an optional user-supplied mesh sizing field. In addition, the rules allow perturbation, either on purpose or as a model of numerical error during mesh construction. Finally, they allow for choosing sets of points in parallel. The rules appear in Section 5. I prove that an algorithm that follows the rules and inserts only safe sets of points terminates (Section 6). I discuss the size of a mesh made of safe points: it is within a constant factor of optimal in some settings, and as small as we know how to produce in other settings (Section 7). Finally, I give examples of how to prove an algorithm uses safe points in Section 9.

2 Quality measures

In meshing for the finite element method, the usual quality criterion we wish to achieve is that all the simplices have good aspect ratio. The typical way to provably achieve good aspect ratio is to start by producing a mesh with good *radius-edge ratio*, and to modify it by either adding a small number of additional points [LT01] or by computing a weighted Delaunay triangulation [CD03]. The radius-edge ratio of a simplex compares the radius r of the smallest ball that circumscribes the simplex and the length e of the shortest edge of the simplex; we say that the simplex has good radius-edge ratio if $r/e \leq \rho$ for some constant ρ. Theorem 1 describes the legal values of ρ. In the proofs, it is easier to make statements about the dual Voronoi diagram, so henceforth I use the following, closely-related definition of quality:

Definition 1. *Let $\mathrm{NN}(v)$ be the distance from a vertex v to its nearest neighbour in the mesh. Let $R(v)$ be the distance from v to the farthest node of its Voronoi cell — equivalently, $R(v)$ is the largest circumradius of any Delaunay simplex that has v as one of its vertices. The **quality** of v is the ratio $R(v)/\mathrm{NN}(v)$. We say that v is ρ-**good** if $R(v) \leq \rho\,\mathrm{NN}(v)$.*

The vertex quality defined here and the more traditional radius-edge ratio are nearly equivalent: If a simplex has radius-edge ratio larger than ρ, then there is a vertex of that simplex with quality worse than ρ. Conversely, if a vertex has quality worse than ρ, then there is a simplex on that vertex with radius-edge ratio worse than some constant ρ'.

3 Input Geometry

I follow recent practice and allow the input geometry to be quite general: the input can be a complex including curves and manifold surfaces.

Definition 2. *A **piecewise manifold complex** $\mathcal{X}$ is a finite set of compact manifolds, plus the empty set, with the following properties:*

- *For any manifold $f \in \mathcal{X}$, the boundary of f is a subset of $\mathcal{X}$.*
- *For any two manifold f and f' both in $\mathcal{X}$, $f \cap f'$ is a subset of $\mathcal{X}$.*
- *There is exactly one d-manifold, known as the **domain** Ω; all other members of $\mathcal{X}$ are submanifolds of Ω.*

For clarity, I say that an i-manifold $f \in \mathcal{X}$ is a *feature* with dimension $\dim(f) = i$. A 0-manifold is an input point. There are only a few possible relationships between features. Consider two features f and f'. If $f \subset f'$ then f is a *subfeature* of f'. If $f \cap f' = \emptyset$, then f and f' are *disjoint*. Finally, if f and f' are not disjoint, and neither is a subfeature of the other, they are *adjacent*.

The proofs in this paper require a further condition on the input: that any two adjacent features form a right or obtuse angle at their intersection. I call this a **non-acute** complex. Mesh refinement algorithms typically also require more: that features intersect generically (two adjacent i-features intersect in a set of $(i-1)$-features), and that each feature be smooth, or even linear. I do not make use of the former assumption. Some results in this paper apply only to the piecewise linear case; I will make it clear when the results are limited in this way.

In the end, we want the mesh that the algorithm outputs to *respect* the geometry. A set of vertices M (a mesh) respects a linear feature if it appears as the union of simplices in some triangulation of M, typically the Delaunay or Constrained Delaunay. For manifolds, what it means to respect the geometry is less clear. Typically, we will want each feature f to have a corresponding set F of Delaunay simplices that form a manifold. F should be homeomorphic to f, close in Hausdorff distance, and should have similar surface normals. The details are not particularly important to the present work; instead, I show that my framework applies to algorithms that are known to produce meshes that respect the geometry.

Scientific applications often need to refine some areas more than is required to respect the geometry; for example, an application may know there is an eddy in the middle of a wide flow. Therefore, the input can also specify a **spacing function**, denoted sf. The spacing function denotes that at a point $p \in \Omega$, there should be at least two mesh vertices within a distance of $\mathrm{sf}(p)$. Due to the quality requirement, this definition of spacing function is equivalent to imposing an area constraint on triangles, or an edge length constraint on simplices in general. After sliver elimination, the spacing function is also equivalent to imposing a volume constraint on simplices.

4 Local Feature Size

Ruppert [Rup95] noted a beautiful connection between a function he called the *local feature size* defined by a piecewise linear input, and the performance of his algorithm. I denote this function as the traditional local feature size $\mathrm{lfs}_{\mathrm{trad}}$. Ruppert proves two things: first, that when his algorithm inserts a vertex v, the distance $\mathrm{NN}(v)$ from v to its nearest neighbour is in $\Omega(\mathrm{lfs}_{\mathrm{trad}}(v))$. Second, that in any mesh that both respects the geometry and has bounded aspect ratio simplices, the shortest edge of the simplex that contains a point x has length in $O(\mathrm{lfs}_{\mathrm{trad}}(x))$. In other words, the

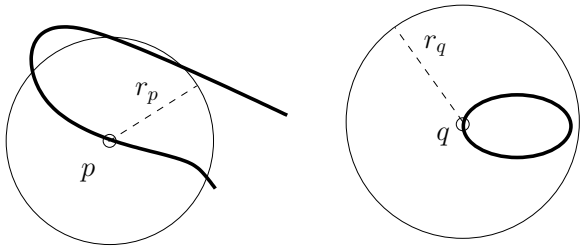


Fig. 2: Two curved features: a curve (left), and an ellipse (right). The ball $B(p, r_p)$ intersects the curve twice, so $\mathrm{lfs}(p) < r_p$. Similarly, $B(q, r_q)$ intersects the ellipse in a sphere, not a ball, so $\mathrm{lfs}(q) < r_q$.

local feature size reflects both how much refinement Ruppert's (or his successors') algorithm will produce around a point, and how much refinement even the optimal mesh must suffer.

I follow this approach, but with a new definition of local feature size to account for curved features. The traditional local feature size says that at a point x, the ball $B(x, \mathrm{lfs}_{\mathrm{trad}}(x))$ is the smallest ball that intersects at least two features f and f' that are disjoint from each other. However, consider a geometry that doubles back on itself, such as the curves illustrated in Figure 2. A useful notion of local feature size will allow for greater refinement depending on whether the curve approaches itself. To accommodate this, I change the disjointness requirement of $\mathrm{lfs}_{\mathrm{trad}}$ to instead requiring that $B(x, \mathrm{lfs}(x))$ should intersect the feature(s) in a topological ball. The quality requirement means that the size of mesh elements grows only linearly with the distance from areas with small lfs, so I also impose a 1-Lipschitz condition on lfs. More formally:

Definition 3. *Given an input piecewise smooth complex $\mathcal{X}$ and a spacing function $\mathrm{sf} : \Omega \to \mathbb{R}^+$, define the **local feature size** $\mathrm{lfs} : \Omega \to \mathbb{R}^+$ as the maximum function that satisfies the following four requirements:*

- *Consider a point p on a feature $f \in \mathcal{X}$. For all $r < \mathrm{lfs}(p)$, for all $f' \in \mathcal{X}$, the intersection of $B(p, r)$ with $f \cup f'$ must be contractible to a point.*
- *For all $p \in \Omega$, $\mathrm{lfs}(p) \leq \mathrm{sf}(p)$.*
- *lfs is 1-Lipschitz: For all p and q in Ω, $\mathrm{lfs}(p) \leq \mathrm{lfs}(q) + |pq|$.*

One unfortunate feature of this local feature size function is that it is only bounded away from zero if features meet at non-acute angles — at an acute angle, the local feature size I define is forced to zero. This is a traditional problem with all Delaunay mesh refinement algorithms; some algorithms specially handle small angles, while others fail or risk failing when faced with acute angles. Properly handling inputs with acute angles in the present framework remains a major open problem.

Constant	Use	Range	Typical values
ρ	Quality	See Theorem 1	$\sqrt{2}$, 2, or 4
δ	Perturbation	$[0, 1)$	1
γ	Spacing Function	$\gamma > 0$	∞
β	Lemmas 3 and 7	Fixed	2 or $\sqrt{2}$
d	Dimension	$\mathbb{N}$	2 or 3

Fig. 3: Constants for the safety definitions and proofs.

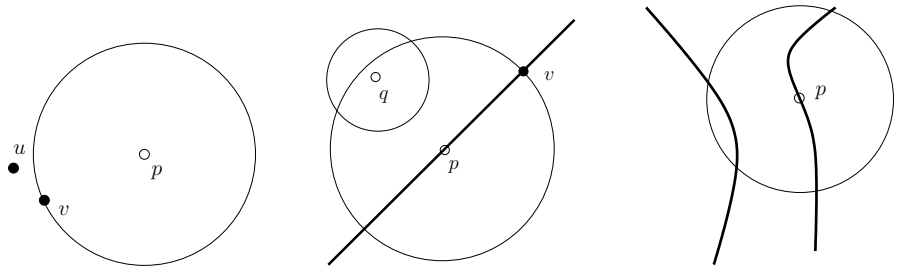
5 Safe Steiner Points

We now have the terminology to discuss where it is safe to insert Steiner points into a mesh. Fundamentally, we want to ensure that whenever an algorithm inserts a safe Steiner point p, the relation $\mathrm{NN}(p) \in \Omega(\mathrm{lfs}(p))$ holds. The definition of safety is split into three parts. First, I describe a set of conditions under which a point p is *almost safe*: inserting p would not violate the bound. Inserting an almost safe point might make it impossible to resolve the geometry without later violating the bound. To prevent this, the definition of full *safety* adds a requirement that points must be sufficiently far from input features. Finally, to support parallel algorithms, I define a notion of a *safe set* of points that can all be inserted at once.

I define safety not on points but on triples $\langle p, r, f\rangle \in \Omega \times \mathbb{R} \times \mathcal{X}$. They should be read as denoting an open ball $B(p, r)$ with radius r, centered on a point p that lies on a feature f. Ideally, the ball would contain no vertices of f, though vertices may be on its surface. To model perturbation or rounding errors, I allow vertices as close to p as δr. The safety rules refer to a number of other constants. The quality bound ρ is as described in Section 2. Almost all extant Delaunay refinement algorithms allow specifying ρ as an input parameter. Indeed, many prior investigators have noted that in practice, their algorithms terminate even with ρ smaller than the proofs allow. The constant γ allows refinement based directly on the spacing function; smaller values mean more refinement. Finally, Lemma 3 defines a constant $\beta = 2$, which I show how to improve to $\beta = \sqrt{2}$ in some circumstances in Section 8 — smaller is better. Figure 3 summarizes the constants.

Definition 4 (Almost-safe). *Let $\mathcal{X}$ be a piecewise manifold complex. Let $B(p, r)$ be a ball centered on an input feature $f \in \mathcal{X}$, but not centered on any lower-dimensional feature. The triple $\langle p, r, f\rangle$ is **almost safe** if no vertex on f, nor any 0-manifold that is a subfeature of f, lies within $B(p, \delta r)$, and also one or more of the following conditions hold (see Figure 4):*

- INPUT: *$p \in \mathcal{X}$ and $r = \infty$.*
- SPACING: *There is some $x \in B(p, r)$ such that $r \geq \gamma\,\mathrm{lfs}(x)$.*
- ENCROACHMENT: *For some $f' \in \mathcal{X}$, $B(p, r) \cap (f \cup f')$ is not contractible.*
- QUALITY: *$B(p, r)$ contains a vertex v, and $|pv| \geq \rho\,\mathrm{NN}(v)$.*
- YIELDING: *$B(p, r)$ contains both a point q that lies on f_q and a vertex v that lies on f; f is a subfeature of f_q; and q is the center of an almost-safe triple. Under these conditions, we say that q yields to p.*



(a) QUALITY (b) YIELDING (c) ENCROACHMENT

Fig. 4: Illustration of almost-safe points.

An almost-safe triple could be arbitrarily close to a feature. The QUALITY and YIELDING rules would then allow arbitrary amounts of refinement, defeating our goal of having limited refinement. Following prior work, I solve this issue by requiring that an almost-safe triple that can yield must not be inserted:

Definition 5 (Safe). *Let $\langle p, r, f\rangle$ be an almost-safe triple. The triple is safe if p does not yield to any point on a subfeature of f.*

The safety rules are all local checks, so we should not be surprised that we can apply them in parallel. I use the technique of serialization: an algorithm is allowed to insert a set of safe triples in parallel if there exists at least one sequential ordering of the insertions under which each triple is safe when it is inserted. It is more convenient to use the following geometric definition, which I prove (in Lemma 5) is equivalent:

Definition 6 (Safe Set). *Let $\mathcal{S} = \{\langle p_i, r_i, f_i\rangle, \ldots\}$ be a set of triples. We say the set is safe if $\mathcal{S}$ does not have a pair of distinct triples on the same feature, $\langle p, r, f\rangle$ and $\langle q, r_q, f\rangle$, where $|pq| < \delta \min(r, r_q)$.*

Definition 7 (Safe Meshing Algorithm). *An algorithm is a safe meshing algorithm for an input $\mathcal{X}$ if it incrementally inserts safe sets, and no other points.*

6 Termination proof

The key theorem of this paper shows that the lfs function I define acts as a lower bound on the spacing between vertices in the final output mesh of a safe algorithm, assuming that the parameters mentioned previously are appropriately set. Then the safe algorithm must terminate, since the meshing domain is compact. For most extant algorithms, all but one of the parameters is implicit in the algorithm. Then the only free parameter is the quality criterion ρ, and the theorem shows how large ρ must be for the algorithm to provably terminate. One should keep in mind that in practice, most algorithms usually terminate when ρ is set smaller than required by the proof.

Theorem 1. *Consider the output of a safe meshing algorithm with the parameters listed in Figure 3 set to constant values that satisfy*

$$\delta^d \rho > \beta^{d-1} \quad \textit{and} \quad \gamma > 0$$

Then, at any vertex v in the mesh, $\mathrm{NN}(v) \in \Omega(\mathrm{lfs}(v))$.

The proof of the theorem rests on several lemmas. I start by proving a technical detail: when a point yields, it is close to a subfeature; and when a point does not yield, it is far from any subfeature. I use this fact to prove two mutually inductive lemmas: when we propose an almost-safe triple, it is far (compared to the local feature size) from from vertices on its feature; and when we commit to inserting a truly safe point, it is far from points not only on its feature, but anywhere in the mesh. I state the lemmas here, positing the existence of a constant c and of a set of constants c_i with the property that $c \geq c_0 \geq \ldots \geq c_d$. A sceptical reader will verify that the proofs are not circular: the induction is on dimension among almost-safe triples, and on the order in which vertices are inserted.

Lemma 1 (Proposal). *Inspect the domain at any time during the run of A. Given an almost-safe triple $\langle p, r, f\rangle$ where f is an i-feature,* $\mathrm{lfs}(p) \leq c_i r$.

Lemma 2 (Commitment). *For all vertices u in the mesh at any point during the run of A, we have* $\mathrm{lfs}(u) \leq (c+1)\,\mathrm{NN}(u)$. *Immediately after the algorithm has inserted a safe vertex v, we have* $\mathrm{lfs}(v) \leq c\,\mathrm{NN}(v)$.

The Proposal and Commitment lemmas impose constraints on c and c_i. Lemma 4 shows that the set of constraints is satisfiable, assuming the parameters are chosen appropriately. Having proved it, we know that an algorithm that inserts safe vertices one by one matches the theorem's requirements. A final lemma (Lemma 5) completes the proof by showing that the bounds also apply to parallel algorithms that insert safe sets, because the points could have been safely inserted one by one.

6.1 Yielding relations

Lemma 3. *Consider an almost-safe triple $\langle p, r, f\rangle$. If it yields due to the* YIELDING *rule to an almost-safe triple $\langle q, r_q, f_q\rangle$, where f_q is a subfeature of f, then $r \leq \frac{\beta}{\delta} r_q$, with $\beta = 2$. Conversely, if it does not yield to any lower-dimensional triple, then no subfeature of f passes within $B(p, \frac{\delta}{1+\delta} r)$.*

Proof. For the first part, note that there is some vertex v on f_q within $B(q, r_q)$, and that p is also within the ball. Then $|pv| \leq 2r_q$. This vertex also lies on f; then $|pv| \geq \delta r$ and $r \leq (2/\delta) r_q$. For the second part, consider some point q on a subfeature f_q such that $|pq| = \alpha r$ for some α. There are no vertices on f (and thus none on f_q either) within $B(p, \delta r)$. If $B(q, \delta\alpha r)$ were to be empty of any vertices, then $\langle q, \alpha r, f_q\rangle$ would be almost-safe due to the YIELDING rule and p could yield. Therefore, $B(q, \delta\alpha r)$ must contain a vertex. In other words, $\alpha r + \delta\alpha r \geq \delta r$ and $\alpha \geq \frac{\delta}{1+\delta}$.

6.2 Proof of the Proposal Lemma

The Proposal Lemma itself is proved in cases, one case per rule. Each case shows that for an almost-safe triple $\langle p, r, f\rangle$ of dimension i, $\mathrm{lfs}(p) \leq c_i r$.
Case INPUT: Then r is infinite, and the statement is trivial.
Case SPACING: By definition, $r \geq \gamma \,\mathrm{lfs}(x)$ for some $x \in B(p, r)$. The Lipschitz condition bounds $\mathrm{lfs}(p) \leq \mathrm{lfs}(x) + r \leq (1 + \gamma^{-1})r$, which satisfies the lemma so long as $c_i \geq \boxed{c_d \geq 1 + \gamma^{-1}}$.
Case ENCROACHMENT: By the definition of the local feature size, $\mathrm{lfs}(p) \leq r$.
Case QUALITY: There is a vertex v in $B(p, r)$. The Lipschitz condition on lfs allows us to conclude $\mathrm{lfs}(p) \leq \mathrm{lfs}(v) + |pv| \leq \mathrm{lfs}(v) + r$. Since v has already been inserted, we can inductively apply the Commitment Lemma to see that that $\mathrm{lfs}(v) \leq (c + 1)\,\mathrm{NN}(v)$. The QUALITY rule specifies that $\mathrm{NN}(v) \leq r/\rho$; therefore $\mathrm{lfs}(p) \leq ((c + 1)/\rho + 1)r$. To satisfy the statement of the lemma, we need $c_{\dim(f)} \geq (c + 1)/\rho + 1$. Because we assumed that $c_{\dim(f)} \geq c_d$, we know that the lemma holds in the QUALITY case as long as $\boxed{c_d \geq 1 + (c + 1)/\rho}$.
Case YIELDING: Let q be the point on f_q that yields to p. We will again use the Lipschitz condition $\mathrm{lfs}(p) \leq \mathrm{lfs}(q) + |pq|$, with $|pq| \leq r$. There triple $\langle q, r_q, f_q\rangle$ is almost-safe. Because the dimension of f_q is strictly greater than the dimension of f, we can inductively assume that $\mathrm{lfs}(q) \leq c_{\dim(f_q)} r_q$. Lemma 3 shows that $r_q \leq \beta r/\delta$. Then $\mathrm{lfs}(p) \leq c_{\dim(f_q)} r_q + r < (1 + \frac{\beta}{\delta} c_{\dim(f_q)})r$. To ensure that this is less than $c_{\dim(f)} r$ as claimed, we need $1 + \frac{\beta}{\delta} c_{\dim(f_q)} \leq c_{\dim(f)}$. Generalizing over all possible dimensions of f and f_q, and remembering that $c_i \geq c_{i+1}$, we can summarize this constraint as $\boxed{c_i \geq 1 + \frac{\beta}{\delta} c_{i+1}}$ for $i \in [1, d - 1]$.

6.3 Proof of the Commitment Lemma

I now show that when an algorithm commits to inserting a safe triple $\langle p, r, f\rangle$, having previously only inserted safe points, the distance to any other vertex v currently in the mesh satisfies $\mathrm{lfs}(p) \leq c|pv|$. In particular, this holds for the nearest neighbour, which proves the lemma as previously stated. There are four cases, depending on the relation between f and the feature f_v on which v was inserted:
Case f and f_v are disjoint: Then $\mathrm{lfs}(p) \leq |pv|$.
Case f is a subfeature of f_v: When v was inserted, it did not yield to the ball $B(p, r)$. Therefore, v is not contained in that ball: $|pv| \geq r$, and $c \geq 1$.
Case f_v is a subfeature of f, or $f = f_v$: Since v lies on f, for p to be almost-safe $|pv| \geq \delta r$. The Proposal Lemma applied to p states that $\mathrm{lfs}(p) \leq c_{\dim(f)} r$, which is itself bounded by $(c_{\dim(f)}/\delta)|pv|$. Given that $\dim(f) \geq 1$, the lemma holds if $\boxed{c \geq c_1/\delta}$
Case f and f_v are adjacent: Consider the ball $b \equiv B(p, |pv|)$. If $b \cap (f \cup f_v)$ is not contractible, then $\mathrm{lfs}(p) \leq |pv|$. Otherwise, it must intersect a subfeature g common to f and f_v; the distance $|pg| \leq |pv|$. From the Proposal Lemma, we know that $\mathrm{lfs}(p) \leq c_{\dim(f)} r$. It remains to compare r to $|pg|$. If g is a 0-manifold, then

$|pg| \geq \delta r$, which bounds c as in the prior case. If g is a 1-manifold or higher (and thus f is a 2-manifold or higher), then $|pg| \geq (1+\delta)r/\delta$, as proved in Lemma 3. Therefore, $\mathrm{lfs}(p) \leq c_2 r \leq c_2 \frac{\delta}{1+\delta}|pv|$. This satisfies our requirements if $c \geq c_2 \frac{\delta}{1+\delta}$. Note that this is a strictly looser requirement than the requirement of the prior case, because $0 < \delta \leq 1$ and $c_1 \geq c_2$.

Finally, the Commitment Lemma also claims a corollary: at every vertex v of the mesh, $\mathrm{lfs}(v) \leq (c+1)\,\mathrm{NN}(v)$. Consider u, the vertex nearest v. If v was inserted after u, then we just proved that $\mathrm{lfs}(v) \leq c|uv|$. If instead u was inserted after v, then we know that at the time u was inserted, $\mathrm{lfs}(u) \leq c|uv|$. Then, the Lipschitz condition shows $\mathrm{lfs}(v) \leq \mathrm{lfs}(u) + |uv| \leq (c+1)|uv|$ as claimed.

6.4 Satisfiability

Lemma 4. *The constraints on c and c_i imposed by the Proposal and Commitment lemmas can be satisfied if $\delta^d \rho > \beta^{d-1}$ and $\gamma > 0$.*

Proof. Let us collect the constraints. From the Proposal Lemma, we know the following: The SPACING case requires that $c_d \geq \gamma^{-1}$. The ENCROACHMENT case requires $c_i \geq 1$. The QUALITY case requires $c_d \geq (1+c)/\rho + 1$. The YIELDING case requires $c_i \geq c_{i+1}\beta/\delta + 1$. We can unwind the constraints from yielding and encroachment to see that they require $c_1 \geq (\frac{\beta}{\delta})^{d-1} c_d + (d-1)\frac{\beta}{\delta}$, assuming $d \geq 2$. Finally, the Commitment Lemma requires $c \geq c_1/\delta$. Therefore, we can bound c by the constraint $c \geq (\frac{\beta^{d-1}}{\delta^d})c_d + (d-1)\frac{\beta}{\delta^2}$.

We have two lower bounds on c_d. If the tighter one is $c_d \geq 1 + \gamma^{-1}$, then c is a constant as long as γ^{-1} is a constant, which is true if γ is strictly positive. Otherwise, the tighter bound is $c_d \geq 1 + (1+c)/\rho$. Substituting it in and confounding the constant additive terms into a constant K, we find that $(\delta^d\rho - \beta^{d-1})c \geq K$. By assumption, the multiplicative term is positive: there does indeed exist such a c.

6.5 Serialization

Lemma 5. *A set of safe triples is a safe set if and only if there exists a serial ordering of single insertions under which each triple is safe when it is inserted.*

Proof. The forward direction is fairly simple: Consider a set of safe triples, $\mathcal{S}$, for which there exists a serial ordering of insertions where each insertion is a safe insertion. When a triple $\langle p, r, f\rangle$ is inserted, we know it is safe. In particular, we know that for every previously-inserted safe triple $\langle q, r_q, f\rangle$, $|pq| \geq \delta r \geq \delta \min(r, r_q)$.

The reverse direction takes greater effort: Define the dimension of a triple $\langle p, r, f\rangle$ as the dimension of f. I claim that if we insert the highest-dimension triples first, breaking ties according to the radius of each triple, then under this ordering, every insertion is safe.

We need to prove that each $\langle p, r, f\rangle$ in the safe set $\mathcal{S}$ remains safe even after inserting earlier members of $\mathcal{S}$. Safety requires three conditions. First, p must not yield: there must not be a subfeature with a large empty ball. This was true when

$\mathcal{S}$ was chosen, so after adding a few more vertices, it must remain true. Second, we need to show that there does not exist a vertex v on f such that $|pv| < \delta r$. If v was present before any of the vertices of the safe set were inserted, then $\langle p, r, f\rangle$ cannot have been safe. Otherwise, v stems from another member $\langle v, r_v, f_v\rangle$ of the safe set. Since the set is safe, $|pv| \geq \delta \min(r, r_v)$. If f_v is of lower dimension than f, it would not have been inserted yet, according to the ordering; therefore $\dim(f) \leq \dim(f_v)$. But v is on both f and f_v, which proves that $f = f_v$. By the ordering, since v was inserted before p, $r_v \geq r$. Then we can conclude that $|pv| \geq \delta r$.

Third, we need to show that $\langle p, r, f\rangle$ still matches one of the rules even after inserting a few other triples. If, when it was chosen to join $\mathcal{S}$, the triple matched the INPUT, SPACING, or ENCROACHMENT rules, then it still does. If it matched the YIELDING rule, then the point that yields to p may have been an almost-safe triple that is now a vertex; it still matches the rule. Finally, in the QUALITY case, the vertex v that $B(p, r)$ contained still exists, and the distance to v's nearest neighbour can only have fallen, so the rule still matches.

7 Mesh size

Proving that an algorithm terminates is critical, but we will typically want more: that the mesh it outputs is small. In this section, I give an upper bound on the size of the mesh, in terms of an integral of the local feature size. I also discuss lower bounds on mesh size.

7.1 Upper bound

Theorem 2. *If $\mathcal{X}$ is a piecewise manifold complex with non-acute angles, and its domain Ω is bounded by a piecewise linear shape (for example, a box), then a safe meshing algorithm will output a mesh with no more than $O(\int_\Omega \mathrm{lfs}^{-d}(x)dx)$ vertices.*

Proof. A meshing algorithm would be safe if it inserted no points at all, but it would then be hard to analyze properties of the mesh. I avoid this technicality by specifying that after the algorithm has done its work, we will refine it further, until no point is safe. During this postprocess, we will set γ to 1 if it was otherwise specified to be larger. This process only adds vertices. Let M be the set of all vertices inserted either by the original algorithm or by the postprocess.

The proof is reversed from what is intuitive: we start with the integral and show that it is lower-bounded by a constant factor times the number of vertices. Consider the Voronoi diagram of M. I define $V(v)$ to be the Voronoi cell of a vertex v. The Voronoi cells, restricted to the domain Ω, tessellate Ω, so the integral over Ω is the sum of the integral over each restricted Voronoi cell:

$$\int_\Omega \mathrm{lfs}(x)^{-d}dx = \sum_{v\in M} \int_{V(v)\cap\Omega} \mathrm{lfs}(x)^{-d}dx$$

The ball $b \equiv B(v, \mathrm{NN}(v)/2)$ is entirely within the Voronoi cell of v, so we can lower-bound the term for v with $\int_{b\cap\Omega} \mathrm{lfs}(x)^{-d}dx$. For any such x, $\mathrm{lfs}(x) \leq \mathrm{lfs}(v) + |vx|$. We know from Theorem 1 that $\mathrm{lfs}(v) \leq (c+1)\,\mathrm{NN}(v)$, and from the radius of the ball that $|vx| \leq \mathrm{NN}(v)/2$; thus $\mathrm{lfs}(x) \leq (c+1.5)\,\mathrm{NN}(v)$. Therefore, the integral on the Voronoi cell of v reduces to

$$\int_{V(v)\cap\Omega} \mathrm{lfs}(x)^{-d}dx \geq \frac{\mathrm{Volume}(b\cap\Omega)}{((c+1.5)\,\mathrm{NN}(v))^d)}$$

The volume of b is exactly $V_d \times (\mathrm{NN}(v)/2)^d$, where V_d is the volume of the unit d-ball. However, the ball may protrude outside the domain. This does not happen if v is in the strict interior of Ω: if it did protrude, that would imply that v was very close to the boundary, but then the YIELDING rule would apply. On the boundary, the least fraction of b that is within the domain is realized when v is a corner. Due to the non-acute condition on the geometry, in an infinitesimal neighbourhood of the corner, Ω is a cone at least as large as a quadrant. Because we refined with $\gamma \leq 1$, and because the bounding facets are assumed to be linear, Ω is a cone throughout b. In other words, $\mathrm{Volume}(b\cap\Omega) \geq \mathrm{Volume}(b)/2^d = V_d\,\mathrm{NN}(v)^d/2^{2d}$. Substituting into the original integral and sum, the $\mathrm{NN}(v)^d$ terms cancel and we are left with:

$$\int_\Omega \mathrm{lfs}(x)^{-d}dx \geq \sum_{v\in M} \frac{V_d}{(c+1.5)^d\cdot 2^{2d}} = \frac{V_d}{(4c+6)^d}|M|$$

7.2 Lower bound

A two-dimensional proof of Ruppert [Rup95] that easily extends to any dimension states that any triangulation — in particular, the one with the fewest vertices — with bounded aspect ratio simplices that respect $\mathcal{X}$ (under some restrictions) must contain at least $\Omega(\int_\Omega \mathrm{lfs}_{\mathrm{trad}}(x)^{-d}dx)$ vertices. For the proof to hold, $\mathcal{X}$ must be a non-acute piecewise *linear* complex, and the domain must be convex. This is highly reminiscent of the upper bound we proved in the previous section; the differences are that the upper bound also applied to curved interior surfaces, and used the lfs function instead of $\mathrm{lfs}_{\mathrm{trad}}$. It happens that lfs and $\mathrm{lfs}_{\mathrm{trad}}$ are within a constant factor of each other, which proves that on any non-acute PLC, a safe meshing algorithm outputs a number of vertices within a constant factor of the smallest possible mesh of good aspect ratio. Keep in mind that a safe meshing algorithm only outputs a mesh with good radius/edge ratio, and must then be post-processed to achieve good aspect ratio; however, the post-process only adds a linear number of vertices. Unfortunately, little is known about lower bounds for meshes that respect curved input.

Lemma 6. *If $\mathcal{X}$ is piecewise linear and all its features meet at non-acute angles, and if* sf *is unspecified (or everywhere infinite), then* $\mathrm{lfs}_{\mathrm{trad}}(p) \leq \mathrm{lfs}(p) \leq 3\,\mathrm{lfs}_{\mathrm{trad}}(p)$.

Proof. For the upper bound, consider a point p in Ω. By definition, two disjoint features f and f' intersect the ball $B(p, \mathrm{lfs}_{\mathrm{trad}}(p))$. Let q be the closest point to p on f. The ball $B(q, 2\,\mathrm{lfs}_{\mathrm{trad}}(p))$ is centered on f and intersects f'. Therefore $(f\cup f')\cap$

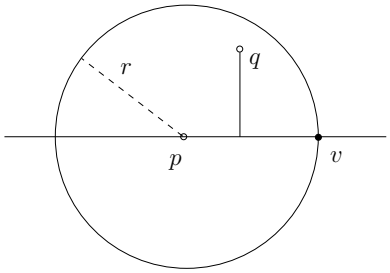


Fig. 5: PROJECTION: The projection of q onto the line vp lies to the right of p, as does v.

$B(q, 2\,\mathrm{lfs}_{\mathrm{trad}}(p))$ cannot be contractible, and $\mathrm{lfs}(q) \leq 2\,\mathrm{lfs}_{\mathrm{trad}}(p)$. Finally, since lfs is 1-Lipschitz, $\mathrm{lfs}(p) \leq \mathrm{lfs}(q) + |pq| \leq 3\,\mathrm{lfs}_{\mathrm{trad}}(p)$.

For the lower bound, there are two cases depending on whether $\mathrm{lfs}(p)$ is set by the contractibility condition or by the Lipschitz condition; the spacing function condition does not apply. First, let p be on a feature f, and let f' be a feature such that $(f \cup f') \cap B(p, \mathrm{lfs}(p))$ is not contractible. If f and f' met anywhere, they would meet at a non-acute angle; but then $B(p, \mathrm{lfs}(p))$ would intersect them in a topological ball. So f and f' are disjoint, and $\mathrm{lfs}_{\mathrm{trad}}(p) \leq \mathrm{lfs}(p)$. Second, if $\mathrm{lfs}(p)$ is set by the Lipschitz condition, there is a q with $\mathrm{lfs}(q)$ set by the contractibility condition such that $\mathrm{lfs}(p) = \mathrm{lfs}(q) + |pq|$. We know $\mathrm{lfs}(q) \geq \mathrm{lfs}_{\mathrm{trad}}(q)$ from the prior argument: $\mathrm{lfs}(p) \geq \mathrm{lfs}_{\mathrm{trad}}(q) + |pq|$. The $\mathrm{lfs}_{\mathrm{trad}}$ function is 1-Lipschitz, so we know that $\mathrm{lfs}_{\mathrm{trad}}(q) \geq \mathrm{lfs}_{\mathrm{trad}}(p) - |pq|$. A final substitution yields $\mathrm{lfs}(p) \geq \mathrm{lfs}_{\mathrm{trad}}(p)$ as claimed.

Corollary 1. *A safe meshing algorithm run on a non-acute piecewise linear complex with convex domain outputs only a constant factor more vertices than necessary.*

8 Improving the requirement on ρ

We have proved so far that most algorithms that insert only safe sets of points terminate as long as $\rho \geq 2^{d-1}$. Ruppert and Shewchuk proved something stronger: that their algorithms terminate if $\rho > 2^{(d-1)/2}$. To improve my bound, I propose a slightly stricter definition of safety, which will allow us to prove a smaller value for β. Recall from Lemma 3 that the value β is defined by the YIELDING rule, and is found by seeing how far a point q can be from a vertex v on the feature f to which it yields. Using the triangle inequality, $|qv| \leq 2r$. But under the following condition (see Figure 5), which is only slightly more strict than YIELDING, we can prove $|qv| \leq \sqrt{2}r$:

Definition 8 (PROJECTION). *Let $B(p, r)$ be a ball centered on an input feature $f \in \mathcal{X}$. Let v be a vertex on f, with $\delta r \leq |pv| \leq r$. Let q be a point on a feature f_q that is a superfeature of f, and q is either a vertex of the mesh, or is the center of*

an almost-safe triple. Then $\langle p, r, f\rangle$ satisfies the PROJECTION *rule if both v and the orthogonal projection of q onto the line pv lie on the same side of p.*

Lemma 7. *Consider an almost-safe triple $\langle p, r, f\rangle$. If it yields via the* PROJECTION *rule to an almost-safe triple $\langle q, r_q, f_q\rangle$, where f_q is a subfeature of f, then $r \leq \frac{\beta}{\delta} r_q$, with $\beta = \sqrt{2}$. If the input is a piecewise linear complex, and p does not yield, then no subfeature of f intersects the ball $B(p, \delta r/(1+\delta))$.*

Proof. Let $\langle p, r, f\rangle$ be almost-safe, due to the PROJECTION rule. There is a vertex v nearby (in $B(p, r)$), and a point q that yields to p. Consider the plane that goes through p, v, and q. Without loss of generality, we can rotate and translate this plane such that p is the origin, $v = (x_v, 0)$ with positive x_v, and $q = (x_q, y_q)$. We know that $x_q^2 + y_q^2 \leq r^2$, and that $\delta r \leq x_v \leq r$. Finally, the projection condition shows that $x_q \geq 0$. We can compute $|qv|$ directly to find $|qv|^2 \leq (x_q - x_v)^2 + y_q^2 \leq 2r^2 - 2x_v x_q$. Since both x_v and x_q are positive, we can drop the negative term to conclude $|qv| \leq \sqrt{2}r$. But $|qv| \geq \delta r_q$, or q would not be almost-safe.

The reverse condition follows by a proof of Shewchuk [She98]: in a piecewise linear complex, if p can yield under the YIELDING rule to some q, then it can yield under the PROJECTION rule to some q', not necessarily the same point as q. Contrapositively, if PROJECTION does not apply, then YIELDING does not apply. Therefore, we can use the proof of Lemma 3.

Therefore, an algorithm that uses the PROJECTION rule is guaranteed to terminate, with a small mesh, when $\rho \geq (1+\delta)\sqrt{2}^{d-1}/\delta^d$, which is a substantial improvement in the achievable quality ρ. Unfortunately, there is not currently a proof that we can indeed use PROJECTION on non-linear inputs.

9 Algorithms

So far, we have seen that the safety rules are sufficiently restrictive for us to guarantee an algorithm that follows the rules will terminate with a small mesh. The question remains, however: are the rules sufficiently permissive to implement interesting algorithms? The present section answers in the affirmative by showing that existing algorithms fit the framework.

9.1 Ruppert/Shewchuk refinement

In Ruppert/Shewchuk refinement [Rup95, She98], the input is assumed to be a piecewise linear complex in two (for Ruppert) or three (for Shewchuk) dimensions, with linear features meeting at non-acute angles. Given that Ruppert's algorithm is essentially a special case of Shewchuk's algorithm, here I will only describe the latter. The algorithm maintains a mesh (a Delaunay triangulation) for the domain Ω, but also one for each polygonal facet, and for each segment (though these are trivial). The initial step is to compute the Delaunay triangulations of the input points on

each feature. This corresponds to repeatedly invoking the INPUT rule, which is always safe. A *subsimplex* is a simplex of a segment or of a polygonal facet mesh. In the affine plane of their correponding feature, subsimplices have an associated circumball $B(p, r)$, which naturally extends to $\mathbb{R}^3$. Shewchuk defines an *encroached* subsimplex as being one whose associated ball contains either a vertex of a disjoint feature, or a point being considered for insertion. When a subfacet is encroached, we should insert its circumcenter p — unless the circumcenter itself encroaches a subsegment. This corresponds to the ENCROACHMENT and YIELDING rules. Finally, if a tetrahedron is skinny (that is, if it has circumradius r, shortest edge uv, and $r \geq \rho|uv|$), then we may attempt to insert its circumcenter, yielding as necessary. The nearest neighbour of u can be no further than $|uv|$ (it may be far closer), and the circumball of the tetrahedron has u on its surface, so the circumball of the tetrahedron is almost safe according to the QUALITY rule. Shewchuk proves a projection lemma: when a subsimplex is encroached, another subsimplex on the same feature is also encroached for which the PROJECTION rule applies. Upon yielding, Shewchuk proposes to split *all* encroached subsimplices. This is unsafe; however, the natural implementation is to push all the subsimplices onto a work queue and process them sequentially, not splitting subsimplices that no longer exist. A subsimplex no longer exists if another point is inserted in its circumball: the points actually inserted form a safe set. Shewchuk uses $\delta = 1$ in theory (in practice, numerical errors may reduce δ). Theorem 1 thus proves that the algorithm terminates for any $\rho > 2$, matching Shewchuk's bound; or, in two dimensions, for $\rho > \sqrt{2}$, matching Ruppert's bound.

9.2 Parallel Sparse Voronoi Refinement

Sparse Voronoi Refinement [HMP06, HMP07] (SVR) follows a formalism much like Ruppert/Shewchuk refinement. SVR differs in one main regard: not all the input points are in the mesh at the outset. Instead, when SVR considers inserting a circumcenter p from a simplex with circumradius r, it first checks whether there is an uninserted point in $B(p, kr)$ for a constant k given by the user. If so, the uninserted point is inserted; this is an application of the INPUT rule. If not, p is inserted, which shows that $\delta = k$. The key invariants in SVR are that every mesh (for each feature) always has mediocre or better quality, and subsimplices are unencroached. When no ball is almost-safe due to QUALITY or YIELDING, SVR finds a vertex v that contains an uninserted input point in its Voronoi cell. If v is a Steiner point, SVR performs an INPUT insertion. Otherwise, SVR inserts the Voronoi node farthest from v. Let p be that point. We know that $\mathrm{lfs}_{\mathrm{trad}}(v) \leq |pv|$ because v is an input point, and there is another input point in the Voronoi cell. Then by the Lipschitz condition, $\mathrm{lfs}_{\mathrm{trad}}(p) \leq 2|pv|$, and $\mathrm{lfs}(p) \leq 6|pv|$. In other words, this move is a SPACING move with $\gamma = 1/6$. Parallel SVR differs from SVR in two respects: first, it chooses a set of such operations, each of them safe as we've shown, with the requirement that for any two triples $\langle p, r, f\rangle$ and $\langle q, r_q, f_q\rangle$, where $f \subseteq f_q$, the two balls $B(p, r)$ and $B(q, r_q)$ do not intersect (in fact, it requires rather more). In other words, $|pq| \geq r + r_q$. This is much more strict than the minimum requirement for set safety. To achieve good parallelism, the algorithm also uses the ENCROACHMENT rule explicitly, inserting points

to ensure that protecting balls of disjoint features do not overlap. All this matches the requirements of Theorem 1: Parallel SVR terminates with a small mesh, for any $k^d\rho > 2^{d-1}$. When these algorithms were written, the authors were not aware of the PROJECTION rule and Shewchuk's proof that it applies; using it, parallel SVR would terminate for any $k^d\rho > \sqrt{2}^{d-1}$.

9.3 Off-centers and the CoreDisk

Üngör finds that by inserting the *off-center*, rather than the circumcenter, but otherwise following Ruppert's algorithm, the size of the output mesh falls by almost half in practice [Üng04]. Given a bad-quality triangle abc with shortest edge ab, the *off-center* is the point p found by walking along the perpendicular bisector of ab until either p is the circumcenter of abc, or the triangle abp is of barely good quality. The distance $|ap|$ is at least $\rho|ab|$, so the off-center qualifies as being almost safe due to the QUALITY rule. This shows that off-center refinement terminates in two dimensions when $\rho > \sqrt{2}$.

Reportedly, the off-center does not extend well to three dimensions. However, Jampani and Üngör [JU07] define the *CoreDisk* of an edge ab as being a circle defined by the points p such that $|ap| = |bp| = \rho|ab|$. They recommend sampling a certain set of points outside the CoreDisk, but still on the Voronoi facet normal to $|ab|$. Each point is almost-safe due to the QUALITY rule. In addition, points are chosen so that each pair p and q subtend an angle $\theta > 60°$ above the midpoint of ab; therefore, $|pq| \geq |pa|$. In other words, the points form a safe set. The authors described their algorithm in a periodic point set model, but we can imagine using their CoreDisk sample to handle bad-quality simplices within Shewchuk's algorithm, using the PROJECTION rule as necessary. Theorem 1 then proves that *CoreDisk* refinement in three dimensions terminates on an input non-acute piecewise linear complex when $\rho > 2$.

9.4 Piecewise Smooth Complexes

Rineau and Yvinec [RY07] recently demonstrated a method for meshing non-acute piecewise *smooth* complexes, where each feature is a smooth manifold. They use the YIELDING rule almost exactly as described here, they refine skinny tetrahedra identically as Shewchuk's algorithm does (which we showed was the QUALITY rule), and they use an ENCROACHMENT rule to recover an initial surface mesh. Assuming $\delta = 1$, their algorithm terminates when $\rho > 4$. Slightly deviating from the present framework, they allow different quality requirements ρ_3 for tetrahedra in space and ρ_2 for triangles on 2-manifolds; it is not too hard to see how to extend my framework to show termination assuming $\rho_2 > 2$ and $\rho_3 > 4$.

10 Conclusions

The major limitation of my framework is that it requires a non-acuteness condition on the geometry. There is increasing understanding of how one may sidestep the

July 31, 2008

issue [CP06, CDL07, PW04]. Informally, the techniques consist of protecting the acute angle with a bounded number of points, then using standard Delaunay refinement away from the acute angles. Away from the acute angle, the local feature size does not go to zero, so the present techniques apply. The major obstacles to incorporating protection techniques into my framework are the definition of local feature size and of the ENCROACHMENT rule: my local feature size goes to zero at a small angle, and the ENCROACHMENT rule in fact allows inserting infinitely many points. On the other hand, protection techniques also do not guarantee inserting an optimal number of points.

Algorithms that insert only safe sets are guaranteed to terminate if ρ is large enough compared to β and δ. To improve the bound (and allow smaller values of ρ) on piecewise smooth complexes, probably the easiest path would be to show that some analogue to the PROJECTION rule applies. Even on piecewise linear complexes, termination can be guaranteed for slightly better (smaller) values of ρ using the technique of diametral lenses [Che93, She02]. Under the purely incremental setting I described in Figure 1, lenses do not help; to get an improvement requires occasionally removing some Steiner vertices.

The work here showed a general proof framework for getting good radius-edge meshes on sufficiently nice geometries, and showed how to apply it to some extant algorithms. The promise of the framework is evident in that we can easily compose algorithms and prove they are still correct: for instance, the Jampani and Üngör CoreDisk idea for choosing Steiner points in space can be applied to the Rineau and Yvinec algorithm, producing an algorithm that *provably* terminates on non-acute PSCs so long as $\rho \geq 4$, and that is also likely to have very good output size behaviour in practice.

References

[BOG02] Charles Boivin and Carl Ollivier-Gooch. Guaranteed-quality triangular mesh generation for domains with curved boundaries. *Inter. J. Num. Meth. Eng.*, 55(10):1185–1213, 2002.

[CC06] Andrey N. Chernikov and Nikos P. Chrisochoides. Generalized delaunay mesh refinement: From scalar to parallel. In *15th International Meshing Roundtable*, pages 563–580, 2006.

[CC07] Andrey N. Chernikov and Nikos P. Chrisochoides. Three-dimensional semi-generalized point placement method for delaunay mesh refinement. In *16th International Meshing Roundtable*, pages 25–44, October 2007.

[CD03] Siu-Wing Cheng and Tamal K. Dey. Quality meshing with weighted delaunay refinement. *SIAM J. Comput.*, 33(1):69–93, 2003.

[CDL07] Siu-Wing Cheng, Tamal K. Dey, and Joshua A. Levine. A practical Delaunay meshing algorithm for a large class of domains. In *International Meshing Roundtable*, pages 477–494, 2007.

[Che89] L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report TR-89-983, Department of Computer Science, Cornell University, 1989.

[Che93] L. Paul Chew. Guaranteed-quality mesh generation for curved surfaces. In *Symposium on Computational Geometry*, pages 274–280, 1993.

[CP06] Siu-Wing Cheng and Sheung-Hung Poon. Three-dimensional delaunay mesh generation. *Discrete & Computational Geometry*, 36(3):419–456, 2006.

[HMP06] Benoît Hudson, Gary L. Miller, and Todd Phillips. Sparse Voronoi Refinement. In *15th International Meshing Roundtable*, pages 339–356, Birmingham, Alabama, 2006. Also available as Carnegie Mellon University Tech. Report CMU-CS-06-132.

[HMP07] Benoît Hudson, Gary L. Miller, and Todd Phillips. Sparse parallel Delaunay mesh refinement. In *19th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 339–347, 2007.

[HPÜ05] Sariel Har-Peled and Alper Üngör. A time-optimal Delaunay refinement algorithm in two dimensions. In *21st Symposium on Computational Geometry*, pages 228–236, 2005.

[JU07] Ravi Jampani and Alper Üngör. Construction of sparse well-spaced point sets for quality tetrahedralizations. In *16th International Meshing Roundtable*, pages 63–80, 2007.

[LT01] Xiang-Yang Li and Shang-Hua Teng. Generating well-shaped Delaunay meshes in 3D. In *SODA*, pages 28–37, 2001.

[ORY05] Steve Oudot, Laurent Rineau, and Mariette Yvinec. Meshing volumes bounded by smooth surfaces. In *14th International Meshing Roundtable*, pages 203–219, 2005.

[PW04] Steven E. Pav and Noel Walkington. Robust three dimensional delaunay refinement. In *13th International Meshing Roundtable*, pages 145–156, 2004.

[Rup95] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995. Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (Austin, TX, 1993).

[RY07] Laurent Rineau and Mariette Yvinec. Meshing 3D domains bounded by piecewise smooth surfaces. In *16th International Meshing Roundtable*, pages 442–460, 2007.

[She96] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, Berlin, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[She98] Jonathan Richard Shewchuk. Tetrahedral Mesh Generation by Delaunay Refinement. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, pages 86–95, Minneapolis, Minnesota, June 1998. Association for Computing Machinery.

[She02] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, 2002.

[STÜ04] Daniel Spielman, Shang-Hua Teng, and Alper Üngör. Parallel Delaunay refinement with off-centers. In *EUROPAR*, 2004.

[STÜ07] Daniel Spielman, Shang-Hua Teng, and Alper Üngör. Parallel Delaunay refinement: Algorithms and analyses. *IJCGA*, 17:1–30, 2007.

[Üng04] Alper Üngör. Off-centers: A new type of Steiner point for computing size-optimal quality-guaranteed Delaunay triangulations. In *LATIN*, pages 152–161, 2004.