Sliver Removal by Lattice Refinement

François Labelle
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, California 94720
flab@cs.berkeley.edu

ABSTRACT

I present an algorithm that can provably eliminate slivers in the interior of a tetrahedral mesh, leaving only tetrahedra with dihedral angles between 30 and 135 degrees and radius-edge ratios of at most 1.368, except near the boundary. In comparison, previous bounds on dihedral angles were microscopic. The final mesh can respect specified input vertices and a user-defined sizing function. The algorithm comes with a bound on the sizes of the features it creates, and can provably grade from small to large tetrahedra.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

Tetrahedral mesh generation, mesh quality, sliver tetrahedron, dihedral angle

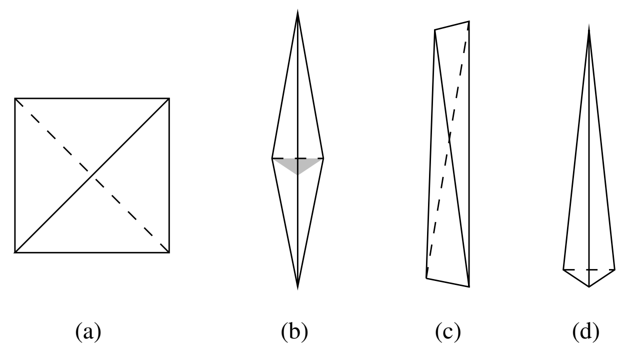

Figure 1: (a) Sliver tetrahedron: a good radius-edge ratio does not rule out poor dihedral angles. (b) Spear tetrahedron: a lower bound ($60°$) on dihedral angles does not rule out a very large dihedral angle. (c) Splinter tetrahedron: an upper bound ($90°$) on dihedral angles does not rule out very small dihedral angles. (d) Needle tetrahedron: bounds on both small ($60°$) and large ($90°$) dihedral angles do not rule out an arbitrary large radius-edge ratio.

1. INTRODUCTION

Decomposing a domain into simple elements like tetrahedra is often the first step toward the numerical simulation of a physical phenomenon. For this purpose, the elements should be well-shaped for reasons of accuracy and stability [5, 9, 20]. In most applications, tetrahedra that are close to regular are favored, while tetrahedra that are close to degenerate should be avoided. Many measures of tetrahedron quality have been proposed to capture this concept quantitatively. Most of these measures are equivalent in the sense that a bound on one implies a bound on the others [12]. Since a single bad element can potentially ruin a whole simulation, it is desirable to have a guarantee on the quality of the worst element of the mesh.

For domains with no acute angles, three-dimensional Delaunay refinement [19] comes with a guarantee (upper bound) on the ratio of a tetrahedron's circumradius to the length of its shortest edge, or *radius-edge ratio* for short. Unfortunately, this guarantee does not rule out the *sliver tetrahedron*, which can have dihedral angles arbitrarily close to $0°$ and $180°$; see Figure 1(a). In this context, the problem of obtaining well-shaped tetrahedra is reduced to eliminating slivers, and an algorithm with a guarantee on radius-edge ratios should be complemented with a guarantee on the smallest dihedral angle. See Figure 1(b,c,d) for examples of tetrahedra that demonstrate that bounds on dihedral angles alone may not be sufficient to rule out some types of degenerate tetrahedra.

1.1 Previous Work

Since 3D Delaunay refinement apparently comes so close to providing well-shaped elements, it is natural to ask whether slivers can be eliminated in a post-processing step. Chew [4] eliminates slivers by inserting a vertex randomly in a ball around the circumcenter of sliver tetrahedra. Cheng

Supported in part by the National Science Foundation under Awards CCR-0204377 and CCF-0430065, and in part by an Alfred P. Sloan Research Fellowship.

SCG'06, June 5–7, 2006, Sedona, Arizona, USA.

et al. [3] show that if the radius-edge ratios of the tetrahedra are bounded, then slivers can be eliminated without inserting any extra vertices: one can switch to a *weighted* Delaunay tetrahedralization and select the weights of the vertices in such a way that all slivers disappear. Edelsbrunner et al. [7] show that smoothing can be used instead of weights. For these last two results, the authors assumed a periodic space to avoid having to deal with the domain boundary. This is a reasonable simplification in order to make progress on a very hard problem. Nevertheless, handling of the boundary is important for applications, and has been accomplished by Li and Teng [11] by improving upon the algorithm sketched by Chew [4], and by Cheng and Dey [2] who extended the work of Cheng et al. [3] instead. In all cases, the actual dihedral angle bounds, while positive, are too minuscule to be worth computing explicitly: they are probably less than 10^{-6} degrees. However, experiments by Edelsbrunner and Guoy [6] show that the algorithm of Cheng et al. [3] can eliminate almost all slivers with dihedral angles below 5° in practice.

Another option is to design an algorithm that makes use of an underlying regular grid. Yerry and Shephard [22] pioneered a tetrahedral mesh generation technique based on an octree. Field [8] proposes filling the interior of a domain with a tetrahedral mesh constructed from a complicated icosahedral assembly. Naylor [16] argues that the Delaunay tetrahedralization of the body-centered cubic lattice is a better choice, both for quality and simplicity reasons. Field and Smith [10] propose a method to obtain *graded* meshes based on the body-centered cubic lattice; however their algorithm description is informal and bounds on dihedral angles are not provided. Molino et al. [15] also construct graded meshes using the body-centered cubic lattice and obtain good angles in practice for domains with smooth boundaries. Mitchell and Vavasis [14] present an octree-based algorithm to create meshes with a very small guarantee on aspect ratios.

1.2 Summary of Results

In this paper, I present a Delaunay refinement algorithm which also has a structured grid flavor. I show how slivers can be eliminated by inserting new vertices carefully chosen from a discrete set called a *lattice*. The minimum dihedral angle guarantee of 30° is spectacularly good—even a 1° guarantee would have been significant progress.

Inserting new vertices to get rid of slivers is a powerful tool which can be abused: a first solution is to lay down a very fine regular grid, and to slightly perturb it to match the input vertices. A second solution is to find a way to wrap each input vertex individually with a small shell of lattice vertices, and to fill the outside of the shells in some regular way. See Figure 2. The problem with these two solutions is that they require a lot of new vertices. The algorithm that I propose has the second solution as its worst case, but is likely to be much better in practice for two reasons.

1. A new vertex is inserted only in response to a bad-quality tetrahedron. If the Delaunay tetrahedralization of the input is already a good quality mesh, then it won't be changed by the algorithm. If the input has only a few bad tetrahedra, it is likely that a few vertex insertions will suffice to eliminate them because good angles can also be obtained by chance, especially if the quality requirements are relaxed to, say, a 15° minimum dihedral angle.

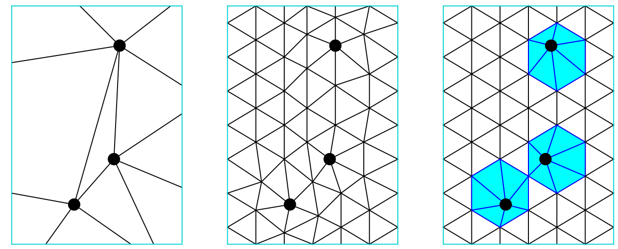

Figure 2: 2D equivalents of two possible strategies to get rid of slivers by refinement. (a) The Delaunay triangulation of the input contains a poorly shaped element. (b) Use a distorted fine grid. (c) Create small shells around input vertices and use a regular mesh outside.

2. Although one can start with standard Delaunay refinement to bound the radius-edge ratio and then use my algorithm to eliminate the remaining slivers, it is a much better idea to use my algorithm to do both at the same time. In lattice refinement, vertices with arbitrary coordinates are the enemy; they should not be needlessly generated.

In Section 4, I present a way to incrementally generate a good quality mesh made out of lattice vertices only. The generated tetrahedra can grade from small to large and are guaranteed to have dihedral angles in the interval $[30^\circ, 135^\circ]$ and radius-edge ratios at most 1.119. Using this technique as part of a Delaunay refinement algorithm guarantees good quality tetrahedra away from the boundary (and internal features of the domain), where the mesh is locally composed of lattice vertices only.

In Section 5, I add the ability to handle the simplest of internal features: input vertices with arbitrary coordinates. This makes my result directly comparable to previous results [4, 3]. This also slightly broadens the possible applications to, for example, the simulation of a number of point-like heat sources in 3D, where each source should be a vertex of the mesh. The dihedral angle guarantee stays the same, $[30^\circ, 135^\circ]$, and the bound on radius-edge ratio slightly worsens to 1.368. The algorithm can still grade from small to large tetrahedra, although the constant in the grading guarantee is weaker.

2. SIMPLIFIED MESH GENERATION

Ideally one would like to be able to mesh any *piecewise linear complex* (PLC) [13] with high-quality tetrahedra, even when the boundary is complicated. This paper doesn't handle the boundary. Provably eliminating slivers is a very hard problem that has seen little progress, so even a partial solution is significant.

Here are a few concrete ways to interpret boundary-less mesh generation, the first two being mathematically precise.

Periodic space [3, 7]. Run the algorithm on the periodic space $[0, 1)^3$, where space wraps around at the boundary like in the game *Asteroids*. Assuming that the space is initialized with enough vertices that inserting a new vertex won't create a tetrahedron that uses that new vertex twice, the periodic space behaves locally like $\mathbf{R}^3$ and the algorithm can run on it.

Superset mesh. Given a domain $\Omega \subset \mathbf{R}^3$ with boundary, one can ask for a good quality mesh that does not necessarily respect the boundary but is a superset of Ω, sometimes called a *simulation envelope* [18]. This can be accomplished by considering only the quality of tetrahedra that intersect Ω and allowing the insertion of new vertices outside of Ω. When the algorithm terminates, delete the tetrahedra that do not intersect Ω. The result is a superset mesh of Ω with good quality tetrahedra only.

Good quality except at the boundary. The algorithm is used as part of an existing solution to mesh PLCs. If the lattice refinement algorithm is about to insert a vertex that is too close to the boundary or even outside the domain, abort the insertion and do what the PLC algorithm would do instead.

In this paper the results and proofs are written with the notation of the superset mesh problem.

2.1 Definitions

Definition 1 (Sizing Function). *The* sizing function *is a scalar field $s : \Omega \to (0, \infty]$ used to prescribe the approximate size of the mesh at each point of the domain.*

I will use the sizing function as follows: the final mesh should be such that a closed ball of radius $s(p)$ centered at p is non-empty, for all $p \in \Omega$. In the case of a Delaunay mesh, this directly implies an upper bound of $s(p)$ for the circumradius of a tetrahedron, where p is the circumcenter.

Since the simplified setting does not allow input edges or faces, I use a definition of local feature size adapted to the case of an input vertex set with a sizing function. (See Ruppert [17] for the original definition.)

Definition 2 (Local Feature Size). *Let p be any point in $\mathbf{R}^3$. The local feature size due to input vertices, $\mathrm{lfs}_i(p)$, is the distance between p and its second-closest input vertex. The local feature size due to the sizing function is $\mathrm{lfs}_s(p) = \inf\{cs(a) + \mathrm{dist}(p, a) : a \in \Omega\}$, where c is some positive constant that will depend on the algorithm. The combined local feature size is $\mathrm{lfs}(p) = \min(\mathrm{lfs}_i(p), \mathrm{lfs}_s(p))$.*

By considering the case $a = p$ in the definition of $\mathrm{lfs}_s(p)$ we see that $\mathrm{lfs}_s(p) \le cs(p)$ for $p \in \Omega$.

Lemma 1. *For any two points p and q,*
(a) $\mathrm{lfs}_i(q) \le \mathrm{lfs}_i(p) + \mathrm{dist}(p, q)$,
(b) $\mathrm{lfs}_s(q) \le \mathrm{lfs}_s(p) + \mathrm{dist}(p, q)$,
(c) $\mathrm{lfs}(q) \le \mathrm{lfs}(p) + \mathrm{dist}(p, q)$;
i.e. the functions are 1-Lipschitz.

Proof. The proof is adapted from Ruppert [17].

(a) By the definition of $\mathrm{lfs}_i(p)$, there are two input vertices v and w at distance at most $\mathrm{lfs}_i(p)$ from p. By the triangle inequality, v and w are at distance at most $\mathrm{lfs}_i(p) + \mathrm{dist}(p, q)$ from q. Therefore $\mathrm{lfs}_i(q) \le \mathrm{lfs}_i(p) + \mathrm{dist}(p, q)$.

(b)

$$\begin{aligned}
\mathrm{lfs}_s(q) &= \inf\{cs(a) + \mathrm{dist}(q, a) : a \in \Omega\} \\
&\le \inf\{cs(a) + \mathrm{dist}(q, p) + \mathrm{dist}(p, a) : a \in \Omega\} \\
&= \inf\{cs(a) + \mathrm{dist}(p, a) : a \in \Omega\} + \mathrm{dist}(p, q) \\
&= \mathrm{lfs}_s(p) + \mathrm{dist}(p, q).
\end{aligned}$$

(c) Follows by taking the minimum of both sides of inequalities (a) and (b). □

3. SOME LATTICES

For conciseness I define addition and scalar multiplication of point sets as follows: $A + B = \{a + b : a \in A \text{ and } b \in B\}$ and $cA = \{ca : a \in A\}$.

Definition 3 (Simple Cubic Lattice). *The simple cubic lattice SC_0 and its scaling SC_k by powers of two are defined as*

$$\begin{aligned}
&\mathrm{SC}_0 = \mathbf{Z}^3, \\
&\mathrm{SC}_k = 2^k \mathrm{SC}_0 \text{ for } k \in \mathbf{Z}.
\end{aligned}$$

Definition 4 (Body-Centered Cubic Lattice). *The body-centered cubic lattice BCC_0 and its scaling BCC_k by powers of two are defined as*

$$\begin{aligned}
&\mathrm{BCC}_0 = \{(0,0,0), (\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2})\} + \mathbf{Z}^3, \\
&\mathrm{BCC}_k = 2^k \mathrm{BCC}_0 \text{ for } k \in \mathbf{Z}.
\end{aligned}$$

The following lattice is included for comparison only. It is known to describe a sphere packing of maximum density, but is not used in this paper.

Definition 5 (Face-Centered Cubic Lattice). *The face-centered cubic lattice FCC_0 and its scaling FCC_k by powers of two are defined as*

$$\begin{aligned}
&\mathrm{FCC}_0 = \{(0,0,0), (\tfrac{1}{2}, \tfrac{1}{2}, 0), (\tfrac{1}{2}, 0, \tfrac{1}{2}), (0, \tfrac{1}{2}, \tfrac{1}{2})\} + \mathbf{Z}^3, \\
&\mathrm{FCC}_k = 2^k \mathrm{FCC}_0 \text{ for } k \in \mathbf{Z}.
\end{aligned}$$

The sets above are called *point lattices* because they are discrete subgroups of Euclidean space under vector addition of point coordinates. Other regular point patterns, such as the centers of spheres in a hexagonal close packing, do not share this property. None of my results depend on this subgroup property, so non-lattice point sets can be used in future work if the lattices that I have chosen turn out to have limitations.

Proposition 2 (Nesting of Lattices).

$$\mathrm{BCC}_{k+1} \subset \mathrm{SC}_k \subset \mathrm{BCC}_k \text{ for } k \in \mathbf{Z}.$$

A lattice L_1 is said to be *finer* than a lattice L_2 if $L_1 \supset L_2$. L_1 is said to be *coarser* than L_2 if $L_1 \subset L_2$.

Given any full-rank lattice $L \subset \mathbf{R}^3$, let $e(L)$ be the minimum distance between two distinct points in L. Let $r(L)$ be the radius of the largest possible empty open ball in $\mathbf{R}^3 \backslash L$.

Lemma 3. *Any ball of radius $r(L)$ contains at least one point of L in its interior, or at least 4 on its boundary.*

Proof. Let B be an open ball of radius $r(L)$ with no point of L in its interior. By definition of $r(L)$, B must be a largest possible empty open ball. Since B is of maximum radius, there must be at least 4 points of L on its boundary to prevent the radius from being increased further. The result follows. □

Proposition 4.

$$\begin{aligned}
e(\mathrm{SC}_k) &= 2^k, & e(\mathrm{BCC}_k) &= 2^k\sqrt{3}/2, \\
r(\mathrm{SC}_k) &= 2^k\sqrt{3}/2, & r(\mathrm{BCC}_k) &= 2^k\sqrt{5}/4.
\end{aligned}$$

PROOF. The first two are clear. For the last two, note that maximal open balls occur at Voronoi vertices of the lattices. For example, $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ is a Voronoi vertex of SC_0 with a maximal empty ball of radius $\sqrt{3}/2$, and $(\frac{1}{2}, \frac{1}{4}, 0)$ is a Voronoi vertex of BCC_0 with a maximal empty ball of radius $\sqrt{5}/4$. Due to the symmetry of these lattices, all other maximal balls are of equal sizes. □

The Delaunay tetrahedralization of the body-centered cubic lattice gives a mesh with dihedral angles of 60° and 90°, and radius-edge ratios of $r(\mathrm{BCC}_k)/e(\mathrm{BCC}_k) = \sqrt{15}/6$ (≈ 0.645). This is an excellent choice to fill a volume with a *uniform* tetrahedral mesh [16]. Section 4 shows how lattices of different sizes can be used in concert to generate tetrahedra that can grade from small to large.

4. SIMPLE GRADED MESHING

An algorithm to generate graded meshes of guaranteed quality that honor a sizing function s (but not input vertices) can be devised easily: one can construct a balanced octree and take its Delaunay tetrahedralization. By analyzing the 2^6 possible types of cells in a balanced octree, we find that the dihedral angles are in the interval $[19.471°, 135°]$ and radius-edge ratios at most 1.347. These bounds can potentially be improved using tiles, as was done in 2D by Bern et al. [1].

The technique presented in this section is simple and generates tetrahedra with dihedral angles in the interval $[30°, 135°]$ and radius-edge ratios of at most $\sqrt{5}/2$ (< 1.119). It is formulated as an incremental insertion method to make it easy to combine with standard Delaunay refinement.

The method depends crucially on the following two lemmas, which assert that a tetrahedron with vertices in a lattice and a small circumradius either has good dihedral angles, or its circumsphere contains a lattice point (that the algorithm can insert to eliminate the tetrahedron).

LEMMA 5. *Let t be a tetrahedron with vertices in SC_k and a circumradius of $r(\mathrm{BCC}_{k+1})$ or less. Then the dihedral angles of t are all at least 30° and at most 135°.*

PROOF. By rescaling, it suffices to show that a tetrahedron with vertices in SC_0, dihedral angle of less than 30° or of more than 135°, and circumradius of $\sqrt{5}/2$ or less doesn't exist. Because of the bound on circumradius, there is a finite number of possible tetrahedra to test. The verification was carried out by a computer. Figure 3 shows tetrahedra that achieve the extreme angles. □

LEMMA 6. *Let t be a tetrahedron with vertices in BCC_k and a circumradius of $r(\mathrm{SC}_k)$ or less. If t has a dihedral angle less than $\arccos(\sqrt{6}/3)$ ($\approx 35.264°$) or larger than $\arccos(-\sqrt{3}/3)$ ($\approx 125.264°$), then there exists a point of SC_k inside the circumsphere of t.*

PROOF. By rescaling, it suffices to show that a tetrahedron with vertices in BCC_0, dihedral angle of less than $\arccos(\sqrt{6}/3)$ or of more than $\arccos(-\sqrt{3}/3)$, and circumradius of $\sqrt{3}/2$ or less must always contain a point of SC_0 inside its circumsphere. Because of the bound on circumradius, there is a finite number of possible tetrahedra to test. The verification was carried out by a computer. Figure 4 shows a tetrahedron that achieves the extreme angles. □

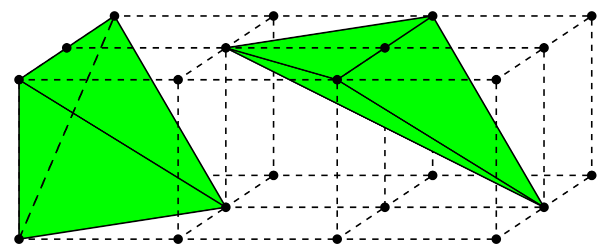

Figure 3: Two worst case tetrahedra for Lemma 5. Each tetrahedron has vertices in SC_0 and a circumradius of $\sqrt{5}/2$ or less. They are the only such tetrahedra to achieve at least one of the angle bounds of Lemma 5. The first has dihedral angles in $[30°, 120°]$ and the second has dihedral angles in $[30°, 135°]$.

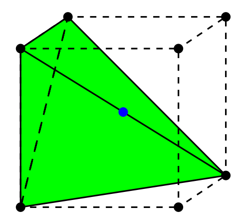

Figure 4: A worst case tetrahedron for Lemma 6. It has vertices in BCC_0, a circumradius of $\sqrt{3}/2$ or less, and no point of SC_0 inside its circumsphere. It is the only such tetrahedron to achieve at least one of the angle bounds of Lemma 6 (it achieves both).

4.1 Algorithm

The algorithm generates a mesh by incrementally inserting lattice vertices while maintaining a Delaunay tetrahedralization of these lattice vertices. When a vertex is created it is assigned a *label*, either SC_k or BCC_k for some k, and is tagged by a *type number*. The labels are needed in the implementation, but the type numbers are not (they are only used in the analysis). Each vertex belongs to the lattice of its label, but the label is not necessarily the coarsest lattice that contains the vertex.

The algorithm inserts vertices for one of two reasons: the sizing function, or a bad-quality tetrahedron. In the latter case, the inserted vertex always comes from a lattice that is strictly coarser than the finest lattice label of the tetrahedron vertices. This allows a proof of good grading. The refinement strategy is closer to Üngör's off-centers [21] than to the circumcenter method.

The complete algorithm appears in Figure 5. Because the algorithm is building a superset mesh, any empty half-space that is tangent to a vertex, an edge, or a triangle of the tetrahedralization can be considered a degenerate circumsphere with missing vertices at infinity. Vertices at infinity are considered infinitely coarse in the nesting of lattices. Obviously, the tangent vertices can only be on the convex hull of the tetrahedralization. If such a half-space intersects Ω, then the corresponding fictive tetrahedron qualifies for quality enforcement.

Input: A domain $\Omega \subset \mathbf{R}^3$ and a sizing function $s : \Omega \to (0, \infty]$.

Procedure: Repeat the enforcement steps below in any order until none apply. Maintain a Delaunay tetrahedralization as new vertices are inserted. When finished, remove all tetrahedra whose interiors do not intersect Ω.

Size enforcement: If there exists a point $p \in \Omega$ such that a closed ball B of radius $s(p)$ centered at p is empty:

Let j be the largest integer such that $r(\mathrm{SC}_j) \le s(p)$. By Lemma 3 there exists a point w of SC_j in B. **Insert w with label** SC_j into the Delaunay tetrahedralization.

Quality enforcement: If the Delaunay tetrahedralization contains a tetrahedron t whose interior intersects Ω and which has a dihedral angle smaller than 30° or larger than 135°, or a radius-edge ratio larger than $\beta = \sqrt{5}/2$, do:

Among the 4 vertices of t, find the vertex v with the finest lattice label, according to the nesting of lattices.

Case 1: The label of v is SC_k.

By assumption, the circumradius of t is larger than $e(\mathrm{SC}_k)\beta = r(\mathrm{BCC}_{k+1})$, or t has a dihedral angle that is less than 30° or greater than 135°. By Lemma 5 the circumradius of t is guaranteed to be larger than $r(\mathrm{BCC}_{k+1})$. Let B be a closed ball of radius $r(\mathrm{BCC}_{k+1})$ tangent at v inside the circumsphere of t. By Lemma 3 there exists a point w of BCC_{k+1} in $B \backslash \{v\}$. **Insert w with label BCC_{k+1} and type 1.**

Case 2: The label of v is BCC_k.

By assumption, the circumradius of t is larger than $e(\mathrm{BCC}_k)\beta > r(\mathrm{SC}_k)$, or t has a dihedral angle that is less than 30° or greater than 135°. If the circumradius of t is larger than $r(\mathrm{SC}_k)$, then let B be a closed ball of radius $r(\mathrm{SC}_k)$ tangent at v inside the circumsphere of t. By Lemma 3 there exists a point w of SC_k in $B \backslash \{v\}$. Else by Lemma 6 there exists a point w of SC_k inside the circumsphere of t. **Insert w with label SC_k and type 2.**

Figure 5: A simple algorithm to create a superset mesh of a domain Ω with quality tetrahedra. The final mesh respects a user-defined sizing function. The vertex types are used in the correctness proof.

As written, the algorithm starts with no vertices at all, so it must first perform a size enforcement step. When there is at least one vertex, the empty half-space rule can apply and lead to quality enforcement steps.

4.2 Analysis

A mesh generation algorithm has *good grading* if the sizes of the elements can vary from small to large over a short distance. Since the work of Ruppert [17], a grading guarantee is usually a proof of a linear relationship between the nearest neighbor distance of a vertex v of the final mesh and its local feature size $\mathrm{lfs}(v)$.

As a first step we show that there exist positive constants a and b such that the following holds.

$$\mathrm{lfs}(v) \le \begin{cases} 2^k a & \text{if } v \text{ has label } \mathrm{SC}_k, \\ 2^k b & \text{if } v \text{ has label } \mathrm{BCC}_k. \end{cases} \tag{1}$$

We show by induction that these bounds are maintained by the algorithm. The constants a and b (and c of Definition 2) are derived at the end of this section. Once we have these bounds, the following theorem shows that we obtain good grading.

THEOREM 7. *If a mesh consists of only* SC *or* BCC *lattice vertices, and if there exist positive constants a and b such that (1) holds, then for any vertex v of the mesh, the distance to its nearest neighbor is at least*

$$\min\left(\tfrac{1}{1+a}, \tfrac{1}{1+2b/\sqrt{3}}\right)\mathrm{lfs}(v).$$

PROOF. (Adapted from Ruppert [17]). Let v be any vertex of the mesh. Let w be its nearest neighbor.

If the label of v is as fine or finer than the label of w then:

In case v has label SC_k: $\mathrm{dist}(v,w) \ge e(\mathrm{SC}_k) = 2^k$. $\mathrm{lfs}(v) \le 2^k a$. So $\mathrm{dist}(v,w) \ge \frac{1}{a}\mathrm{lfs}(v)$.

In case v has label BCC_k: $\mathrm{dist}(v,w) \ge e(\mathrm{BCC}_k) = 2^k\sqrt{3}/2$. $\mathrm{lfs}(v) \le 2^k b$. So $\mathrm{dist}(v,w) \ge \frac{\sqrt{3}}{2b}\mathrm{lfs}(v)$.

In either case, $\mathrm{dist}(v,w) \ge \min(\frac{1}{a}, \frac{\sqrt{3}}{2b})\mathrm{lfs}(v)$.

Else (w is finer than v): We use Lemma 1 and apply the bound above to w.

$$\begin{aligned} \mathrm{lfs}(v) &\le \mathrm{lfs}(w) + \mathrm{dist}(v,w) \\ &\le \mathrm{dist}(v,w)/\min(\tfrac{1}{a}, \tfrac{\sqrt{3}}{2b}) + \mathrm{dist}(v,w) \\ &= \max(1+a, 1+\tfrac{2b}{\sqrt{3}})\mathrm{dist}(v,w). \end{aligned}$$

So $\mathrm{dist}(v,w) \ge \min(\frac{1}{1+a}, \frac{1}{1+2b/\sqrt{3}})\mathrm{lfs}(v)$. $\square$

We perform a separate analysis for each type of inserted vertex w in the algorithm of Figure 5.

4.2.1 *Insertion due to size*

The new vertex w is of label SC_j and satisfies $\mathrm{lfs}(w) \le cs(w) < 2cr(\mathrm{SC}_j) = 2^j\sqrt{3}c$. We require that $\sqrt{3}c \le a$, so that $\mathrm{lfs}(w) \le 2^j a$ and the insertion preserves (1).

4.2.2 *Type 1 insertion*

The new vertex w has label BCC_{k+1} at a distance at most $2r(\mathrm{BCC}_{k+1}) = 2^k\sqrt{5}$ from v with label SC_k. So $\mathrm{lfs}(w) \le \mathrm{lfs}(v) + \mathrm{dist}(v,w) \le 2^k a + 2^k\sqrt{5}$. We require that $a + \sqrt{5} \le 2b$, so that by induction $\mathrm{lfs}(w) \le 2^{k+1}b$.

4.2.3 *Type 2 insertion*

The new vertex w has label SC_k at a distance at most $2r(\mathrm{SC}_k) = 2^k\sqrt{3}$ from v with label BCC_k. So $\mathrm{lfs}(w) \le \mathrm{lfs}(v) + \mathrm{dist}(v,w) \le 2^k b + 2^k\sqrt{3}$. We require that $b + \sqrt{3} \le a$, so that by induction $\mathrm{lfs}(w) \le 2^k a$.

4.2.4 Guarantee

The requirements can be satisfied by setting $a = 2\sqrt{3} + \sqrt{5}$, $b = \sqrt{3}+\sqrt{5}$, and $c = a/\sqrt{3}$. By applying Theorem 7 we deduce that during the course of the algorithm, the distance between any vertex v and its nearest neighbor is at least $\mathrm{lfs}(v)/6.701$.

Assuming there is a positive lower bound on the sizing function s, this result gives a positive lower bound on the distance between any pair of vertices. This in turn implies termination of the algorithm, because Ω has finite volume. (A slightly bigger volume must be considered in the case of a superset mesh).

5. ADDING VERTEX CONSTRAINTS

In this section we extend lattice refinement to allow input vertices with arbitrary coordinates in Ω. The algorithm inserts lattice vertices, but never "too close" to an input vertex. Each lattice point has an associated *forbidden region* around it. No lattice point is ever inserted that has an input vertex in its forbidden region.

Definition 6 (Forbidden Region). *Let $p \in \mathbf{R}^3$ and $k \in \mathbf{Z}$. The forbidden region $R(p,k)$ is a cube with side $2^k(2+\sqrt{6})/2$ centered at p. I.e.*

$$R(p,k) = \{q : \|q-p\|_\infty \leq 2^k(2+\sqrt{6})/4\}.$$

Also define

$$\rho = \sqrt{3}(2+\sqrt{6})/4$$

to be the safety radius constant, which is the distance from p to the furthest point of $R(p,0)$.

The lattice points that the algorithm inserts are called *refinement vertices* to distinguish them from input vertices. The algorithm maintains the invariant that if a refinement vertex v has label SC_k or BCC_{k+1}, then $R(v,k)$ contains no input vertex.

Theorem 8. *For any $p \in \mathbf{R}^3$ and any $k \in \mathbf{Z}$, let $S = \{p\} \cup (\mathrm{SC}_k \backslash R(p,k))$. Then every Delaunay tetrahedron in S has dihedral angles in the interval $[30°, 127.903°]$, and a radius-edge ratio of at most 1.368.*

Proof. By rescaling, it suffices to consider the case $k = 0$. Because the forbidden region is a cube with half-side $\sigma = (2+\sqrt{6})/4$ (≈ 1.112), if the coordinate of p along some axis falls in the set $(\sigma, 3-\sigma)+\mathbf{Z}$, then two points of SC_0 will be covered by the forbidden region $R(p,0)$ along that axis. If the coordinate falls in the complement $[3-\sigma, 1+\sigma]+\mathbf{Z}$, then three points of SC_0 will be covered along that axis. The total number of points of SC_0 that are removed by the set difference with $R(p,0)$ is therefore 8, 12, 18 or 27, depending on the three coordinates of p.

By invoking translational and mirror symmetries, we can assume that the coordinate of p along some axis is in the interval $I = (\sigma, \frac{3}{2}]$ or $J = [3-\sigma, 2]$. The possible cases for p are then reduced to these four:

1. $p \in I^3$ and $S = \{p\} \cup (\mathrm{SC}_0 \backslash \{1,2\}^3)$
2. $p \in I^2 \times J$ and $S = \{p\} \cup (\mathrm{SC}_0 \backslash \{1,2\}^2 \times \{1,2,3\})$
3. $p \in I \times J^2$ and $S = \{p\} \cup (\mathrm{SC}_0 \backslash \{1,2\} \times \{1,2,3\}^2)$
4. $p \in J^3$ and $S = \{p\} \cup (\mathrm{SC}_0 \backslash \{1,2,3\}^3)$

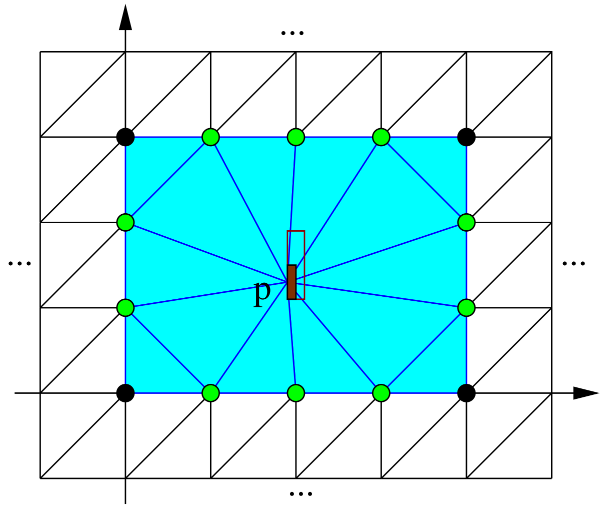

Figure 6: A 2D equivalent of a Delaunay tetrahedralization of the set $S = \{p\} \cup (\mathrm{SC}_0 \backslash R(p,0))$ in the proof of Theorem 8. In this example, p lies in the rectangular frame at the center, so the hole in SC_0 created by the set difference with $R(p,0)$ is a 4×3 rectangle. By symmetry, we can assume that p lies in the smaller sub-rectangle. Hollow circles are the box face vertices.

The hole in SC_0 created by the set difference with $R(p,0)$ is a box with side either 3 or 4 in each axis direction (see Figure 6).

In each case the lattice vertices forming the box around p can be naturally classified as corner, edge or face vertices. I claim that in a Delaunay tetrahedralization of S, p always connects exactly to the convex hull of the box face vertices only. This is because the box corner and box edge vertices are isolated from p by Delaunay tetrahedra. The tetrahedra whose circumspheres come closest to containing p occur in case 1. An example is a tetrahedron with circumsphere center $(\frac{1}{2}, \frac{1}{2}, \frac{3}{2})$ and radius $\sqrt{3}/2$. The circumsphere goes through the point $(\sigma, \sigma, \frac{3}{2})$ which is avoided by $p \in (\sigma, \frac{3}{2}]^3$. This is why I chose this value of σ.

The angle bounds are more difficult to prove formally than in Lemmas 5 and 6 because the coordinates of point p can vary continuously in a range. Nonetheless the dihedral angles are smooth functions of the position of p and the extrema are easily found. See Table 1 for a summary of the bounds in each case. Figure 7 shows tetrahedra that achieve the lower bounds. □

Corollary 9. *Let $p \in \mathbf{R}^3$ and $k \in \mathbf{Z}$. Let t be a tetrahedron with vertices in the set $S = \{p\} \cup (\mathrm{SC}_k \backslash R(p,k))$. Suppose that p is not inside the circumsphere of t. If t has a dihedral angle smaller than $30°$ or larger than $127.903°$, or a radius-edge ratio larger than 1.368, then there exists a point q of $\mathrm{SC}_k \backslash R(p,k)$ inside the circumsphere of t.*

Proof. By Theorem 8, t cannot be Delaunay in S. Therefore there exists a point q in S inside the circumsphere of t. Since we assumed that p is not inside the circumsphere of t, $q \neq p$. So $q \in \mathrm{SC}_k \backslash R(p,k)$. □

case	box face vertices	min dihedral	max dihedral	max radius-edge ratio	achieved at $p =$
1	24	$\approx 31.962°$	$\approx 125.264°$ (*)	≈ 1.198	(σ, σ, σ)
2	32	$\approx 30.129°$	$\approx 125.264°$ (*)	≈ 1.267	$(\sigma, \sigma, 3-\sigma)$
3	42	$30°$	$< 127.903°$	≈ 1.329	$(\sigma, 3-\sigma, 3-\sigma)$
4	54	$\approx 34.785°$	$\approx 125.264°$ (*)	< 1.368	$(3-\sigma, 3-\sigma, 3-\sigma)$

Table 1: Four cases in the proof of Theorem 8. In all cases the dihedral angles are in the interval $[30°, 127.903°]$ and the radius-edge ratios are less than 1.368. The bounds marked (*) are intrinsic to the simple cubic lattice and are achieved independently of the position of p.

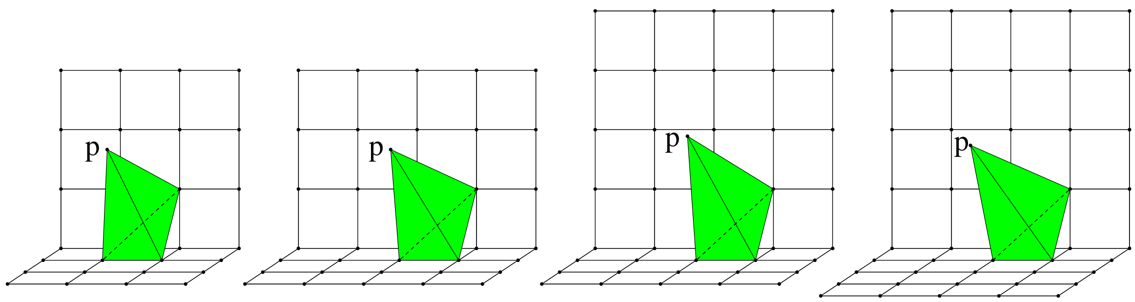

Figure 7: Tetrahedra achieving a minimum dihedral angle in each case of Theorem 8. Only two sides of the box around the input vertex p are shown. The minimum dihedral angles and coordinates of p appear in Table 1. (The coordinate system is different for this figure.)

5.1 Algorithm

The algorithm is an extension of the algorithm of Section 4.1. The detailed algorithm is described in Figure 8. At places the algorithm makes queries of the form *"Is there an input vertex in the region $R(w, k)$?"* This kind of query can be answered efficiently by using the Delaunay tetrahedralization as a search structure.

As before, new vertices are inserted for one of two reasons: the sizing function, or a bad-quality tetrahedron. The latter is separated into three cases referring to specific parts of Figure 8:

In case 1, an attempt is made to insert a new vertex from a strictly coarser lattice, to obtain good mesh grading. The attempt is aborted if the candidate for insertion is "too close" to an input vertex.

In case 2, there is an input vertex nearby and the algorithm has permission to insert a new vertex from a lattice that is just as fine as the finest vertex of the bad-quality tetrahedron, or even one level finer (inserting a vertex from SC_k when the finest tetrahedron vertex label is BCC_{k+1}). Corollary 9 allows an analysis of this case.

In case 3, there are two input vertices nearby. Corollary 9 cannot be used because it can only deal with one nearby input vertex at a time. We have no other choice than to refine the mesh with a lattice vertex and have the situation slowly simplify itself.

5.2 Analysis

The analysis is structured in the same way as in Section 4.2. We find positive constants a and b such that (1) holds, and derive the constant c used in Definition 2.

We perform a separate analysis for each type of the inserted vertex w. The first three analyses are identical to what was done in Section 4.2 (except for type 1.2 where k is defined to be one less).

5.2.1 *Insertion due to size*

As in Section 4.2, we require that $\sqrt{3}c \leq a$, so that $\mathrm{lfs}(w) \leq 2^j a$ and the insertion preserves (1).

5.2.2 *Type 1.1 insertion*

As in Section 4.2 (for a Type 1 insertion), we require that $a + \sqrt{5} \leq 2b$, so that by induction $\mathrm{lfs}(w) \leq 2^{k+1} b$.

5.2.3 *Type 1.2 insertion*

The new vertex w has label SC_{k+1} at a distance at most $2r(\mathrm{SC}_{k+1}) = 2^{k+1}\sqrt{3}$ from v with label BCC_{k+1}. So $\mathrm{lfs}(w) \leq \mathrm{lfs}(v) + \mathrm{dist}(v, w) \leq 2^{k+1} b + 2^{k+1}\sqrt{3}$. We require that $b + \sqrt{3} \leq a$, so that by induction $\mathrm{lfs}(w) \leq 2^{k+1} a$.

5.2.4 *Type 3 insertion*

w has label SC_j for some j. By construction, the circumradius of t is at most $2(r(\mathrm{SC}_j) + 2^j \rho)$.

If t has at least 2 input vertices, then $\mathrm{lfs}_i(w) \leq 2(r(\mathrm{SC}_j) + 2^j \rho) + r(\mathrm{SC}_j) = 2^j(3\sqrt{3}/2 + 2\rho)$.

Else, t has at least 3 refinement vertices with labels SC_k or coarser, so the circumradius of t is at least $e(\mathrm{SC}_k)/2 = 2^k/2$. It is also at most $2(r(\mathrm{SC}_j) + 2^j\rho)$, so $2^k \leq 2^j 4(r(\mathrm{SC}_0) + \rho) = 2^j(2\sqrt{3} + 4\rho)$.

From Table 2, the distance between v and u_1 (or u_2) is at most $2^k(4r(\mathrm{SC}_0) + 2\rho) = 2^k(2\sqrt{3} + 2\rho) \leq 2^j(2\sqrt{3} + 4\rho)(2\sqrt{3} + 2\rho)$. $\mathrm{dist}(v, w) \leq 2^j(3\sqrt{3}/2 + 2\rho)$. By the triangle inequality, the distance between w and u_1 or u_2 is at most $2^j((2\sqrt{3} + 4\rho)(2\sqrt{3} + 2\rho) + 3\sqrt{3}/2 + 2\rho)$. Let

$$\alpha = (2\sqrt{3} + 4\rho)(2\sqrt{3} + 2\rho) + 3\sqrt{3}/2 + 2\rho.$$

Then $\mathrm{lfs}_i(w) \leq 2^j \alpha$. We require that $\alpha \leq a$ so that $\mathrm{lfs}(w) \leq 2^j a$.

Input: A domain $\Omega \subset \mathbf{R}^3$ and a finite number of input vertices with arbitrary coordinates in Ω. If desired, a sizing function $s : \Omega \to (0, \infty]$.

Procedure: Compute a Delaunay tetrahedralization of the input vertices. Repeat the enforcement steps below in any order until none apply. Maintain a Delaunay tetrahedralization as new vertices are inserted. When finished, remove all tetrahedra whose interiors do not intersect Ω.

Size enforcement: If there exists a point $p \in \Omega$ such that a closed ball B of radius $s(p)$ centered at p is empty:

Let j be the largest integer such that $r(\mathrm{SC}_j) \leq s(p)$. By Lemma 3 there exists a point w of SC_j in B. **Insert** w **with label** SC_j into the Delaunay tetrahedralization.

Quality enforcement: If the Delaunay tetrahedralization contains a tetrahedron t whose interior intersects Ω and which has a dihedral angle smaller than $30°$ or larger than $135°$, or a radius-edge ratio larger than $\beta = 1.368$, then apply Case 1 if t has at least 1 refinement vertex, or Case 3 if t has at least 2 input vertices. (If both cases apply, then choose either.)

Case 1: t has at least 1 refinement vertex.

Let v be the refinement vertex of t with the finest label, according to the nesting of lattices.

1.1 If the label of v is SC_k for some k.

1.1.1 If the circumradius of t is larger than $r(\mathrm{BCC}_{k+1})$, then let B be a closed ball of radius $r(\mathrm{BCC}_{k+1})$ tangent at v inside the circumsphere of t. By Lemma 3 there exists a point w of BCC_{k+1} in $B \backslash \{v\}$. w is a candidate for insertion.

1.1.2 Else if t has at least one input vertex u_1, then go to Case 2.

1.1.3 Else (t has 4 refinement vertices). The radius-edge ratio of t is at most $r(\mathrm{BCC}_{k+1})/e(\mathrm{SC}_k) = \sqrt{5}/2 < \beta$, so t must have a dihedral angle smaller than $30°$ or larger than $135°$. This is impossible by Lemma 5, so this subcase never happens.

If the forbidden region $R(w, k)$ of the candidate w doesn't contain any input vertex, then **insert** w **with label** BCC_{k+1} **and type 1.1**. Else, let u_1 be an input vertex in the forbidden region and go to Case 2.

1.2 Else (the label of v is BCC_{k+1} for some k).

1.2.1 If the circumradius of t is larger than $r(\mathrm{SC}_{k+1})$, then let B be a closed ball of radius $r(\mathrm{SC}_{k+1})$ tangent at v inside the circumsphere of t. By Lemma 3 there exists a point w of SC_{k+1} in $B \backslash \{v\}$. w is a candidate for insertion.

1.2.2 Else if t has at least one input vertex u_1, then go to Case 2.

1.2.3 Else (t has 4 refinement vertices). The radius-edge ratio of t is at most $r(\mathrm{SC}_{k+1})/e(\mathrm{BCC}_{k+1}) = 1 < \beta$, so t must have a dihedral angle smaller than $30°$ or larger than $135°$. By Lemma 6 there exists a point w of SC_{k+1} inside the circumsphere of t. w is a candidate for insertion.

If the forbidden region $R(w, k+1)$ of the candidate w doesn't contain any input vertex, then **insert** w **with label** SC_{k+1} **and type 1.2**. Else, let u_1 be an input point in the forbidden region and go to Case 2.

Case 2: t has at least 1 refinement vertex. v is the finest refinement vertex of t, and has label SC_k or BCC_{k+1}. u_1 is an input vertex "not too far" from v.

2.1 If the circumradius of t is larger than $r(\mathrm{SC}_k) + 2^k\rho/2$, then let B be a closed ball of radius $r(\mathrm{SC}_k) + 2^k\rho/2$ tangent at v inside the circumsphere of t. Let B' be a closed ball of radius $r(\mathrm{SC}_k)$ lying in B as far as possible from u_1. By Lemma 3 there exists a point w of SC_k in $B' \backslash \{v\}$. By construction the forbidden region $R(w, k)$ doesn't contain u_1. w is a candidate for insertion.

2.2 Else if t has no input vertex except possibly u_1, by Corollary 9 there exists a point w of SC_k inside the circumsphere of t such that its forbidden region $R(w, k)$ doesn't contain u_1. w is a candidate for insertion.

2.3 Else (t has an input vertex $u_2 \neq u_1$) go to Case 3.

If the forbidden region of $R(w, k)$ doesn't contain *any* input vertex, then **insert** w **with label** SC_k **and type 2**. Else let u_2 be an input vertex in the forbidden region. Go to Case 3.

Case 3: There are at least 2 input vertices u_1 and u_2 that are vertices of t, or that are "not too far" from the circumsphere of t.

Let j be the largest integer such that $r(\mathrm{SC}_j) + 2^j\rho$ is less than the circumradius of t. By Lemma 3 there exists a point w of SC_j in a closed ball of radius $r(\mathrm{SC}_j)$ centered at the circumcenter of t. **Insert** w **with label** SC_j **and type 3**. By construction no input point is in the forbidden region $R(w, j)$.

Figure 8: An algorithm to create a superset mesh of a domain Ω with quality tetrahedra. The final mesh respects specified input vertices and, if desired, a user-defined sizing function.

subcase	bound if the insertion succeeds	bound if the algorithm jumps to next case
1.1.1	$\text{dist}(v, w) \le 2r(\text{BCC}_{k+1})$	$\text{dist}(v, u_1) \le 2r(\text{BCC}_{k+1}) + 2^k\rho$
1.1.2	n/a	$\text{dist}(v, u_1) \le 2r(\text{BCC}_{k+1})$
1.2.1	$\text{dist}(v, w) \le 2r(\text{SC}_{k+1})$	$\text{dist}(v, u_1) \le 2r(\text{SC}_{k+1}) + 2^{k+1}\rho$
1.2.2	n/a	$\text{dist}(v, u_1) \le 2r(\text{SC}_{k+1})$
1.2.3	$\text{dist}(v, w) < 2r(\text{SC}_{k+1})$	$\text{dist}(v, u_1) < 2r(\text{SC}_{k+1}) + 2^{k+1}\rho$
2.1	$\text{dist}(v, w) \le 2r(\text{SC}_k) + 2^k\rho$	$\text{dist}(v, u_2) \le 2r(\text{SC}_k) + 2^k \cdot 2\rho$
2.2	$\text{dist}(v, w) < 2r(\text{SC}_k) + 2^k\rho$	$\text{dist}(v, u_2) < 2r(\text{SC}_k) + 2^k \cdot 2\rho$
2.3	n/a	$\text{dist}(v, u_2) \le 2r(\text{SC}_k) + 2^k\rho$

Table 2: Distance bounds in cases 1 and 2 of the algorithm of Figure 8, while trying to refine a bad-quality tetrahedron t. v is the refinement vertex of t with the finest label, either SC_k or BCC_{k+1}, for some k. If the insertion succeeds then w is the new vertex; otherwise the algorithm identifies a nearby input vertex u_1 or u_2 and jumps to the next case.

5.2.5 Type 2 insertion

This case is the most difficult because in the neighborhood of an input vertex, new lattice vertices are not necessarily coarser than their neighbors, so we have to travel a bit to find either a coarser lattice vertex or a second input vertex.

For the purpose of analysis, each new vertex w produced by the algorithm will be assigned a *parent* $p(w)$ and a *nearby input vertex* $n(w)$, depending on the type of inserted vertex w.

Type 1.1: Let $p(w) = v$, $n(w) = n(v)$.
Type 1.2: Let $p(w) = \text{null}$, $n(w) = \text{null}$.
Type 2: Let $p(w) = v$, $n(w) = u_1$.
Type 3: Let $p(w) = \text{null}$, $n(w) = \text{null}$.

LEMMA 10. *If $p(w) \ne \text{null}$ then*

$$\text{dist}(p(w), w) \le 2^k(\sqrt{3} + \rho).$$

PROOF. Since $p(w) \ne \text{null}$, w is of type 1.1 or 2. We look at the values of $\text{dist}(v, w)$ in Table 2. The bound is simply the largest of these two:
Type 1.1: $2r(\text{BCC}_{k+1}) = 2^k\sqrt{5}$;
Type 2: $2r(\text{SC}_k) + 2^k\rho = 2^k(\sqrt{3} + \rho)$. $\square$

LEMMA 11. *If $n(w) \ne \text{null}$ then*

$$\text{dist}(n(w), w) \le 2^k(\sqrt{5} + 2\sqrt{3} + 2\rho).$$

The tighter bound

$$\text{dist}(n(w), w) \le 2^k(2\sqrt{3} + 2\rho)$$

is valid if w is known to be of type 2.

PROOF. Since $n(w) \ne \text{null}$, w is of type 1.1 or 2.

For type 2, we look at the largest possible value of $\text{dist}(v, u_1)$ in Table 2. The two candidates are $2r(\text{BCC}_{k+1}) + 2^k\rho = 2^k(\sqrt{5} + \rho)$ and $2r(\text{SC}_{k+1}) + 2^{k+1}\rho = 2^k(2\sqrt{3} + 2\rho)$. The latter is the largest.

For type 1.1, we use the triangle inequality $\text{dist}(n(w), w) \le \text{dist}(n(v), v) + \text{dist}(p(w), w)$ where $n(v) = n(w)$ and $p(w) = v$. v must be of type 2 because $n(v) \ne \text{null}$ and w has label SC_k, so $\text{dist}(n(v), v) \le 2^k(2\sqrt{3} + 2\rho)$. Because w is of type 1.1, $\text{dist}(p(w), w) \le 2^k\sqrt{5}$. By combining these inequalities the result follows. $\square$

Now follow the chain of parents as follows: initialize $w' := w$. While $p(w')$ is of type 1.1 or 2, and $n(p(w')) = n(w)$, do $w' := p(w')$.

Note that by construction, every vertex of the chain from w to w' has label SC_k or BCC_{k+1} with the *same* k.

If $p(w')$ is of type 1.2 (with label SC_k) or 3, then: using bounds derived for these cases, $\text{lfs}(p(w')) \le 2^k(b + \sqrt{3})$ or $\text{lfs}(p(w')) \le 2^k\alpha$, so $\text{lfs}(p(w')) \le 2^k \max(b + \sqrt{3}, \alpha)$.

$\text{dist}(p(w'), w') \le 2^k(\sqrt{3} + \rho)$ by Lemma 10.
$\text{dist}(w', n(w')) \le 2^k(\sqrt{5} + 2\sqrt{3} + 2\rho)$ by Lemma 11.
$\text{dist}(n(w), w) \le 2^k(2\sqrt{3} + 2\rho)$ by Lemma 11.
$n(w') = n(w)$ because otherwise the *while* loop would have stopped on the previous iteration.

By combining these inequalities we obtain $\text{dist}(p(w'), w) \le 2^k(\sqrt{5} + 5\sqrt{3} + 5\rho)$.

By Lemma 1 we have $\text{lfs}(w) \le \text{lfs}(p(w')) + \text{dist}(p(w'), w) \le 2^k \max(b + \sqrt{3}, \alpha) + 2^k(\sqrt{5} + 5\sqrt{3} + 5\rho)$.

We require $\max(b + \sqrt{3}, \alpha) + \sqrt{5} + 5\sqrt{3} + 5\rho \le a$ so that by induction $\text{lfs}(w) \le 2^k a$.

Else ($p(w')$ is of type 1.1 or 2 and $n(p(w')) \ne n(w)$):
$\text{dist}(n(p(w')), p(w')) \le 2^k(\sqrt{5} + 2\sqrt{3} + 2\rho)$ by Lemma 11.
$\text{dist}(p(w'), w') \le 2^k(\sqrt{3} + \rho)$ by Lemma 10.
$\text{dist}(w', n(w')) \le 2^k(\sqrt{5} + 2\sqrt{3} + 2\rho)$ by Lemma 11.
$\text{dist}(n(w), w) \le 2^k(2\sqrt{3} + 2\rho)$ by Lemma 11.
$n(w') = n(w)$ because otherwise the while-loop would have stopped on the previous iteration.

By combining these inequalities we obtain

$$\text{dist}(n(p(w')), w) \le 2^k(2\sqrt{5} + 7\sqrt{3} + 7\rho).$$

$n(p(w'))$ and $n(w)$ are two distinct input vertices. This implies $\text{lfs}_i(w) \le 2^k(2\sqrt{5} + 7\sqrt{3} + 7\rho)$.

We require $2\sqrt{5} + 7\sqrt{3} + 7\rho \le a$, so that $\text{lfs}(w) \le 2^k a$.

5.2.6 Guarantee

The requirements can be satisfied by setting $a = \alpha + \sqrt{5} + 5\sqrt{3} + 5\rho$, $b = (a + \sqrt{5})/2$ and $c = a/\sqrt{3}$. Theorem 7 does not apply directly because of the input vertices (which have no labels). We need to add two cases to the proof of Theorem 7:

1. If v and w are both input vertices, then we directly have $\text{dist}(v, w) \ge \text{lfs}(v)$.
2. If v is a refinement vertex and w is an input vertex, then:

   In case v has label SC_k: $\text{dist}(v, w) \ge 2^k\sigma$, where $\sigma = (2 + \sqrt{6})/4$.

   In case v has label BCC_k: $\text{dist}(v, w) \ge 2^k\sigma/2$.

The bound given by Theorem 7 becomes

$$\min\left(\tfrac{1}{1+a}, \tfrac{1}{1+2b/\sigma}\right)\text{lfs}(v).$$

By applying this, we deduce that during the course of the algorithm the distance between any vertex v and its nearest neighbor is at least $\text{lfs}(v)/109.8$. This implies that the algorithm terminates, as argued in Section 4.2.4.

6. CONCLUSION

The use of two types of lattices is complicated but appears to be necessary to obtain a lower bound of 30° on dihedral angles. For a system that would give good grading based uniquely on one of the simple, body-centered, or face-centered cubic lattices I could only obtain these respective approximate lower bounds: 17.023°, 14.312°, and 14.197°.

I hope that this work is just a first step toward a lattice refinement algorithm for domains with boundary constraints. As a step in that direction, I showed in Section 5 how internal input vertices can be supported. I believe that planar constraints can also be supported without too much difficulty. Sharp features like segments and corners are more challenging because when a segment is split, there is only one degree of freedom for the position of the split point. Of course, the dihedral angle guarantees don't have to stay as good as 30° and 135°.

It would be interesting to perform experiments with the algorithm of Figure 8 to see how much refinement is done in practice compared to standard Delaunay refinement, and when trying to eliminate slivers from an already refined mesh.

7. ACKNOWLEDGMENTS

I would like to thank Jonathan Shewchuk for helpful discussions, comments, and suggestions.

8. REFERENCES

[1] M. Bern, D. Eppstein, and J. R. Gilbert. Provably Good Mesh Generation. *Journal of Computer and System Sciences*, 48(3):384–409, June 1994.

[2] S.-W. Cheng and T. K. Dey. Quality Meshing with Weighted Delaunay Refinement. *SIAM Journal on Computing*, 33(1):69–93, 2003.

[3] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng. Sliver Exudation. *Journal of the ACM*, 47(5):883–904, Sept. 2000.

[4] L. P. Chew. Guaranteed-Quality Delaunay Meshing in 3D. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, pages 391–393, Nice, France, June 1997. Association for Computing Machinery.

[5] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam, 1978.

[6] H. Edelsbrunner and D. Guoy. An Experimental Study of Sliver Exudation. In *Tenth International Meshing Roundtable*, pages 307–316, Newport Beach, California, Oct. 2001. Sandia National Laboratories.

[7] H. Edelsbrunner, X.-Y. Li, G. Miller, A. Stathopoulos, D. Talmor, S.-H. Teng, A. Ungor, and N. Walkington. Smoothing and Cleaning Up Slivers. In *Proceedings of the 32nd Annual Symposium on the Theory of Computing*, pages 273–278, Portland, Oregon, May 2000. Association for Computing Machinery.

[8] D. A. Field. Implementing Watson's Algorithm in Three Dimensions. In *Proceedings of the Second Annual Symposium on Computational Geometry*, pages 246–259, Yorktown Heights, New York, June 1986. Association for Computing Machinery.

[9] D. A. Field. Qualitative Measures for Initial Meshes. *International Journal for Numerical Methods in Engineering*, 47:887–906, 2000.

[10] D. A. Field and W. D. Smith. Graded Tetrahedral Finite Element Meshes. *International Journal for Numerical Methods in Engineering*, 31:413–425, 1991.

[11] X.-Y. Li and S.-H. Teng. Generating Well-Shaped Delaunay Meshes in 3D. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms*, pages 28–37, Washington, D.C., Jan. 2001. Association for Computing Machinery.

[12] A. Liu and B. Joe. Relationship between Tetrahedron Shape Measures. *BIT*, 34:268–287, 1994.

[13] G. L. Miller, D. Talmor, S.-H. Teng, N. Walkington, and H. Wang. Control Volume Meshes Using Sphere Packing: Generation, Refinement and Coarsening. In *Fifth International Meshing Roundtable*, pages 47–61, Pittsburgh, Pennsylvania, Oct. 1996.

[14] S. A. Mitchell and S. A. Vavasis. Quality Mesh Generation in Higher Dimensions. *SIAM Journal on Computing*, 29(4):1334–1370, 2000.

[15] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra. In *Twelfth International Meshing Roundtable*, pages 103–114, Santa Fe, New Mexico, Sept. 2003.

[16] D. J. Naylor. Filling Space with Tetrahedra. *International Journal for Numerical Methods in Engineering*, 44:1383–1395, 1999.

[17] J. Ruppert. A Delaunay Refinement Algorithm for Quality 2-Dimensional Mesh Generation. *Journal of Algorithms*, 18(3):548–585, May 1995.

[18] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and Approximating Implicit Surfaces from Polygon Soup. *ACM Transactions on Graphics*, 23(3):896–904, Aug. 2004. Special issue on Proceedings of ACM SIGGRAPH 2004.

[19] J. R. 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.

[20] J. R. Shewchuk. What Is a Good Linear Element? Interpolation, Conditioning, and Quality Measures. In *Eleventh International Meshing Roundtable*, pages 115–126, Ithaca, New York, Sept. 2002. Sandia National Laboratories.

[21] A. Üngör. Off-Centers: A New Type of Steiner Points for Computing Size-Optimal Guaranteed-Quality Delaunay Triangulations. In *Latin American Theoretical Informatics*, pages 152–161, Buenos Aires, Argentina, Apr. 2004.

[22] M. A. Yerry and M. S. Shephard. Automatic Three-Dimensional Mesh Generation by the Modified-Octree Technique. *International Journal for Numerical Methods in Engineering*, 20:1965–1990, 1984.