GENERALIZED INSERTION REGION GUIDES FOR DELAUNAY MESH REFINEMENT *

ANDREY N. CHERNIKOV† AND NIKOS P. CHRISOCHOIDES‡

Abstract. Mesh generation by Delaunay refinement is a widely used technique for constructing guaranteed quality triangular and tetrahedral meshes. The quality guarantees are usually provided in terms of the bounds on circumradius-to-shortest edge ratio and on the grading of the resulting mesh. Traditionally circumcenters of skinny elements and middle points of boundary faces and edges are used for the positions of inserted points. However, recently variations of the traditional algorithms are being proposed that are designed to achieve certain optimization objectives by inserting new points in neighborhoods of the center points. In this paper we propose a general approach to the selection of point positions by defining one-, two-, and three-dimensional selection regions such that any point insertion strategy based on these regions is automatically endowed with the theoretical guarantees proven here. In particular, for the input models defined by planar linear complexes under the assumption that no input angle is less than $90°$, we prove the termination of the proposed generalized algorithm, as well as fidelity and good grading of the resulting meshes.

Key words. Delaunay refinement, mesh generation

AMS subject classifications. 65L50, 65D18, 68W10

1. Introduction. Delaunay refinement is a popular mesh generation method which allows for the mathematical proofs of termination and good grading. These properties are particularly important for parallel and/or time critical applications when human intervention and reruns are prohibitively time consuming.

Delaunay refinement algorithms are based on the idea of inserting new points into the mesh to improve the aggregate quality of elements (triangles in two dimensions or tetrahedra in three dimensions). Quality is traditionally defined as the ratio of the circumradius of the element to the length of its shortest edge [7, 9, 13, 19, 21]. The use of this measure leads to the improvement of the minimum angle in two dimensions which helps to improve the conditioning of the stiffness matrix used by a field solver. In three dimensions this measure does not yield such direct benefits, however, it has been shown [13] that bounded circumradius-to-shortest edge ratio of mesh elements is sufficient to obtain optimal convergence rates for the solution of Poisson equation using the control volume method. The analysis and rigorous proofs of Delaunay refinement were pioneered by Ruppert [19] and Chew [6], and further developed by Shewchuk [20, 21].

One of the central questions in Delaunay refinement research has been the choice of the positions for the new points. The traditional approach uses circumcenters of mesh elements, however, a number of other locations have been used to achieve various mesh optimizations [3, 7, 10, 11, 17, 23]. Our goal is to provide a single theoretical framework which makes it possible to develop multiple custom point placement techniques by means of defining special regions, such that any Delaunay refinement-based technique which places points in these regions will automatically be endowed with termination and good grading guarantees. Then the development of parallel versions

*This work was supported in part by NSF grants CCF-1139864 , CCF-1139864, and CSI-1139864, as well as by the John Simon Guggenheim Foundation and the Richard T. Cheng Endowment.

†Department of Computer Science, Old Dominion University, Norfolk, Virginia 23529 (achernik@cs.odu.edu).

‡Department of Computer Science, Old Dominion University, Norfolk, Virginia 23529 (nikos@cs.odu.edu).

of all these techniques is reduced to the parallelization of this single generalized approach [4].

In the previous work [3] we listed two-dimensional and three-dimensional point insertion methods and suggested the use of two-dimensional and three-dimensional regions, respectively, which we called selection disks. These regions were defined for the highest dimension only, and, hence, we termed the approach semi-generalized. Later [5] we developed a fully generalized approach for two dimensions, i.e., we defined the selection regions for both one-dimensional elements (segments) and two-dimensional elements (triangles). We also pursued the constrained Delaunay approach to improve the angle bound and to extend the selection regions in two dimensions [8]. In this paper we develop a three-dimensional fully generalized algorithm which utilizes selection regions simultaneously for one-, two-, and three-dimensional elements. The theory we develop here offers the following non-trivial contributions:

- We formulate a novel three-dimensional fully generalized Delaunay refinement algorithm.
- We prove the geometric fidelity of the meshes produced by this algorithm, i.e., that new points are always inserted inside the domain.
- We prove that this algorithm terminates, and moreover produces well graded meshes.

The rest of the paper is organized as follows. In Section 2 we introduce the background, the basic facts, and the definitions. In Section 3 we define the proposed fully generalized three-dimensional Delaunay refinement algorithm. Then in Section 4 we prove the geometric fidelity of the algorithm, in Section 5 we prove that the algorithm terminates, and in Section 6 we prove that it produces well graded meshes. Section 7 concludes the paper.

2. Background. We consider the input domain Ω described by a Planar Linear Complex (PLC) [7, 9, 14, 19, 21]. A PLC $\mathcal{X}$ consists of a set of vertices, a set of straight line segments, and a set of planar facets. $\mathcal{X}$ describes a nonconvex bounded polyhedral domain with holes. Each element of $\mathcal{X}$ is considered *constrained* and must be preserved during the construction of the mesh, although it can be subdivided into smaller elements through the insertion of new vertices. The vertices of $\mathcal{X}$ must be a subset of the final set of vertices in the mesh.

Let the mesh $\mathcal{M}_{\mathcal{X}}$ for the given PLC $\mathcal{X}$ consist of a set V of vertices and a set T of tetrahedra formed on the vertices from V. To measure the quality of a tetrahedron t we follow the traditional approach [7, 9, 14, 19, 21] and use the circumradius-to-shortest edge ratio, or radius-edge ratio for short, which we denote as $\rho(t)$. The algorithm takes as input an upper bound $\bar{\rho} \geq 2$ on radius-edge ratio and outputs a mesh with all tetrahedra satisfying this bound. We call the tetrahedra that violate this bound *skinny*.

Consider an n-simplex ξ ($n = 1, 2, 3$) embedded in three dimensions, which is a straight line segment, a triangular face, or a tetrahedron. Let us call the open ball corresponding to the smallest sphere which passes through the vertices of ξ the *circumball* of ξ. The center of the circumball is called the *circumcenter* of the element.

Definition 2.1 (Delaunay simplex [20]). *An edge, triangular face, or tetrahedron whose vertices are members of V is said to be* Delaunay *if there exists an empty sphere that passes through all of its vertices.*

Here a sphere is considered empty if it does not contain any of the mesh vertices in its interior; however, parts of edges and faces are allowed.

If all simplices in a 3D mesh are Delaunay, then the whole mesh is said to satisfy

BowyerWatson($\mathcal{M}$, $\mathbf{v}$)
Input: $\mathcal{M} = (V^n, T^n)$ is the mesh of PLC $\mathcal{X}$ at time step n before the insertion of $\mathbf{v}$
Output: $\mathcal{M} = (V^{n+1}, T^{n+1})$ after the insertion of $\mathbf{v}$
1: $V^{n+1} \leftarrow V^n \cup \{\mathbf{v}\}$
2: $T^{n+1} \leftarrow T^n \setminus \mathcal{C}_{T^n}(\mathbf{v}) \cup \{(\mathbf{v}\xi) \mid \xi \in \partial\mathcal{C}_{T^n}(\mathbf{v})\}$
// Here $(\mathbf{v}\xi)$ is the tetrahedron obtained by connecting vertex $\mathbf{v}$
// to the vertices of triangle ξ.

Fig. 2.1. *The Bowyer-Watson point insertion algorithm.*

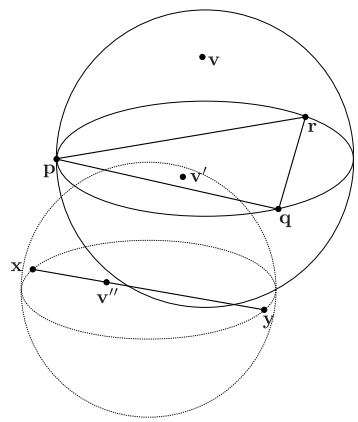

Fig. 2.2. *Encroachment in three dimensions.*

the *Delaunay property*. Moreover, all the 2D meshes of the PLC facets are also Delaunay in their respective planes.

Traditional Delaunay mesh generation algorithms start with the construction of the initial mesh, which conforms to $\mathcal{X}$, and then refine this mesh until it has no more skinny tetrahedra. The general idea of Delaunay refinement is to insert additional (so-called Steiner) points inside the circumballs of skinny tetrahedra, which leads to their removal, until they are gradually eliminated and replaced by better quality tetrahedra.

We will use the notion of *cavity* [9] which is the set of tetrahedra in the mesh whose circumballs include a given point $\mathbf{v}$. We will denote $\mathcal{C}_T(\mathbf{v})$ to be the cavity of $\mathbf{v}$ with respect to the set of tetrahedra T and $\partial\mathcal{C}_T(\mathbf{v})$ to be the set of triangles that form the boundary of the cavity, i.e., the triangles which belong to only one tetrahedron in $\mathcal{C}_T(\mathbf{v})$. We will use the Bowyer-Watson (B-W) point insertion algorithm [2, 24], which can be written shortly as in Figure 2.1.

Delaunay refinement algorithms observe special *encroachment* rules. In particular, if a Steiner point $\mathbf{v}$ is considered for insertion, but it lies within the circumball of a constrained subfacet ξ, $\mathbf{v}$ is not inserted but a point on the facet containing ξ is inserted instead. Similarly, if $\mathbf{v}$ is inside the circumball of a constrained subsegment, then a point inside this subsegment is inserted instead. In contrast to faces and edges, there can be no encroached vertices. Consider the example in Figure 2.2. The new Steiner point $\mathbf{v}$ is inside the circumball of a constrained face $\mathbf{pqr}$. In this case, $\mathbf{v}$ is rejected and the algorithm attempts to insert point $\mathbf{v}'$ in the facet containing triangle $\mathbf{pqr}$. If $\mathbf{v}'$ does not encroach upon any constrained subsegments, it is inserted into the mesh. If, however, it encroaches upon a constrained subsegment, which is $\mathbf{xy}$ in our

```
PointInsertion(𝒳, ℳ, v)
Input: 𝒳 is the input PLC
       ℳ is a Delaunay mesh of 𝒳 before the insertion of v
Output: ℳ after the insertion or the rejection of v
if v encroaches upon some constrained subsegments
    Mark one of these subsegments as encroached
elseif v encroaches upon some constrained subfaces
    Mark the subface containing the projection of v as encroached
else
    BowyerWatson(ℳ, v)
endif
```

Fig. 2.3. *The point insertion scheme which covers encroachment.*

example, $\mathbf{v}'$ is also rejected and point $\mathbf{v}''$ in $\mathbf{xy}$ is inserted. With each boundary face we associate an inside and an outside normal vector. Also, for each tetrahedron we consider its centroid, which is obviously inside both the tetrahedron and the domain. If $\mathbf{v}$ is a point chosen to remove a skinny tetrahedron t, then a boundary face ξ is considered encroached upon by $\mathbf{v}$ if and only if $\mathbf{v}$ is inside the circumball of ξ and the centroid of t is towards the inside direction of ξ. Similar rules are used for other encroachment instances.

These encroachment rules serve two related goals: (1) to preserve all of the subfeatures of the PLC in the Delaunay mesh by means of ensuring they have empty spheres passing through their vertices (the smallest such sphere is chosen — the one corresponding to the circumball); (2) to ensure that inserted Steiner vertices are inside the mesh domain Ω (as can be seen from Theorem 4.1). The algorithm in Figure 2.3 shows the logic behind point insertion and rejection.

To make the proof of termination possible, Delaunay refinement needs to avoid infinite encroachment sequences. To prevent infinite encroachment on adjacent features, we follow the previous approaches and make a simplifying assumption that there are no angles less than 90° between adjacent features of the PLC. As a result, no point on one of the PLC features, say ξ, can be inside the circumball of an adjacent PLC feature, say ξ', independently of where the point is located in a selection ball defined in ξ.

A common approach for dealing with small input angles consists in isolating the regions around them and meshing their neighborhood with a predefined pattern. Bern, Eppstein, and Gilbert [1] cut off the acute interior angles by creating isosceles triangles at their apex vertices. Mitchell and Vavasis [15] in the context of their octree-based algorithm enclose such angles in protected boxes which are triangulated in a specific way. Ruppert [19] describes how to use circles with radius equal to a fraction of the local feature size to shield vertices at sharp interior angles by creating triangles around these vertices. Rand and Walkington [16] extend this approach to three-dimensional Delaunay refinement by using two types of protective regions, so-called collars and intestines. Miller, Pav, and Walkington [12] developed a variation of Ruppert's algorithm which allows for a larger angle bound by introducing a special rule for treating skinny triangles across from input angles. We expect that the application of these techniques can be combined with our analysis to eliminate the restriction of 90° minimum input angle, however this is beyond the scope of the current exposition.

Definition 2.2 (Local feature size [21]). *The* local feature size function $\mathrm{lfs}(\mathbf{v})$ *for a given point* $\mathbf{v}$ *is equal to the radius of the smallest ball centered at* $\mathbf{v}$ *that intersects*

two non-incident elements of the PLC.

The lfs () function satisfies the Lipschitz condition:

LEMMA 2.3 (Lipschitz condition, Lemma 2 in [21]). *Given any PLC and any two points* $\mathbf{u}$ *and* $\mathbf{v}$*, the following inequality holds:*

$$\mathrm{lfs}\,(\mathbf{v}) \leq \mathrm{lfs}\,(\mathbf{u}) + \|\mathbf{u} - \mathbf{v}\|. \tag{2.1}$$

Here and in the rest of the paper we use the standard Euclidean norm $\|\cdot\|$.

The traditional proofs of termination and of good grading of Delaunay refinement algorithms explore the relationships between the insertion radius of a point and that of its parent. Stated shortly, the *insertion radius* of point $\mathbf{v}$ is the length of the shortest edge connected to $\mathbf{v}$, and the *parent* is the vertex which is "responsible" for the insertion of $\mathbf{v}$ [21].

DEFINITION 2.4 (Insertion radius [21]). *The* insertion radius $R\,(\mathbf{v})$ *of point* $\mathbf{v}$ *is the length of the shortest edge which would be connected to* $\mathbf{v}$ *if* $\mathbf{v}$ *is inserted into the mesh, immediately after it is inserted.*

The following definition of a parent vertex generalizes the corresponding definition in [21]. In our analysis, even though the child is not necessarily the circumcenter of an encroached subfacet or subsegment, the parent is still defined to be the same vertex.

DEFINITION 2.5 (Parent of a Steiner vertex). *The* parent $\hat{\mathbf{v}}$ *of vertex* $\mathbf{v}$ *is the unique vertex which is defined as follows:*

- *If* $\mathbf{v}$ *is an input vertex, it has no parent.*
- *If* $\mathbf{v}$ *lies on an encroached subsegment or subfacet* ξ*, then* $\hat{\mathbf{v}}$ *is the earliest detected point (possibly rejected for insertion) encroaching upon* ξ*.*
- *If* $\mathbf{v}$ *is inserted to remove a chosen skinny tetrahedron* t*,* $\hat{\mathbf{v}}$ *is the most recently inserted vertex of the shortest edge of* t*.*

3. Generalized Delaunay Refinement: Definitions and Algorithm. We begin this section by introducing the definitions and the analysis tools that allow us to insert Steiner vertices in arbitrary positions within so-called selection balls.[1] We conclude the section by describing the complete algorithm.

DEFINITION 3.1 (Type-d vertex). *A vertex in three-dimensional space is considered of Type-*0*, Type-*1*, Type-*2*, or Type-*3 *if it is an input vertex, lies on an input segment, lies on an input planar face, or none of the above, respectively.*

DEFINITION 3.2 (Selection ball). *For a* d*-simplex* ξ *with circumcenter* $\mathbf{c}$ *and circumradius* r*, under one of the following conditions:*

- $d = 1$*; or*
- $d = 2$*; or*
- $d = 3$*, the shortest edge length of* ξ *is equal to* l*, and the radius-edge ratio* $\rho = r/l \geq \bar{\rho} \geq 2$*;*

the selection ball *of* ξ *is the closed* d*-ball with center* $\mathbf{c}$ *and radius* $r(1-\delta_d)$ *in the hyperplane defined by* ξ *(line, plane, or 3D space), where* δ_d *(*$d = 1, 2, 3$*) are constant parameters chosen such that*

$$\delta_1\delta_2\delta_3 \geq \frac{2}{\bar{\rho}}, \quad \delta_1 \leq 1,\ \delta_2 \leq 1,\ \delta_3 \leq 1. \tag{3.1}$$

See Figure 3.1 for an illustration.

[1] In this paper we deviate from our previously used term *selection disk* and use *selection ball* instead, as it appears to be more intuitively understandable.

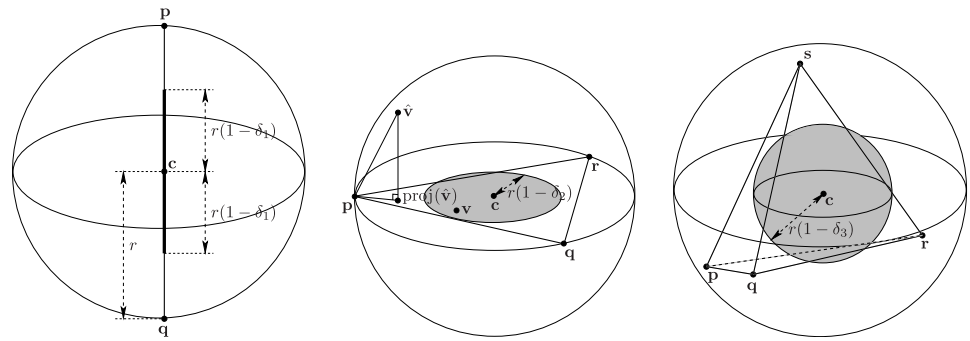

FIG. 3.1. *(Left) One-dimensional selection ball (thick segment). (Center) Two-dimensional selection ball (shaded circle). Also illustrates the proof of Theorem 5.3. (Right) Three-dimensional selection ball (shaded sphere).*

The inequality limiting the product in (3.1) arises from the requirement imposed by the termination condition (see Section 5), i.e., that the algorithm does not create a sequence of ever shorter edges. As can be seen later, in the worst case a new edge length is a multiple of $\delta_1/\sqrt{2} \cdot \delta_2/\sqrt{2} \cdot \delta_3\bar{\rho}$ of some existing edge length, and this multiplier needs to be greater than or equal to one.

DEFINITION 3.3 (Originating vertex). *The* originating vertex *is an inserted (i.e., not rejected) vertex of Type-d ($d = 0, 1, 2$) which either has no parent or has a parent of Type-k ($k = 0, 1, 2$) which lies on a PLC feature non-adjacent to the one containing this vertex.*

For example, all vertices of the PLC satisfy the definition of an originating vertex since they are inserted into the mesh (before the Delaunay refinement phase) and have no parents.

The following definition is adapted and modified from [22].

DEFINITION 3.4 (Parent sequence). *The* parent sequence *of a vertex $\mathbf{v} = \mathbf{v}_1$ is a sequence of vertices $\{\mathbf{v}_i\}_{i=1}^m$, such that all of the following conditions hold:*

(i) $\mathbf{v}_1$ *can be a vertex of any type,*

(ii) $\mathbf{v}_{i+1}$ *is the parent of* $\mathbf{v}_i$*, for* $i = 1, \ldots, m-1$*,*

(iii) $\mathbf{v}_m$ *is an originating vertex.*

The complete Generalized Delaunay Refinement algorithm is presented in Figure 3.2.

4. Proof of Fidelity. In this section we prove that the algorithm maintains geometric fidelity to the domain Ω, in other words, all Steiner vertices are inserted inside Ω. The following theorem shows that the use of encroachment rules prevents Steiner points from being inserted outside Ω. In other words, a point is either encroaching and rejected, or inside Ω.

THEOREM 4.1. *Let t be a d-dimensional simplex of a d-dimensional Delaunay mesh ($d = 2, 3$) of domain Ω bounded by a set Γ of $(d-1)$-dimensional simplices. Let $\mathbf{v}$ be an arbitrary point inside the d-dimensional circumball of t considered for insertion by the stated GDR algorithm under the specified encroachment rules and priority in point insertion. Then either $\mathbf{v} \in \Omega$, or $\mathbf{v}$ encroaches upon some $(d-1)$-dimensional simplex $\xi \in \Gamma$.*

Proof. We present a proof for 3D, and the 2D case follows trivially.

The proof is by contradiction. For the sake of contradiction, assume that $\mathbf{v}$ is outside of Ω and does not encroach upon any of the boundary triangles. Below we

GeneralizedDelaunayRefinement($\mathcal{X}$, $\bar{\rho}$, δ_1, δ_2, δ_3, $\kappa_1()$, $\kappa_2()$, $\kappa_3()$, $\mathcal{M}$)
Input: $\mathcal{X}$ is the PLC which encloses the domain Ω
$\bar{\rho}$ is the upper bound on radius-edge ratio, $\bar{\rho} \geq 2$
δ_1, δ_2, δ_3 are the parameters that define selection regions, see (3.1)
$\kappa_1()$, $\kappa_2()$, $\kappa_3()$ are user-defined functions which return specific positions for Steiner points within respective selection balls
$\mathcal{M} = (V, T)$ is an initial Delaunay mesh of $\mathcal{X}$, where V is the set of vertices and T is the set of tetrahedra
Output: A refined Delaunay mesh $\mathcal{M}$ which respects the bound $\bar{\rho}$

```
Let EncroachedSegments() return the current set of encroached subsegments
Let EncroachedFaces() return the current set of encroached triangular subfaces
Let SkinnyTetrahedra() return the current set of skinny tetrahedra in T
while (EncroachedSegments()≠ ∅ or
        EncroachedFaces()≠ ∅ or
        SkinnyTetrahedra()≠ ∅)
    if EncroachedSegments()≠ ∅
        Pick s ∈ EncroachedSegments()
        v ← κ1(δ1, s)
        PointInsertion(X, M, v)
    elseif EncroachedFaces()≠ ∅
        Pick f ∈ EncroachedFaces()
        v ← κ2(δ2, f)
        PointInsertion(X, M, v)
    elseif SkinnyTetrahedra()≠ ∅
        Pick t ∈ SkinnyTetrahedra()
        v ← κ3(δ3, t)
        PointInsertion(X, M, v)
    endif
endwhile
```

Fig. 3.2. *A high level description of the proposed Generalized Delaunay Refinement algorithm. Here we omit some details that do not influence the analysis, such as the location or type of the encroaching vertex* $\hat{\mathbf{v}}$*, and the implementation of updates and queries to the sets of segments, faces, and tetrahedra.*

will show that under this assumption at least one mesh vertex is inside the circumball of t, and therefore there is a contradiction with the Delaunay property.

Let $\mathbf{u}$ be an arbitrary vertex of t, and let $\mathbf{w}$ be the point of intersection of the straight line segment $\mathbf{uv}$ with the boundary Γ, see Figure 4.1. There are three intersection cases:

(A1) If $\mathbf{w}$ falls onto one of the mesh vertices, let $\mathbf{q} = \mathbf{w}$ be this vertex, and let ξ be one of the boundary triangles with vertex $\mathbf{q}$ and other two vertices $\mathbf{p}$ and $\mathbf{r}$.

(A2) If $\mathbf{w}$ falls onto a mesh edge, let $\mathbf{pq}$ be this edge, ξ be one of the boundary faces with edge $\mathbf{pq}$, and $\mathbf{r}$ be the third vertex of ξ.

(A3) Otherwise, $\mathbf{w}$ has to fall strictly inside some boundary face. Let ξ be this boundary face with vertices $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$.

In Figure 4.2 we show examples of the mutual position of t and ξ.

Let Ξ be the plane defined by ξ. Consider two circles: (1) open circle A' with center $\mathbf{C}'$ and radius R', which is the intersection of Ξ with the circumball of t ($\mathbf{C}$ is the circumcenter of t); and (2) open circle B with center c and radius r, which is the circumscribed circle of triangle ξ. These two circles lie in the same plane Ξ by

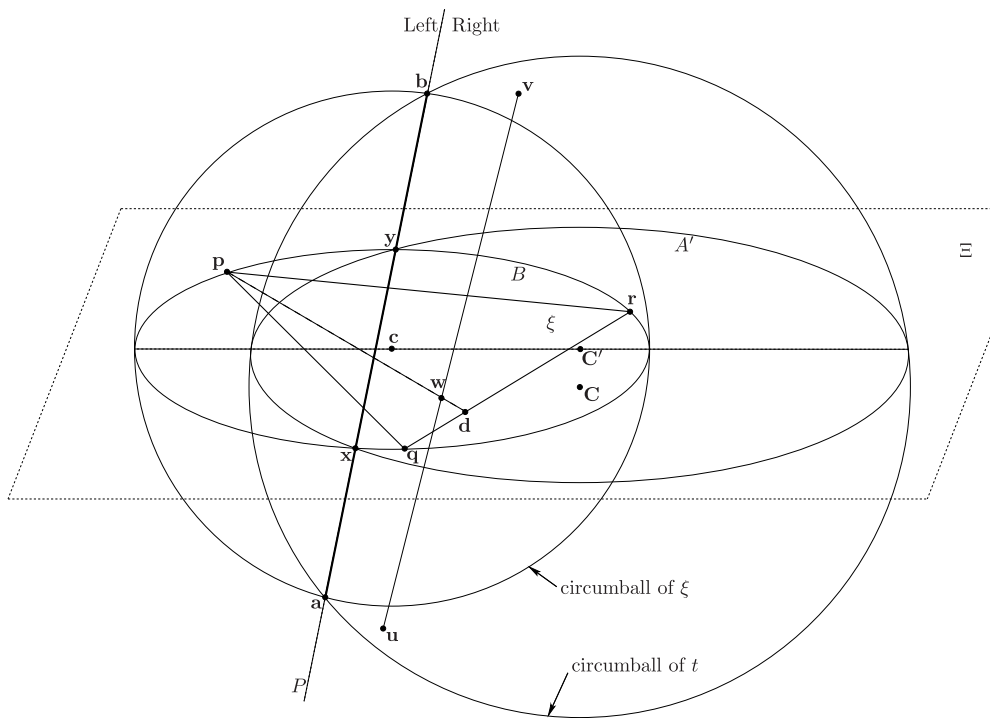

FIG. 4.1. *Illustration to the proof of Theorem 4.1.* Ξ *is the plane containing a boundary simplex* ξ *with vertices* $\mathbf{p}$, $\mathbf{q}$, *and* $\mathbf{r}$. *By construction, the upper side of* ξ *faces outside of the domain* Ω, *and the lower side of* ξ *faces inside of* Ω.

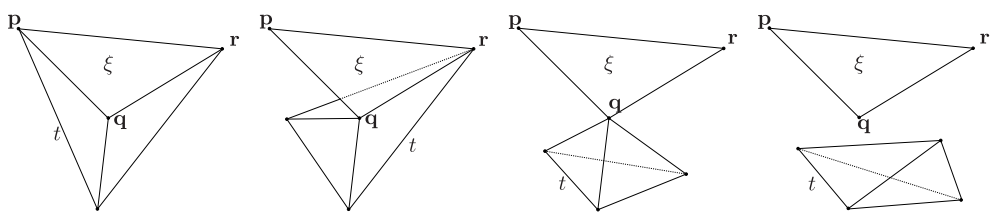

FIG. 4.2. *Examples of the mutual position of tetrahedron* t *and boundary face* ξ, *corresponding (left to right) to three, two, one, and zero shared vertices.*

construction.

If $\mathbf{c} = \mathbf{C}'$, we consider two cases:

(B1) If $r < R'$, vertices $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$ must be inside the circumball of t, and the proof is finished.

(B2) If $r \geq R'$, we consider two subcases:

(B2-a) Suppose $\mathbf{u} = \mathbf{w}$ (and therefore from (A1) $\mathbf{w} = \mathbf{q}$). At least one vertex (say, $\mathbf{s}$) of t must be distinct from $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$. $\mathbf{s}$ must be outside of the circumball of ξ due to the encroachment rules and on the other side of Ξ from $\mathbf{v}$ since t is inside Ω. Then $\mathbf{v}$ clearly cannot be inside the circumball of t, see Figure 4.3-left.

(B2-b) If $\mathbf{u} \neq \mathbf{w}$, the argument is the same as in the previous subcase (with $\mathbf{u}$ instead of $\mathbf{s}$), and this subcase is also not possible.

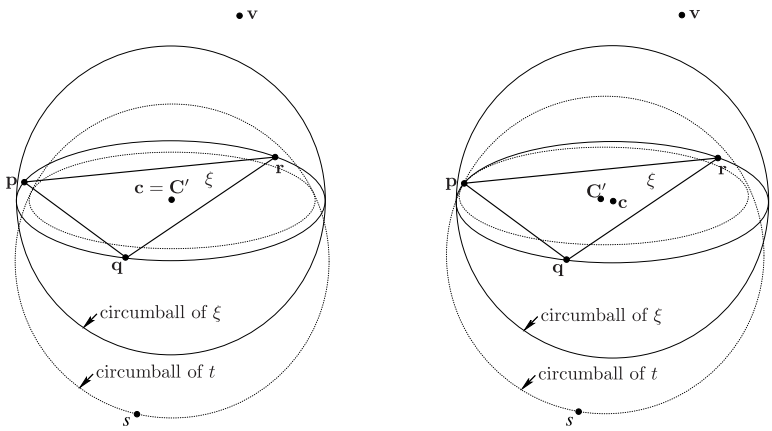

FIG. 4.3. *Illustration to the proof of Theorem 4.1.*

Therefore, for the rest of the proof we assume that $\mathbf{c} \neq \mathbf{C}'$.

For the clarity of presentation we drew Figure 4.1 such that the straight line $\mathbf{cC}'$ is parallel to the view plane. Because of this orientation, the circle at the intersection of the circumballs of t and of ξ is perpendicular to the view plane and is represented in the figure as a straight line segment, $\mathbf{ab}$. We denote the plane defined by this circle as P, and the partition of space induced by P as "left" and "right".

If all three vertices $\mathbf{p}$, $\mathbf{q}$, $\mathbf{r}$ are inside A', they are also inside the circumball of t, and the proof is finished. Otherwise, without loss of generality assuming $\mathbf{p}$ is outside of A', we will show that both $\mathbf{q}$ and $\mathbf{r}$ cannot be outside of A', and therefore at least one of the vertices $\mathbf{q}$ and $\mathbf{r}$ must be inside the circumball of t. Consider the straight line defined by points $\mathbf{p}$ and $\mathbf{w}$ ($\mathbf{p} \neq \mathbf{w}$ by construction, see (A1)-(A3)). Since $\mathbf{w}$ is inside ξ, this line intersects segment $\mathbf{qr}$ at some point, say $\mathbf{d}$. (As special cases we have $\mathbf{d} = \mathbf{q}$ and $\mathbf{w} = \mathbf{d} = \mathbf{q}$.) Now we can state the following facts:

(C1) $\mathbf{p}$ is outside of A' by assumption.
(C2) $\mathbf{p}$ is to the left of or on P, as follows from construction and (C1).
(C3) $\mathbf{v}$ is to the right of P, since it is inside the circumball of t and outside of the circumball of ξ.
(C4) $\mathbf{u}$ is to the right of or on P, as it is on the boundary of t's circumball (being a vertex of t) and outside of the circumball of ξ (by the encroachment rules).
(C5) $\mathbf{w}$ is to the right of P, as follows from construction, (C3) and (C4).
(C6) $\mathbf{d}$ is to the right of P, as follows from construction, (C2) and (C5).
(C7) $\mathbf{w}$ is inside of A' because $\mathbf{v}$ is inside the circumball of t, and $\mathbf{u}$ is inside or on the boundary of the circumball of t.
(C8) $\mathbf{w}$ is inside or on the boundary of B because it is inside or on the boundary of ξ.

Based on these point positions, noting that (C7) and (C8) imply that A' and B intersect, we consider three configurations for the intersection of A' and B:

(D1) A' and B are exactly equal. This case is not possible since $\mathbf{c} \neq \mathbf{C}'$.
(D2) Boundaries of A' and of B touch in a single point $\mathbf{p}$ while A' is inside B. By the argument similar to case (B2), this case is also not possible, see Figure 4.3-right.
(D3) Boundaries of A' and of B intersect in exactly two distinct points, $\mathbf{x}$ and $\mathbf{y}$, and each of these two points may or may not be equal to $\mathbf{p}$. Since part of the

Type of $\mathbf{v}$	Type of $\hat{\mathbf{v}}$					
	Type-0	Type-1	Type-2		Type-3	
	inserted	inserted	rejected	inserted	rejected	inserted
Type-1	$C_{3,1}$	$C_{3,1}$	$C_{2,1}$	$C_{3,1}$	$C_{2,1}$	n/a
Type-2	$C_{3,2}$	$C_{3,2}$	n/a	$C_{3,2}$	$C_{2,2}$	n/a
Type-3	$C_{1,3}$	$C_{1,3}$	n/a	$C_{1,3}$	n/a	$C_{1,3}$

TABLE 5.1

All possible type combinations of a vertex $\mathbf{v}$ *and its parent* $\hat{\mathbf{v}}$ *with the corresponding constants from Theorem 5.3. Vertices of Type-*0 *and Type-*1 *cannot be rejected by the algorithm and therefore the corresponding columns are not shown. For the child vertex* $\mathbf{v}$ *we do not distinguish between inserted and rejected cases because the analysis is the same.*

circle B which is on the right side from P is completely inside A', from (C6) it follows that at least one of the points $\mathbf{q}$ and $\mathbf{r}$ has to be to the right of P, and therefore both inside A' and inside the circumball of t.

□

5. Proof of Termination. In this section we prove the termination of the GDR algorithm. The underlying idea of the analysis is to show that the length of the mesh edges created by the algorithm is bounded from below by a constant which only depends on the input PLC.

First, we need to recall the following lemma.

LEMMA 5.1 (Projection Lemma [21]). *Let f be a subfacet of the Delaunay triangulated facet F. Suppose that f is encroached upon by some vertex $\mathbf{v}$, but $\mathbf{v}$ does not encroach upon any subsegment of F. Then* $\mathrm{proj}_F(\mathbf{v})$ *lies in the facet F, and $\mathbf{v}$ encroaches upon a subfacet of F that contains* $\mathrm{proj}_F(\mathbf{v})$.

Note that this lemma requires that subsegment encroachment is resolved before subfacet encroachment. We satisfied this requirement in the way we formulated the GDR algorithm in Figure 3.2.

Now we need to establish the following two facts that will be used later on.

STATEMENT 5.1. *Given a triangle* $\mathbf{pqr}$ *with circumradius r and point* $\mathbf{u}$ *inside this triangle, the distance from* $\mathbf{u}$ *to the closest vertex of* $\mathbf{pqr}$ *is less than or equal to r.* This Statement follows from the definitions of circumcircle and of circumradius.

LEMMA 5.2. *If $\mathbf{v}$ is a Type-d vertex inserted (or considered for insertion and rejected) inside the selection ball of a d-simplex ξ ($d = 1, 2, 3$) with circumradius equal to r then*

$$R(\mathbf{v}) \geq \delta_d r. \tag{5.1}$$

Proof. By the Delaunay property and the encroachment rules, the circumball of ξ is empty, therefore the closest vertex to $\mathbf{v}$ has to be at a distance greater than or equal to $\delta_d r$. □

THEOREM 5.3 (Point Spacing Theorem). *For a vertex $\mathbf{v}$ of Type-d inserted (or considered for insertion and rejected) by the GDR algorithm either*

$$R(\mathbf{v}) \geq C_{n,d} \cdot R(\hat{\mathbf{v}}), \quad n = 1, 2, \tag{5.2}$$

or

$$R(\mathbf{v}) \geq C_{n,d} \cdot \mathrm{lfs}(\mathbf{v}), \quad n = 3, \tag{5.3}$$

where $C_{n,d}$ are defined separately for each of the cases from Table 5.1 as follows:

$$C_{1,3} = \delta_3 \bar{\rho}, \quad C_{2,d} = \frac{\delta_d}{\sqrt{2}}, \quad C_{3,d} = \frac{\delta_d}{2 - \delta_d}. \tag{5.4}$$

Proof. Below we analyze the corresponding cases.

- The constant $C_{1,3}$ is established by considering the cases when $\mathbf{v}$ is of Type-3. Type-3 vertices are inserted only in the selection balls of skinny tetrahedra, i.e., tetrahedra with

$$\rho = r/l \geq \bar{\rho}, \tag{5.5}$$

where l and r are the shortest edge and the circumradius of a tetrahedron, respectively. By the definitions of the parent and of the insertion radius,

$$R(\hat{\mathbf{v}}) \leq l, \tag{5.6}$$

therefore,

$$\begin{array}{rcll} R(\mathbf{v}) & \geq & \delta_3 r & \text{(from Delaunay property and Lemma 5.2)} \\ & \geq & \delta_3 \bar{\rho} l & \text{(from (5.5))} \\ & \geq & \delta_3 \bar{\rho} R(\hat{\mathbf{v}}) & \text{(from (5.6))}, \end{array}$$

and (5.2) holds with $C_{1,3} = \delta_3 \bar{\rho}$.

The constants $C_{2,2}$ and $C_{2,1}$ are established by considering the encroachment instances when $\mathbf{v}$ and $\hat{\mathbf{v}}$ either both lie in the same PLC facet, or $\hat{\mathbf{v}}$ does not lie on an element of the PLC. These constants are derived separately below.

- To find $C_{2,2}$, let $\mathbf{pqr}$ be the encroached boundary triangle containing $\text{proj}(\hat{\mathbf{v}})$ according to the Projection Lemma 5.1, see Figure 3.1(center). Without loss of generality, let $\mathbf{p}$ be the vertex of the triangle $\mathbf{pqr}$ which is closest to $\text{proj}(\hat{\mathbf{v}})$. Then from Statement 5.1,

$$\|\text{proj}(\hat{\mathbf{v}}) - \mathbf{p}\| \leq r, \tag{5.7}$$

where r is the circumradius of $\mathbf{pqr}$. Since $\hat{\mathbf{v}}$ is inside the 3D circumball of triangle $\mathbf{pqr}$,

$$\|\hat{\mathbf{v}} - \text{proj}(\hat{\mathbf{v}})\| < r. \tag{5.8}$$

From (5.7) and (5.8), considering the right triangle with vertices $\hat{\mathbf{v}}$, $\text{proj}(\hat{\mathbf{v}})$, and $\mathbf{p}$, we obtain that

$$R(\hat{\mathbf{v}}) \leq \|\hat{\mathbf{v}} - \mathbf{p}\| < \sqrt{2} r. \tag{5.9}$$

Therefore

$$\begin{array}{rcll} R(\mathbf{v}) & \geq & \delta_2 r & \text{(from Lemma 5.2)} \\ & > & \delta_2 \frac{R(\hat{\mathbf{v}})}{\sqrt{2}} & \text{(from (5.9))}, \end{array}$$

and (5.2) holds with $C_{2,2} = \frac{\delta_2}{\sqrt{2}}$.

- To find $C_{2,1}$, note that $\hat{\mathbf{v}}$ is a rejected vertex of Type-2 or Type-3 that lies inside the circumball of the encroached segment containing $\mathbf{v}$. Let this segment be $\mathbf{pq}$. Then

$$\begin{aligned} R(\hat{\mathbf{v}}) &\leq \min\{\|\hat{\mathbf{v}}-\mathbf{p}\|, \|\hat{\mathbf{v}}-\mathbf{q}\|\} \\ &< \sqrt{2}r \\ &\leq \sqrt{2}\tfrac{R(\mathbf{v})}{\delta_1} \qquad \text{(from Lemma 5.2).} \end{aligned}$$

  Therefore, (5.2) holds with $C_{2,1} = \frac{\delta_1}{\sqrt{2}}$.

To establish $C_{3,d}$ $(d = 1, 2)$, we consider all the encroachment configurations when the encroaching point $\hat{\mathbf{v}}$ lies on a facet or a segment that is not adjacent upon the facet or the segment containing $\mathbf{v}$. In this case $\hat{\mathbf{v}}$ is an inserted (not rejected) vertex, and $\mathbf{v}$ is an originating vertex. Let ξ be the encroached subsegment or triangular subface with center $\mathbf{c}$ and radius r which contains $\mathbf{v}$. We have two sub-cases, depending on whether or not $\hat{\mathbf{v}}$ is the vertex closest to $\mathbf{v}$, since in a Delaunay triangulation every vertex is always connected with its closest neighbor.

(i) If $\hat{\mathbf{v}}$ is the vertex closest to $\mathbf{v}$ then $R(\mathbf{v}) = \|\mathbf{v}-\hat{\mathbf{v}}\| \geq \mathrm{lfs}(\mathbf{v})$.

(ii) Otherwise, let $\mathbf{w}$ be the vertex closest to $\mathbf{v}$, which, by Delaunay property, has to lie outside the circumball of ξ. Then

$$R(\mathbf{v}) = \|\mathbf{v}-\mathbf{w}\| \geq \delta_d r. \tag{5.10}$$

  Because circumcenter $\mathbf{c}$ of ξ and $\hat{\mathbf{v}}$ lie on non-adjacent features, by Definition 2.2 of the lfs () function,

$$\mathrm{lfs}(\mathbf{c}) \leq \|\mathbf{c}-\hat{\mathbf{v}}\|. \tag{5.11}$$

  Therefore,

$$\begin{aligned} \mathrm{lfs}(\mathbf{v}) &\leq \mathrm{lfs}(\mathbf{c}) + \|\mathbf{v}-\mathbf{c}\| && \text{(from Lemma 2.3)} \\ &\leq \|\mathbf{c}-\hat{\mathbf{v}}\| + \|\mathbf{v}-\mathbf{c}\| && \text{(from (5.11))} \\ &< r + \|\mathbf{v}-\mathbf{c}\| && \text{(because } \hat{\mathbf{v}} \text{ encroaches upon } \xi) \\ &\leq r + (1-\delta_d)r && \text{(since } \mathbf{v} \text{ lies in the selection ball of } \xi) \\ &= (2-\delta_d)r \\ &\leq (2-\delta_d)\tfrac{R(\mathbf{v})}{\delta_d} && \text{(from (5.10)).} \end{aligned}$$

In both sub-cases, $C_{3,d} = \frac{\delta_d}{2-\delta_d}$ $(d = 1, 2)$ satisfy the inequality (5.3).

□

THEOREM 5.4. *The GDR algorithm terminates.*

Proof. Figure 5.1 shows the relationship between the insertion radius $R(\mathbf{v})$ of a vertex $\mathbf{v}$ and the insertion radius $R(\hat{\mathbf{v}})$ of its parent $\hat{\mathbf{v}}$ within a single parent sequence. We can easily see that all loops have a product of multipliers greater than or equal to one. Therefore, for each parent sequence the algorithm does not create edges shorter than $R(\mathbf{v}_m)$ multiplied by a constant, where $\mathbf{v}_m$ is the originating vertex of this sequence. From (5.3) $R(\mathbf{v}_m)$ is $\mathrm{lfs}(\mathbf{v}_m)$ multiplied by a constant. Hence, the algorithm will not create edges shorter than a constant fraction of $\min_{\mathbf{v}\in\Omega} \mathrm{lfs}(\mathbf{v})$ and will eventually terminate because the domain has a finite volume. □

6. Proof of Good Grading. The quantity $D(\mathbf{v})$ is defined as the ratio of $\mathrm{lfs}(\mathbf{v})$ over $R(\mathbf{v})$ [21]:

$$D(\mathbf{v}) = \frac{\mathrm{lfs}(\mathbf{v})}{R(\mathbf{v})}. \tag{6.1}$$

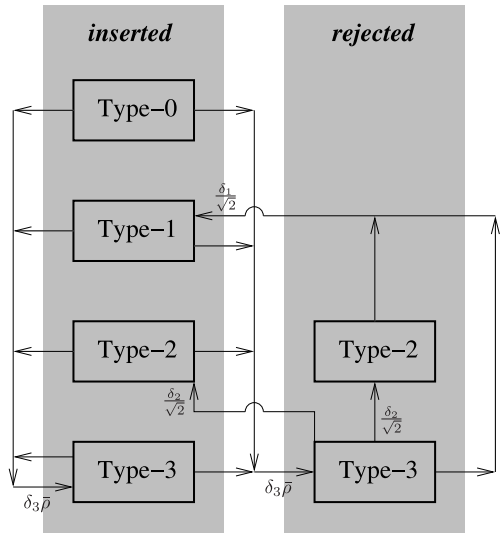

FIG. 5.1. *A diagram illustrating the relationship between the insertion radius $R(\mathbf{v})$ of a vertex $\mathbf{v}$ and the insertion radius $R(\hat{\mathbf{v}})$ of its parent $\hat{\mathbf{v}}$ within a single parent sequence. The head of each arrow points to the box marked with the type of $\mathbf{v}$ and the tail leaves from the box marked with the type of $\hat{\mathbf{v}}$. The arrows are labeled with the minimum value of $R(\mathbf{v})$ in terms of $R(\hat{\mathbf{v}})$, i.e., $R(\mathbf{v}) \geq C \cdot R(\hat{\mathbf{v}})$ where C is the label on the corresponding edge. The arrows pointing to the originating vertices are not shown since they are outside of parent sequences.*

It reflects the density of vertices near $\mathbf{v}$ at the time $\mathbf{v}$ is inserted, weighted by the local feature size. To achieve good mesh grading we need this density to be as small as possible. If the density is bounded from above by a constant, the mesh is said to have a *good grading* property.

LEMMA 6.1. *If $\mathbf{v}$ is a Type-d ($d = 1, 2, 3$) non-originating vertex of the mesh inserted by the GDR algorithm into the selection ball of a d-simplex ξ, and $C_{n,d}$ are the constants specified by Theorem 5.3 for the corresponding cases listed in Table 5.1, then the following inequality holds:*

$$D(\mathbf{v}) \leq B_d + \frac{D(\hat{\mathbf{v}})}{C_{n,d}}, \quad n = 1, 2, \quad \textit{where } B_d = \frac{2 - \delta_d}{\delta_d}. \tag{6.2}$$

Proof. If $\mathbf{c}$ is the circumcenter of ξ and r is the circumradius of ξ, then

$$\begin{aligned}
\|\mathbf{v} - \hat{\mathbf{v}}\| &\leq \|\mathbf{v} - \mathbf{c}\| + \|\mathbf{c} - \hat{\mathbf{v}}\| && \text{(from the triangle inequality)} \\
&\leq (1 - \delta_d) r + \|\mathbf{c} - \hat{\mathbf{v}}\| && \text{(since } \mathbf{v} \text{ is in the selection ball of } \xi) \\
&\leq (1 - \delta_d) r + r && \text{(since } \hat{\mathbf{v}} \text{ is on or inside the circumball of } \xi) \\
&= (2 - \delta_d) r \\
&= \tfrac{2-\delta_d}{\delta_d} \delta_d r \\
&\leq \tfrac{2-\delta_d}{\delta_d} R(\mathbf{v}) && \text{(from Lemma 5.2),}
\end{aligned}$$

or

$$\|\mathbf{v} - \hat{\mathbf{v}}\| \leq B_d \cdot R(\mathbf{v}), \quad B_d = \frac{2 - \delta_d}{\delta_d}. \tag{6.3}$$

Then

$$\begin{aligned}
\operatorname{lfs}(\mathbf{v}) &\leq \operatorname{lfs}(\hat{\mathbf{v}}) + \|\mathbf{v} - \hat{\mathbf{v}}\| && \text{(from Lemma 2.3)} \\
&\leq \operatorname{lfs}(\hat{\mathbf{v}}) + B_d \cdot R(\mathbf{v}) && \text{(from (6.3))} \\
&= D(\hat{\mathbf{v}}) \cdot R(\hat{\mathbf{v}}) + B_d \cdot R(\mathbf{v}) && \text{(from (6.1))} \\
&\leq D(\hat{\mathbf{v}}) \cdot \tfrac{R(\mathbf{v})}{C_{n,d}} + B_d \cdot R(\mathbf{v}) && \text{(from Theorem 5.3).}
\end{aligned}$$

The result follows from the division of both sides by $R(\mathbf{v})$. ∎

THEOREM 6.2. *There exist fixed constants $D_i > 0$, such that, for any vertex $\mathbf{v}$ of Type-i inserted (or considered for insertion and rejected) by the GDR algorithm with $\bar{\rho} > 2$, the following inequality holds:*

$$D(\mathbf{v}) \leq D_i, \quad i = 0, \ldots, 3. \tag{6.4}$$

Therefore, the insertion radius of $\mathbf{v}$ has a lower bound proportional to its local feature size.

Proof. The proof is by induction. The base case covers originating vertices, and the inductive step covers non-originating vertices.

Base case: If $\mathbf{v}$ is an input vertex (Type-0), then all other input vertices must be at a distance of $\operatorname{lfs}(\mathbf{v})$ or greater, and therefore $R(\mathbf{v}) \geq \operatorname{lfs}(\mathbf{v})$. Then $D(\mathbf{v}) = \operatorname{lfs}(\mathbf{v}) / R(\mathbf{v}) \leq 1$, and the theorem holds with

$$D_0 = 1. \tag{6.5}$$

The theorem is also true if $\mathbf{v}$ is a Type-1 or a Type-2 originating vertex (there are no Type-3 originating vertices) since from Theorem 5.3 (case $n = 3$) $D(\mathbf{v}) = \operatorname{lfs}(\mathbf{v}) / R(\mathbf{v}) \leq 1/C_{3,d}$. Therefore, we can choose any D_1 and D_2 which satisfy inequalities (6.6) and (6.7):

$$D_1 \geq \frac{1}{C_{3,1}}, \tag{6.6}$$

$$D_2 \geq \frac{1}{C_{3,2}}. \tag{6.7}$$

Inductive hypothesis: Assume that the theorem is true for any Type-j vertex $\hat{\mathbf{v}}$, i.e., there exists a constant $D_j > 0$ such that

$$D(\hat{\mathbf{v}}) \leq D_j, \quad j = 0, \ldots, 3. \tag{6.8}$$

Inductive step: Now we prove that under the base case and the inductive hypothesis the theorem also holds for any non-originating vertex $\mathbf{v}$ with parent $\hat{\mathbf{v}}$. For the cases $n = 1$ and $n = 2$ from Table 5.1 (remember that for $n = 3$, $\mathbf{v}$ is an originating vertex), we start with (6.2) and apply the inductive hypothesis:

$$D(\mathbf{v}) \leq B_i + \frac{D(\hat{\mathbf{v}})}{C_{n,i}} \leq B_i + \frac{D_j}{C_{n,i}}, \quad n = 1, 2. \tag{6.9}$$

As a result, the inequalities in (6.4) can be satisfied if the constants D are chosen such that the following inequalities hold:

$$B_i + \frac{D_j}{C_{n,i}} \leq D_i, \quad n = 1, 2. \tag{6.10}$$

Using Table 5.1, we expand the inequalities in (6.10) and obtain the following inequalities (6.11)-(6.17):

For $n = 1$:

$$B_3 + \frac{D_0}{C_{1,3}} \leq D_3, \tag{6.11}$$

$$B_3 + \frac{D_1}{C_{1,3}} \leq D_3, \tag{6.12}$$

$$B_3 + \frac{D_2}{C_{1,3}} \leq D_3, \tag{6.13}$$

$$B_3 + \frac{D_3}{C_{1,3}} \leq D_3. \tag{6.14}$$

For $n = 2$:

$$B_2 + \frac{D_3}{C_{2,2}} \leq D_2, \tag{6.15}$$

$$B_1 + \frac{D_2}{C_{2,1}} \leq D_1, \tag{6.16}$$

$$B_1 + \frac{D_3}{C_{2,1}} \leq D_1. \tag{6.17}$$

Now we have a system of recursive inequalities (6.6), (6.7), and (6.11)–(6.17). From (5.4) and (3.1), noting that $\bar{\rho} > 2$, it follows that $C_{1,3}C_{2,1}C_{2,2} > 1$, therefore from (6.12), (6.16), and (6.15) we can derive (6.18)–(6.20):

$$D_1 \geq \frac{C_{1,3}(B_3 + C_{2,2}(B_2 + C_{2,1}B_1))}{C_{1,3}C_{2,1}C_{2,2} - 1}, \tag{6.18}$$

$$D_2 \geq \frac{C_{2,1}(B_1 + C_{1,3}(B_3 + C_{2,2}B_2))}{C_{1,3}C_{2,1}C_{2,2} - 1}, \tag{6.19}$$

$$D_3 \geq \frac{C_{2,2}(B_2 + C_{2,1}(B_1 + C_{1,3}B_3))}{C_{1,3}C_{2,1}C_{2,2} - 1}. \tag{6.20}$$

By direct substitution, we can verify that inequalities (6.18)–(6.20) satisfy the rest of the system, i.e., inequalities (6.6), (6.7), (6.11), (6.13), (6.14), and (6.17). By choosing the smallest admissible values (corresponding to the equalities) for constants D_1, D_2,

and D_3, and by plugging in the expressions for B and C, we conclude that the theorem holds with

$$D_1 = \frac{(2(2-\delta_3) + \sqrt{2}(2-\delta_2)\delta_3 + (2-\delta_1)\delta_2\delta_3)\bar{\rho}}{\delta_1\delta_2\delta_3\bar{\rho} - 2}, \tag{6.21}$$

$$D_2 = \frac{\sqrt{2}(2-\delta_1) + (\sqrt{2}\delta_1(2-\delta_3) + \delta_1(2-\delta_2)\delta_3)\bar{\rho}}{\delta_1\delta_2\delta_3\bar{\rho} - 2}, \tag{6.22}$$

$$D_3 = \frac{\sqrt{2}(2-\delta_2) + (2-\delta_1)\delta_2 + \delta_1\delta_2(2-\delta_3)\bar{\rho}}{\delta_1\delta_2\delta_3\bar{\rho} - 2}. \tag{6.23}$$

□

By examining the expressions for D_1, D_2, and D_3, we also note that $D_1 > D_2 > D_3$ for all admissible values of δ_1, δ_2, δ_3, and $\bar{\rho}$. Indeed, denoting for convenience

$$D_1 - D_2 = \frac{N}{\delta_1\delta_2\delta_3\bar{\rho} - 2},$$

we have

$$\begin{aligned}
N &= (2(2-\delta_3) + \sqrt{2}(2-\delta_2)\delta_3 + (2-\delta_1)\delta_2\delta_3)\bar{\rho} \\
&\quad -\sqrt{2}(2-\delta_1) - (\sqrt{2}\delta_1(2-\delta_3) + \delta_1(2-\delta_2)\delta_3)\bar{\rho} \\
&= 4\bar{\rho} + 2(\sqrt{2}-1)\delta_3\bar{\rho} + (2-\sqrt{2})\delta_2\delta_3\bar{\rho} \\
&\quad -2\sqrt{2} - \delta_1\left((2\sqrt{2} + (2-\sqrt{2})\delta_3)\bar{\rho} - \sqrt{2}\right) \\
&\geq 4\bar{\rho} + 2(\sqrt{2}-1)\delta_3\bar{\rho} + 2(2-\sqrt{2}) \\
&\quad -2\sqrt{2} - \left((2\sqrt{2} + (2-\sqrt{2})\delta_3)\bar{\rho} - \sqrt{2}\right) && \text{(using (3.1))} \\
&> 0 && \text{(since } \bar{\rho} > 2\text{)},
\end{aligned}$$

and therefore $D_1 > D_2$. Similarly it can be shown that $D_2 > D_3$. In other words, grading is always better (in the worst case analysis) in the interior of the domain that on the boundaries.

In Table 6.1 we list the values of D_1, D_2, and D_3 corresponding to a few values of parameters δ_1, δ_2, δ_3, and $\bar{\rho}$. As expected, grading becomes worse with the increase in the size of the selection balls. This follows from the fact that larger selection balls allow for shorter mesh edges. Also, an increase in size of one type of selection balls leads to an increase of all grading constants, which is due to the involvement of parent points in the grading analysis.

7. Summary. We developed a novel generalized three-dimensional guaranteed quality Delaunay refinement algorithm and proved its termination, as well as fidelity and good grading of the resulting mesh. The presented algorithm and analysis extend the previous approaches by introducing a sequence of three types of insertion regions, corresponding to the points inserted on the domain features of each dimensionality. The sizes of the regions can be chosen for each instantiation of the algorithm, based on its requirements with respect to the grading–flexibility tradeoff.

A parallelization [4] of this generalized algorithm immediately implies parallelizations of all conforming instantiations. Subject to further analysis, the proposed selection balls could offer the flexibility and a unified theoretical framework to insert points in a variety of positions dictated by various Delaunay-based mesh optimizations techniques such as the following:

$\bar{\rho}$	δ_1	δ_2	δ_3	D_1	D_2	D_3
2.1	1.0	1.0	1.0	92.70	64.84	45.14
2.5	1.0	1.0	1.0	22.07	14.90	9.83
3.0	1.0	1.0	1.0	13.24	8.66	5.41
3.0	1.0	1.0	0.9	18.74	12.54	8.16
3.0	1.0	1.0	0.8	32.49	22.26	15.04
3.0	1.0	1.0	0.7	128.70	90.30	63.14
3.0	1.0	0.9	1.0	19.10	12.80	7.37
3.0	1.0	0.9	0.9	30.77	21.05	12.62
3.0	1.0	0.9	0.8	81.83	57.16	35.60
3.0	1.0	0.8	1.0	33.73	23.14	12.24
3.0	1.0	0.8	0.9	83.39	58.26	32.11
3.0	1.0	0.7	1.0	136.15	95.57	46.38
3.0	0.9	1.0	1.0	19.35	11.53	7.45
3.0	0.9	1.0	0.9	31.14	19.04	12.75
3.0	0.9	1.0	0.8	82.71	51.86	35.96
3.0	0.9	0.9	1.0	31.71	19.40	11.57
3.0	0.9	0.9	0.9	72.05	45.07	27.91
3.0	0.9	0.8	1.0	85.82	53.84	29.61
3.0	0.8	1.0	1.0	34.61	18.73	12.54
3.0	0.8	1.0	0.9	85.36	47.44	32.84
3.0	0.8	0.9	1.0	86.92	48.32	29.97
3.0	0.7	1.0	1.0	141.43	69.08	48.14
4.0	1.0	1.0	1.0	8.83	5.54	3.21
8.0	1.0	1.0	1.0	5.89	3.45	1.74
16.0	1.0	1.0	1.0	5.04	2.86	1.32
16.1	0.5	0.5	0.5	5713.13	2018.84	712.71
32.0	1.0	1.0	1.0	4.71	2.62	1.15
32.0	0.5	0.5	0.5	70.97	24.03	7.44

TABLE 6.1

Several sample values of algorithm parameters and the corresponding grading constants.

- avoiding slivers by inserting points in the neighborhoods of circumcenters [7, 11],
- reducing the final mesh size by picking specific "off-center" positions [23],
- creating a hierarchy of nested meshes [18] by choosing points on the existing mesh edges [17].

Acknowledgments. We thank the anonymous reviewers for detailed comments which helped us improve the manuscript. We also thank Panagiotis Foteinos for insightful discussions on the proof of good grading.

REFERENCES

[1] MARSHALL WAYNE BERN, DAVID EPPSTEIN, AND JOHN RUSSELL GILBERT, *Provably good mesh generation*, J. Computer & Systems Sciences, 48 (1994), pp. 384–409. Special issue for 31st FOCS.

[2] ADRIAN BOWYER, *Computing Dirichlet tesselations*, Computer Journal, 24 (1981), pp. 162–166.

[3] ANDREY N. CHERNIKOV AND NIKOS P. CHRISOCHOIDES, *Three-dimensional semi-generalized point placement method for Delaunay mesh refinement*, in Proceedings of the 16th Inter-

national Meshing Roundtable, Seattle, WA, Oct. 2007, Springer, pp. 25–44.

[4] ———, *Three-dimensional Delaunay refinement for multi-core processors*, in Proceedings of the 22nd Annual International Conference on Supercomputing, Island of Kos, Greece, 2008, ACM Press, pp. 214–224.

[5] ———, *Generalized two-dimensional Delaunay mesh refinement*, SIAM Journal on Scientific Computing, 31 (2009), pp. 3387–3403.

[6] L. Paul Chew, *Guaranteed-quality triangular meshes*, Tech. Report TR89983, Cornell University, Computer Science Department, 1989.

[7] ———, *Guaranteed-quality Delaunay meshing in 3D*, in Proceedings of the 13th ACM Symposium on Computational Geometry, Nice, France, 1997, pp. 391–393.

[8] Panagiotis A. Foteinos, Andrey N. Chernikov, and Nikos P. Chrisochoides, *Fully generalized two-dimensional constrained Delaunay mesh refinement*, SIAM Journal on Scientific Computing, 32 (2010), pp. 2659–2686.

[9] Paul-Louis George and Houman Borouchaki, *Delaunay Triangulation and Meshing. Application to Finite Elements*, HERMES, 1998.

[10] Xiang-Yang Li, *Generating well-shaped d-dimensional Delaunay meshes*, Theoretical Computer Science, 296 (2003), pp. 145–165.

[11] Xiang-Yang Li and Shang-Hua Teng, *Generating well-shaped Delaunay meshes in 3D*, in Proceedings of the 12th annual ACM-SIAM symposium on Discrete algorithms, Washington, D.C., 2001, pp. 28–37.

[12] Gary L. Miller, Steven E. Pav, and Noel Walkington, *When and why Delaunay refinement algorithms work*, Int. J. Comput. Geometry Appl., 15 (2005), pp. 25–54.

[13] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, and Noel Walkington, *A Delaunay based numerical method for three dimensions: Generation, formulation, and partition*, in Proceedings of the 27th Annual ACM Symposium on Theory of Computing, Las Vegas, NV, May 1995, pp. 683–692.

[14] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, Noel Walkington, and Han Wang, *Control volume meshes using sphere packing: Generation, refinement and coarsening*, in Proceedings of the 5th International Meshing Roundtable, Pittsburgh, PA, Oct. 1996, pp. 47–61.

[15] Scott A. Mitchell and Stephen A. Vavasis, *Quality mesh generation in higher dimensions*, SIAM Journal for Computing, 29 (2000), pp. 1334–1370.

[16] Alexander Rand and Noel Walkington, *Collars and intestines: Practical conforming Delaunay refinement*, in Proceedings of the 18th International Meshing Roundtable, 2009, pp. 481–497.

[17] Maria-Cecilia Rivara, *A study on Delaunay terminal edge method*, in Proceedings of the 15th International Meshing Roundtable, Birmingham, AL, Sept. 2006, Springer, pp. 543–562.

[18] Ulrich Rüde, *Mathematical and Computational Techniques for Multilevel Adaptive Methods*, Frontiers in Applied Mathematics, SIAM, 1993.

[19] Jim Ruppert, *A Delaunay refinement algorithm for quality 2-dimensional mesh generation*, Journal of Algorithms, 18(3) (1995), pp. 548–585.

[20] Jonathan Richard Shewchuk, *Delaunay Refinement Mesh Generation*, PhD thesis, Carnegie Mellon University, 1997.

[21] ———, *Tetrahedral mesh generation by Delaunay refinement*, in Proceedings of the 14th ACM Symposium on Computational Geometry, Minneapolis, MN, 1998, pp. 86–95.

[22] Hang Si, *An analysis of Shewchuk's Delaunay refinement algorithm*, in Proceedings of the 18th International Meshing Roundtable, Salt Lake City, UT, October 2009, Springer, pp. 499–518.

[23] Alper Üngör, *Off-centers: A new type of Steiner points for computing size-optimal guaranteed-quality Delaunay triangulations*, in Proceedings of LATIN, Buenos Aires, Argentina, Apr. 2004, pp. 152–161.

[24] David F. Watson, *Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes*, Computer Journal, 24 (1981), pp. 167–172.