Boundary Refinement in Delaunay Mesh Generation Using Arbitrarily Ordered Vertex Insertion

Démian M. Nave* Nikos Chrisochoides†

Abstract

In general, guaranteed-quality Delaunay meshing algorithms are difficult to parallelize because they require strictly ordered updates to the mesh boundary. We show that, by replacing the Delaunay cavity in the Bowyer-Watson algorithm with what we call the *circumball intersection set*, updates to the mesh can occur in any order, especially at the mesh boundary.

To demonstrate this new idea, we describe a 2D constrained Delaunay meshing algorithm that does not enforce strict ordering of vertex insertions near the mesh boundary. We prove that the sequential version of this algorithm generates a mesh in which the circumradius to shortest edge ratio of every triangle is $\sqrt{2}$ or greater, as long as every angle interior to the polygonal input domain is at least 90^o. We briefly touch upon the parallel version of this algorithm, but we relegate a more complete discussion (with extension to 3D) to a forthcoming paper.

1 Introduction

In our previous work [9], we developed and proved correct a guaranteed–quality parallel 3D Delaunay refinement algorithm for polyhedral domains without obtuse boundary angles. The proof of correctness requires a preprocessing step which generates a potentially dense surface mesh on the boundary $\partial\Omega$ of an input domain Ω containing no edge longer than the minimum local feature size [10] of Ω. Preprocessing $\partial\Omega$ in this way is required by the proof of correctness to prevent concurrent vertex insertions from violating the invariants of the corresponding sequential algorithm (Section 2).

The resulting tetrahedral mesh may be unnecessarily dense, so it would be advantageous to avoid this expensive preprocessing step altogether. Furthermore, it is not obvious how to extend our previous algorithm to more complicated problems, such as meshing domains with sharp angles [12, 4, 3] and generating meshes without slivers [8, 2].

*Pittsburgh Supercomputing Center, Carnegie Mellon University, Pittsburgh, PA, dnave@psc.edu

†Department of Computer Science, College of William & Mary, Williamsburg, VA, nikos@cs.wm.edu

In this paper, we show how to solve the former problem by augmenting the Delaunay cavity of the Bowyer-Watson algorithm [1, 14] with what we call the *circumball intersection set* (the latter problem is relegated to future work). When combined with a suitably modified constrained planar Delaunay meshing algorithm (Section 3), the circumball intersection set allows arbitrarily ordered point insertions without jeopardizing the termination and quality guarantees of the meshing algorithm.

When a triangle f in the mesh is refined by its circumcenter v, we search for the set $\mathcal{C}$ of triangles whose circumscribing 2-balls (*circumballs*) intersect the circumball of f. We show that, if the set $\mathcal{C}$ contains one or more *encroached* subsegments whose minimum-radius (*diametral*) circumballs enclose v, then v should be discarded and the midpoint of one of the encroached subsegments should be added to the mesh instead.

In other words, we only need to examine a local region of the mesh close to v (but potentially larger than the Delaunay cavity) to determine if adding v to the mesh would result in new short edges. We can therefore avoid the usual requirement of most Delaunay meshing algorithms [10, 11, 13] that subsegments be unencroached before adding new interior vertices into the mesh.

We show that a simple 2D algorithm with this modification (Section 3) generates a constrained Delaunay triangulation (CDT) [5] of the input domain in which every triangle t has a circumradius to shortest edge ratio (*ratio(t)*) no greater than $\sqrt{2}$. Further, as a result of allowing poorly-shaped triangles to be refined in any order, this new algorithm is straightforward to parallelize using some of the proof machinery from our previous work [9] (Section 4).

2 Strict Ordering, Violating Invariants

Sequential Delaunay meshing algorithms guarantee quality by enforcing a strict ordering of vertex insertions on or near the domain boundary. It is this strict ordering near the boundary that complicates parallel meshing algorithms—far away from the boundary, the meshing process requires no more than maintaining the Delaunay property of the mesh [6].

In particular, most existing algorithms require that subsegments be unencroached before new interior vertices can be added to the mesh. Not adhering to this

order—when concurrently inserting vertices into a distributed mesh, for example—can lead to a violation of the primary invariant of these algorithms: that no edge in the resulting mesh is shorter than the *local feature size*, $\texttt{lfs}_{p\in\Omega}(p)$ [10].

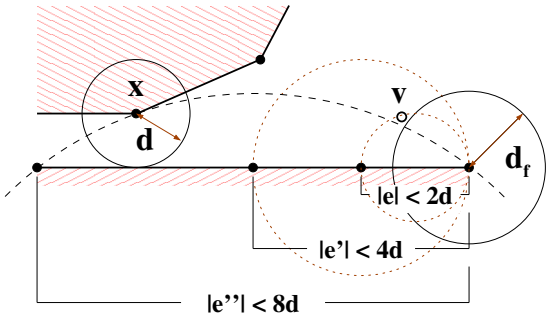

Figure 1: *A typical invariant is violated.* $d = \min_{p\in\Omega} \texttt{lfs}(p)$ and $d_f > d\sqrt{2}$. e'' is encroached upon by input vertex x, but this is not resolved before adding v to the mesh. Consequently, v encroaches upon the subsegment e with $|e| < 2 \cdot d$, violating a typical invariant that encroached subsegments have length greater than $2 \cdot d$.

Consider a typical 2D algorithm [10] that enforces two invariants: (i) refined subsegments have length at least $2 \cdot d$, and (ii) refined triangles have a radius greater than $d\sqrt{2}$, where $d = \min_{p\in\Omega} \texttt{lfs}(p)$. Figure 1 depicts what happens if vertices inserted near the boundary are not properly ordered. A new vertex v is added to the mesh to refine a poorly-shaped triangle, but because v is too close to the subsegment e'', subsegment e with $|e| < 2 \cdot d$ is created and is encroached upon by v, violating the first invariant described above.

In many respects, this is an implementation issue, since we could enforce this precondition on vertex insertions with an oracle that, for each new vertex v, answers the query "does v encroach upon an already encroached subsegment e?" If such a subsegment e does exist, then e would be refined instead of inserting v into the mesh. The Delaunay cavity can serve this purpose efficiently when all subsegments are unencroached before a new vertex is inserted. It is not difficult to show that subsegments encroached upon by v appear on the boundary of the Delaunay cavity for v, so the search for encroached subsegments is a local operation.

However, such an oracle is probably difficult to implement in general, and would almost certainly be inefficient for parallel meshing. Instead, we can take advantage of the fact that most Delaunay meshing algorithms insert new vertices inside the circumball of some poorly-shaped triangle [1]. Because of this property, it is possible to answer the query described above by searching the mesh in a region near a to-be-refined triangle. In the following section, we describe our approach within the context of a simple guaranteed-quality 2D Delaunay meshing algorithm.

[1] One notable exception is the sink insertion method proposed by Edelsbrunner and Guoy [7].

3 An Algorithm for 2D Meshing Using Arbitrarily Ordered Vertex Insertion

Let Ω be a polygonal input domain with boundary $\partial\Omega$, possibly with interior, non-degenerate segment-bounded holes. Assume that no interior angle in $\partial\Omega$ is less than 90^o, and let d be the minimum distance (through Ω) between any two non-incident vertices or segments in $\partial\Omega$. The output of Algorithm 1 is a mesh $\mathcal{M}(\mathcal{K}, \mathcal{D})$ comprising two simplicial complexes: $\mathcal{K}$, containing vertices and subsegments, and $\mathcal{D}$, containing vertices, edges, and triangles. Upon completion, each subsegment $e \in \mathcal{K}$ has length at least d, and each triangle $f \in \mathcal{D}$ has a circumradius to shortest edge ratio (*ratio(f)*) no greater than $\sqrt{2}$.

In the algorithm that follows, we use $\bigcirc s$ to mean the minimum-radius open 2-ball (*circumball*) circumscribing the vertices of s. For example, if e is a subsegment, then $\bigcirc e$ is the open 2-ball having e as a diameter.

Algorithm 1 Create a 2D CDT $\mathcal{M}(\mathcal{K}, \mathcal{D})$. $\mathcal{M}$ consists solely of triangles f with $ratio(f) \leq \sqrt{2}$. The triangles refined by *Main Loop* can be chosen in any order.

SeqArbitrarilyOrdered2D(Ω)

Input: Polygonal domain $\Omega \subset \Re^2$ with no interior angle less than 90^o.

Output: $\mathcal{M}(\mathcal{K}, \mathcal{D})$, a constrained Delaunay mesh such that $\forall f \in \mathcal{D}, ratio(f) \leq \sqrt{2}$

Initialize $\mathcal{M}$:

Let $\mathcal{K}$ and $\mathcal{D}$ be the boundary and interior triangulations of the CDT of Ω.

Main loop:

while $\exists f \in \mathcal{D}$ and $ratio(f) > \sqrt{2}$ **do**
  Refine(f)
end while

Refine(f):

$c \leftarrow circumcenter(f)$
$\mathcal{C} = \{g \in \mathcal{D} \ni: \bigcirc g \cap \bigcirc f \neq \emptyset\}$
if $\exists e \in (\mathcal{C} \cap \mathcal{K}) \ni: c \in \bigcirc e$ and $e \cap \bigcirc f$ not contain an endpoint of e **then**
  Insert the midpoint of e into $\mathcal{K}$ and $\mathcal{D}$
else
  Insert c into $\mathcal{D}$
end if

It is important to note that no particular order is imposed on the triangles chosen for refinement. Moreover,

all decisions regarding mesh boundary updates are confined to a region of the mesh near each newly inserted vertex. These properties are due to the *circumball intersection set*, $\mathcal{C}$, an edge-connected set of triangles whose circumballs intersect the circumball of a refined triangle f. $\mathcal{C}$ is a superset of the *Delaunay cavity* of the circumcenter c of f, the edge-connected set of triangles whose circumballs enclose c. Like the Delaunay cavity of c, $\mathcal{C}$ can be computed by a graph search starting from the refined triangle f [2], the implementation of which is only a minor change to an existing Bowyer-Watson implementation.

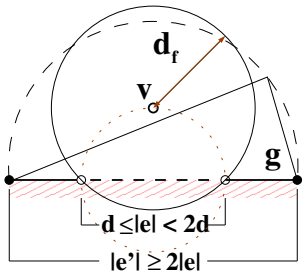

Figure 2: *Case (1.i).* v cannot encroach upon e without the circumball of radius $d_f > d\sqrt{2}$ intersecting both e' and the circumball of the triangle containing e'. The algorithm does not insert v in this case, so the problem depicted in Figure 1 cannot occur.

The following lemma establishes the correctness of Algorithm 1:

Lemma 1 *Let* $d = \min_{p\in\Omega} \texttt{lfs}(p)$. *The following invariants hold after each iteration through* Main Loop*:*

1. *Each refined subsegment has length at least* $2 \cdot d$.
2. *Each refined subfacet has circumradius greater than* $d\sqrt{2}$.

Proof. [Sketch] We show by contradiction that each invariant holds:

Invariant 1: *each refined subsegment* e *has length* $|e| \geq 2 \cdot d$.

Assume that the first failure of the algorithm is a subsegment e with $|e| < 2 \cdot d$. The vertex v causing the refinement of e cannot be in $\mathcal{K}$, since all vertices in $\mathcal{K}$ are at least distance d (through the mesh) from the midpoint of e, and M is constrained Delaunay. Therefore, v must be a triangle circumcenter that either (i) was successfully added to the mesh in a previous iteration, or (ii) was not added to the mesh in the current iteration because it encroached upon e. In either case, the circumradius of the refined triangle f was greater than $d\sqrt{2}$, otherwise refining f would have been the first failure of the invariants.

Case (i): v is an existing interior mesh vertex (this failure is depicted in Figure 1):

At the time v was added to the mesh to refine f, there was a triangle g that contained the subsegment $e' \supset e$, and g did not appear in the circumball intersection set $\mathcal{C}$ of v (otherwise, the algorithm would not have added v to the mesh). This implies that either the path through the mesh from f to g contains triangles not in $\mathcal{C}$, or the circumball of the triangle g containing e' did not intersect $\bigcirc f$.

The first case does not affect correctness since f and g are constrained Delaunay and e is the first failure, so assume that $\bigcirc f \cap \bigcirc g = \emptyset$. By assumption, $radius(f) > d\sqrt{2}$ and v is within d of e', so necessarily $\bigcirc f \cap \bigcirc g \neq \emptyset$ (Figure 2). However, this means that g must have appeared in $\mathcal{C}$, a contradiction. Therefore, this case cannot occur.

Case (ii): v is the circumcenter of triangle f that was not added to the mesh in the current iteration because it encroached upon e:

Because $\mathcal{D}$ is constrained Delaunay, v would be too far away to encroach upon e unless $\bigcirc f$ enclosed a vertex u of e. However, the algorithm checks for this case and would not have refined e, a contradiction.

Since both cases result in a contradiction, **Invariant 1** must hold after each iteration of *Main Loop*. This ensures that no subsegment shorter than d is ever introduced into the mesh.

Invariant 2: *each refined subfacet* f *has* $radius(f) > d\sqrt{2}$.

Assume the first failure is the refinement of a triangle f with $radius(f) \leq d\sqrt{2}$. Because this is the first failure, every subsegment has length at least d (*Invariant 1*), and every edge resulting from inserting a triangle circumcenter has length greater than $d\sqrt{2}$. Therefore, the shortest edge of f must have length at least d, so $ratio(f) \leq \sqrt{2}$. But, f would have been refined only if $ratio(f) > \sqrt{2}$, a contradiction. Therefore, this case cannot occur. □

This proof shows that the invariants hold throughout the algorithm, which allows us to prove the following properties of the resulting mesh:

Theorem 1 *Algorithm* SeqArbitrarilyOrdered2D *terminates, and the resulting mesh* $\mathcal{M}$ *has the following properties:*

1. *The length of every edge in* $\mathcal{M}$ *is at least* d.
2. *For every triangle* $f \in \mathcal{M}$, $ratio(f) \leq \sqrt{2}$.

[2] Unlike the Delaunay cavity of c, edges in the boundary of $\mathcal{C}$ are *not* necessarily visible to c.

3. The minimum triangle angle is 20.7^o.

Proof. Note that exactly one new vertex is added to the mesh each time through *Main Loop*. From Lemma 1, we know that no two vertices of the mesh are ever closer than d, therefore the algorithm must terminate since only finitely many edges of length d can be placed within Ω (a finite area). Termination is enough to guarantee the bound on triangle circumradius to shortest edge ratio, and the bound on the minimum angle follows. □

4 A Brief Note on Parallelization

The following observation is a direct consequence of Lemma 1:

Corollary 2 ([9]) *The correctness of* SeqArbitrarily-Ordered2D *does not depend on the order in which any two triangles are refined in* Main Loop.

This is in fact the only statement in our previous series of proofs [9] whose correctness is directly dependent upon the behavior of the sequential meshing algorithm. We therefore make the following conjecture that *ParArbitrarilyOrdered2D*, a parallel version of algorithm Algorithm 1 is correct:

Conjecture 1 *Algorithm* ParArbitrarilyOrdered2D *terminates, and the distributed mesh has the same properties as those guaranteed by* SeqArbitrarilyOrdered2D.

Unlike in our previous work, this conjecture does not require a potentially expensive preprocessing step, due to the definition of the circumball intersection set and the proof of Lemma 1.

5 Conclusions and Future Work

Our primary contribution in this paper is the introduction of the circumball intersection set, which permits arbitrarily ordered vertex insertions, and, consequently, yields a naturally parallelizable sequential guaranteed-quality 2D meshing algorithm (Algorithm 1). We believe that the circumball intersection set will allow us to more easily parallelize a wide range of other 3D Delaunay-based meshing algorithms, in particular those that prevent slivers and those that can handle sharp boundary angles.

6 Acknowledgements

Part of this work was supported by NSF grants #ACI-0312980 and #EIA-0203974, and also by NIH grant #P41 RR06009.

References

[1] A. Bowyer. Computing Dirichlet tessellations. *The Computer Journal*, 24(2):162–166, 1981.

[2] S.-W. Cheng and T. K. Dey. Quality meshing with weighted delaunay refinement. In 13^{th} *ACM-SIAM Symposium on Discrete Algorithms*, pages 137–146, 2002.

[3] S.-W. Cheng, T. K. Dey, E. A. Ramos, and T. Ray. Quality meshing for polyhedra with small angles. In 20^{th} *Annual Symposium on Computational Geometry*, 2004.

[4] S.-W. Cheng and S.-H. Poon. Graded conforming delaunay tetrahedralization with bounded radius-edge ratio. In 14^{th} *ACM-SIAM Symposium on Discrete Algorithms*, pages 295–304. Society for Industrial and Applied Mathematics, 2003.

[5] L. P. Chew. Constrained delaunay triangulations. *Algorithmica*, 4:97–108, 1989.

[6] N. Chrisochoides and D. Nave. Parallel Delaunay mesh generation kernel. *In special issue of International Journal for Numerical Methods in Engineering*, 58(2):161–176, 2003.

[7] H. Edelsbrunner and D. Guoy. Sink insertion for mesh improvement. In 17^{th} *Annual Symposium on Computational Geometry*, pages 115–123, 2001.

[8] X.-Y. Li and S.-H. Teng. Generating Well-Shaped delaunay meshes in 3D. In 12^{th} *ACM-SIAM Symposium on Discrete Algorithms*, pages 28–37, 2001.

[9] D. Nave, N. Chrisochoides, and L. P. Chew. Guaranteed–quality parallel Delaunay refinement for restricted polyhedral domains. *Journal of Computational Geometry: Theory and Applications*, 28(2-3):195–215, June 2004. Also appears in 18^{th} Annual Symposium on Computational Geometry.

[10] J. Ruppert. A delaunay refinement algorithm for quality 2–dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.

[11] J. R. Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, Carnegie Mellon University, School of Computer Science, May 1997. Available as Technical Report CMU–CS–97–137.

[12] J. R. Shewchuk. Mesh generation for domains with small angles. In *Sixteenth Annual Symposium on Computational Geometry*, pages 111–112. ACM, 2000.

[13] D. Spielman, S.-H. Teng, , and A. Üngör. Delaunay refinement: algorithms and analyses. In 11^{th} *International Meshing Roundtable*, pages 205–217, 2002.

[14] D. F. Watson. Computing the n–dimensional Delaunay tessellation with application to Voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.