

Boundary Refinement in Delaunay Mesh Generation Using Arbitrarily Ordered Vertex Insertion

Extended Abstract

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° . 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 [8], 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 dense surface mesh on the boundary $\partial\Omega$ of an input domain Ω . This initial surface mesh consists of boundary triangles and subsegments whose circumradii and lengths are bounded by a length proportional to the minimum local feature size on 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).

However, if there is a very large difference between the minimum and maximum local feature size on Ω , the initial surface mesh is significantly over-refined, resulting in an overly-dense (though not necessarily uniformly dense) initial tetrahedral mesh. Although this mesh is

destined for parallel refinement, it would clearly be advantageous to avoid an overly dense mesh to begin with. 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 [7, 2].

In this paper, we achieve these results by replacing the Delaunay cavity search of the Bowyer-Watson algorithm [1, 14] with a search for a superset \mathcal{C} of the Delaunay cavity that contains triangles whose circumscribing 2-balls (*circumballs*) intersect the circumball of the corresponding triangle f . We show that, if the set \mathcal{C} contains certain *encroached* subsegments whose minimum-radius (i.e. diametral) circumballs enclose v , then v should be discarded and the midpoint of some encroached subsegment 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 short edges being introduced into the mesh. We can therefore avoid the usual requirement of most existing Delaunay meshing algorithms [9, 10, 13] that the subsegments (and subfacets in 3D [11, 8]) 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 no triangle has a circumradius to shortest edge ratio 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 [8] (Section 4).

2 Strict Ordering, Violating Invariants

Sequential Delaunay meshing algorithms generally guarantee quality by enforcing a strict ordering of vertex insertions near or on 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 preventing data structure inconsistency and maintaining the Delaunay property of the mesh [6].

*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 particular, most existing algorithms require that subsegments and subfacets in the surface mesh be unencroached before new interior vertices can be added to the mesh. Not adhering to this order can cause a violation of the primary invariant of these algorithms: that no edge in the resulting mesh is shorter than some characteristic length over the domain (e.g. the *local feature size*, $\text{lfs}_{p \in \Omega}(p)$ [9]).

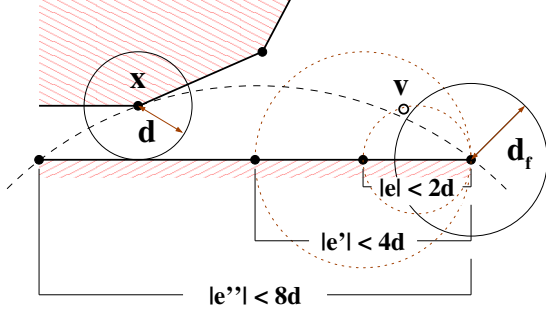


Figure 1: A typical invariant is violated. $d = \min_{p \in \Omega} \text{lfs}(p)$ and $d_f > d\sqrt{2}$. e'' is encroached upon by input vertex x , but this is not resolved before adding v to the mesh. Consequently, v encroaches upon the subsegment e with $|e| < 2 \cdot d$, violating a typical invariant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

Consider a typical 2D algorithm [9] 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 $d = \min_{p \in \Omega} \text{lfs}(p)$. 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 v would be discarded and e would be refined instead.

The Delaunay cavity can serve this purpose when subsegments are unencroached before vertices are inserted. It is not difficult to show that subsegments encroached upon by v appear on the boundary of the Delaunay cavity for v , and thus could be checked before adding v to the mesh.

However, in the case of arbitrarily ordered vertex insertions, such an oracle is probably difficult to implement and would almost certainly be inefficient, particularly for parallel meshing. We have therefore devised a way to answer this query by searching the mesh in a region near a to-be-added vertex v . In the following

section, we describe our approach within the context of a simple guaranteed-quality 2D Delaunay meshing algorithm.

3 A Naturally Parallelizable 2D Meshing Algorithm

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° , 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 ($\text{ratio}(f)$) no greater than $\sqrt{2}$.

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

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

SeqArbitrarilyOrdered2D(Ω)

Input: Polygonal domain $\Omega \subset \mathbb{R}^2$ with no interior angle less than 90° .

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

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  $\text{ratio}(f) > \sqrt{2}$  do
  Refine( $f$ )
end while

```

Refine(f):

```

 $c \leftarrow \text{circumcenter}(f)$ 
 $\mathcal{C} = \{g \in \mathcal{D} \ni |\circ g \cap \circ f| > 0\}$ 
if  $\exists e = e(u, v) \in (\mathcal{C} \cap \mathcal{K}) \ni c \in \circ e \wedge u, v \notin \circ f$  then
  Insert the midpoint of  $e$  into  $\mathcal{K}$  and  $\mathcal{D}$ 
else
  Insert  $c$  into  $\mathcal{D}$ 
end if

```

Note that no particular order is imposed on the triangles chosen for refinement. Further, all decisions regarding mesh boundary updates are confined to a region of the mesh near each newly inserted vertex. The key to these properties lies in the set of triangles \mathcal{C} whose

circumballs intersect the circumball of a to-be-refined triangle f . If c is the circumcenter of f , then \mathcal{C} is a superset of the set of triangles (the *Delaunay cavity*) whose circumballs enclose c .¹ Note that the computation of \mathcal{C} replaces the usual Delaunay cavity computation, and that we have a secondary step to actually insert c into the mesh.²

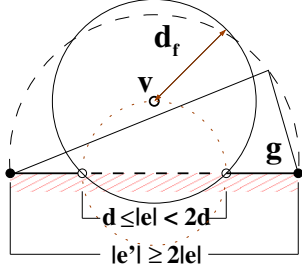


Figure 2: *Case (1.i)*. v cannot encroach upon e without the circumball of radius $d_f > d\sqrt{2}$ intersecting both e' and the circumball of the triangle containing e' . The algorithm does not insert v in this case, so the problem depicted in Figure 1 cannot occur.

The following lemma establishes the correctness of Algorithm 1:

Lemma 1 *Let $d = \min_{p \in \Omega} \text{lf}s(p)$. Then 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 d from the midpoint of e . Therefore, v is 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):

¹Unlike the Delaunay cavity of c , the boundary of \mathcal{C} is *not* necessarily pointwise-convex with respect to c .

²The Delaunay cavity is a subset of \mathcal{C} , though, so it need not be computed separately as suggested here.

At the time v was added to the mesh, there was a triangle g that contained the subsegment $e' \supset e$ but did not appear in the set \mathcal{C} of triangles whose circumballs intersect $\circ f$ (otherwise, v would not have been added to the mesh). This implies that the circumball of the triangle g containing e' did not intersect $\circ f$. However, $\text{radius}(f) > d\sqrt{2}$ and v is within d of e' (it is enclosed by $\circ e$ by assumption), which implies that $\circ f$ intersects $\circ g$ (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 \mathcal{D} is constrained Delaunay, v would be too far away to encroach upon e unless $\circ f$ enclosed a vertex u of e . However, the algorithm checks for this and would not have refined e , so this is 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 $\text{radius}(f) > d\sqrt{2}$.*

Assume the first failure is the refinement of a triangle f with $\text{radius}(f) \leq d\sqrt{2}$. 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 $\text{ratio}(f) \leq \sqrt{2}$. But, f would have been refined only if $\text{ratio}(f) > \sqrt{2}$, a contradiction. Therefore, this case cannot occur. \square

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 $f \in \mathcal{M}$, $\text{ratio}(f) \leq \sqrt{2}$.*
3. *The minimum triangle angle is 20.7° .*

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. \square

4 A Brief Note on Parallelization

The following observation is a direct consequence of Lemma 1:

Corollary 2 ([8]) *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 [8] 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). Moreover, we believe that the circumball intersection set will allow us to 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.

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 *13th 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 *20th Annual Symposium on Computational Geometry*, 2004.
- [4] S.-W. Cheng and S.-H. Poon. Graded conforming delaunay tetrahedralization with bounded radius-edge ratio. In *14th 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] X.-Y. Li and S.-H. Teng. Generating Well-Shaped delaunay meshes in 3D. In *12th ACM-SIAM Symposium on Discrete Algorithms*, pages 28–37, 2001.
- [8] 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 18th Annual Symposium on Computational Geometry.
- [9] J. Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.
- [10] 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.
- [11] J. R. Shewchuk. Tetrahedral mesh generation by Delaunay refinement. In *14th Annual Symposium on Computational Geometry*, pages 86–95, 1998.
- [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 *11th 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.