Computing Acute and Non-obtuse Triangulations

Hale Erten Alper Üngör

Dept. of Computer & Information Science & Engineering, University of Florida
{herten,ungor}@cise.ufl.edu

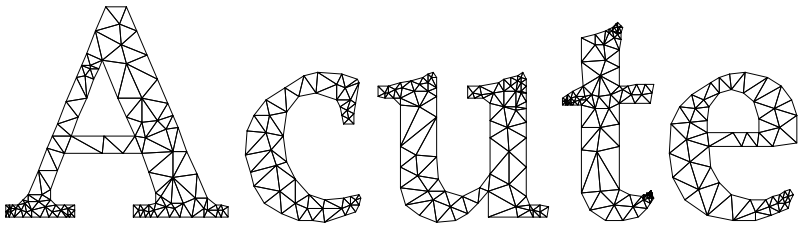

Figure 1: Acute triangulation of the domain **Acute** (in Palatino font).

Abstract

We propose a method for computing acute (non-obtuse) triangulations. That is, for a given two dimensional domain (a set of points or a planar straight line graph), we compute a triangulation of the domain such that all angles are less than (or less than or equal to) $\pi/2$. This leads to the first software to compute such triangulations.

1 Introduction

Large angles in a triangulation is known to lead to undesired numerical results in many scientific applications [1]. A lower bound on the smallest angle of a triangulation implies an immediate upper bound on its largest angle (the converse is not true). Most triangulation algorithms rely on this observation and hence focus on providing lower bounds on the smallest angle.

Problem 1 *[No Small Angle Triangulation]. Given a two dimensional input domain (point set or planar straight line graph) and a constraint angle α, compute a triangulation of the domain such that all angles are larger than α.*

There are many good solutions for this well-known problem [4, 11, 13, 19]. The lower bounds provided for the smallest angle, however, are not strong enough to address the acute and non-obtuse triangulation problems, which are defined below.

Problem 2 *[Acute Triangulation]. Given a two dimensional input domain (point set or planar straight line graph), compute a triangulation of the domain such that all angles are less than $\pi/2$.*

Problem 3 *[Non-obtuse Triangulation]. Given a two dimensional input domain (point set or planar straight line graph), compute a triangulation of the domain such that all angles are less than or equal to $\pi/2$.*

1.1 Motivation

Acute angle constraint is motivated by various scientific computing [20] and computer graphics applications [7, 14, 18, 20]. A recent space-time meshing algorithm, called the *tent-pitcher*, by Üngör and Sheffer [20] relies on the acute triangulation of the initial spatial domain. The algorithm is known to generate a valid space-time mesh if the initial space mesh is an acute triangulation [20], but may fail if there is an obtuse angle or even a right angle. Later, improved versions of the tent-pitching algorithm are proposed [10], removing the acute angle requirement. However, loss of efficiency is expected in practice, that is more space-time elements are needed, whenever the spatial triangulation contains obtuse triangles.

Kimmel and Sethian [14] also make use of acute triangulations in their work on computing geodesic paths on manifolds. Their algorithm, called the fast marching method, computes geodesic distances and shortest paths on triangulated domains. They provide a simpler version of the algorithm with a better accuracy analysis when the underlying triangulation is acute. Acute and non-obtuse triangulations are also employed in other graphics applications for guaranteeing validity of planar mesh embedding via discrete Harmonic maps [8].

1.2 Previous work

While their definitions look similar, these two problems differ greatly in their difficulty levels and hence the availability of the solutions. For instance, there are algorithms for computing a non-obtuse triangulation of simple polygons [2, 3, 5, 9], but no algorithm is known for computing an acute triangulation of arbitrary simple polygons. Algorithmic and existence results for acute triangulations are known when the input is a point set or a low-complexity polygon such as an obtuse triangle, a square, a quadrilateral or a pentagon. Lindgren [15] showed that at least eight triangles are needed to triangulate a square with all acute triangles. Later, Cassidy and Lord [6] showed that for any $n \geq 10$ (but not for $n = 9$) there is an acute triangulation of the square with exactly n triangles. Gardner [12] studied the acute triangulation problem when the input is simply a triangle. Manheimer proved that seven acute triangles are necessary and sufficient to subdivide a non-obtuse triangle [17]. Recently, Maehara [16] showed that an arbitrary quadrilateral can be tiled by 10 (but perhaps not by any fewer) acute triangles. If we restrict ourselves to two-dimensional point sets, a solution to the acute triangulation problem is given by Bern *et al.* [4]. Their approach uses a quadtree, and replaces the standard square quadtree cells by tiles with protrusions and indentations.

2 Proposed Solution

Our solution for the acute and non-obtuse triangulation problems relies on a recent Delaunay refinement algorithm proposed by the authors to address Problem 1 [11]. By changing the key components of this algorithm, we derive a method to compute acute and non-obtuse triangulations.

Delaunay refinement method involves first computing an initial Delaunay triangulation of the input domain, and then iteratively adding points called *Steiner points* to remove the triangles with small angles. Traditionally, circumcenters of bad triangles are used as Steiner points [19]. An alternative type of Steiner points, called *offcenters* , is introduced later, which lead to the design of the first time-optimal Delanuay refinement algorithm [13]. The offcenter insertion algorithm is now used as the default option in the popular Delaunay refinement software `Triangle` [1]. Both the original circumcenter and the offcenter insertion variants of the Delaunay refinement suffer from a so-called *termination problem* for large values of the constrained angle α. Recently, the authors developed a Delaunay refinement algorithm to alleviate this problem. The original Delaunay refinement algorithm of Ruppert is proven to terminate with size-optimal quality triangulations for $\alpha \leq 20.7°$. In practice, it generally works for $\alpha \leq 34°$ and fails to terminate for larger constraint angles. The new variant of the Delaunay refinement algorithm generally terminates for constraint angles up to 42°. Experiments also indicate that this new refinement algorithm computes significantly (almost by a factor of two) smaller triangulations than the output of the previous Delaunay refinement algorithms.

The performance of the new refinement algorithm falls short of the 45° bound, which would imply a solution for Problems 2 and 3. Nevertheless it enabled us to design an acute/non-obtuse algorithm. There are two key components of this algorithm. First, it uses the Voronoi diagram of the iteratively refined domain to perform a search to find locally optimal Steiner points for insertion. Second, it performs a simple relocation step for the previous Steiner points to see a bad triangle can be fixed without a new Steiner point insertion. Below, we describe the adoption of these key components for a solution for acute/non-obtuse triangulation.

2.1 Walking on the Voronoi diagram

Let pqr be an acute (non-obtuse) triangle in a triangulation and pq be its shortest edge. See Figure 2. Let $slab(pq)$ denote the region between the two lines one going through p, the other q and both orthogonal to line segment pq. Let $slice(pq)$ be the intersection of $slab(pq)$ and the circumcircle of pqr. In the following algorithm, we suggest to insert a Steiner point strictly inside the slice region of a non-acute triangle furthest away from all existing vertices. (For non-obtuse triangulations we also consider the boundary of the slice as an insertion region.) Note that this point is either a Voronoi vertex or on a Voronoi edge. We find it simply by searching a local neighborhood on the Voronoi diagram.

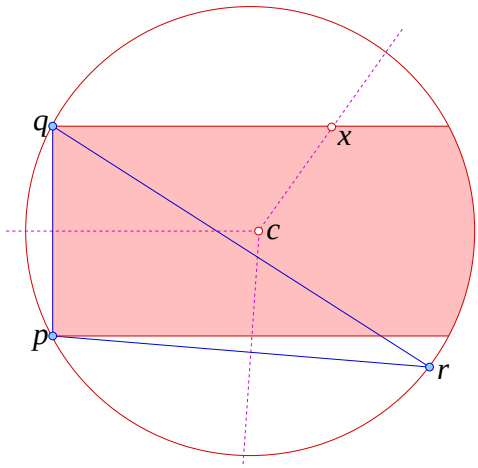

Figure 2: Point x is inside the slice of pq (shaded) that is furthest away from all existing vertices.

[1] `http://www-2.cs.cmu.edu/~quake/triangle.html`

2.2 Relocating Steiner points

We propose simple point relocation strategy to be integrated into the refinement procedure described in the previous section. The feasibility of this point relocation is tested before every iteration of a Steiner point insertion. We first recall the definitions of basic structures in a triangulation. The *star* of a vertex a consists of all triangles that contain a. The *link* of a, then, consists of all edges of triangles in the star that are disjoint from a. A vertex is said to be *free* if it was inserted by the refinement algorithm as a Steiner point. Input vertices are not free and never relocated. For each non-acute (obtuse) triangle, we first attempt to relocate its free vertices (one-at-a-time). See Figure 3. For a free vertex a, such a relocation is feasible only if the intersection of the slabs on the link of a is non-empty. We compute this intersection and perform a simple search in it. If one of its free vertices of a triangle find a new location so that all the triangles in its (new) star become acute (non-obtuse), we perform the relocation. Otherwise, we proceed with a new Steiner point insertion.

Algorithm 1

Compute the Delaunay triangulation ($DelTri$) of the input
Let P denote the maintained point set
while $\exists$ a non-acute triangle pqr in $DelTri(P)$
  relocated := FALSE
  for each free vertex a of pqr
    if $\mathcal{K} = \bigcap_{xy \in link(a)} slab(xy) \neq \emptyset$
      and $\exists b \in \mathcal{K}$ such that all triangles of $star(b)$ in the $DelTri(P \cup \{b\} - \{a\})$ are acute
    then
      delete a; insert b; relocated:=TRUE; **break**;
  endfor
  if relocated == FALSE **then**
    insert a point $x \in slice(pq)$ that is furthest from all existing vertices
endwhile

3 Experimental Study

We tested a preliminary implementation of our algorithm on several data sets. Our results are summarized in Table 3, and sample output meshes are illustrated in Figure 4. Histogram of angles for an output sample is shown in Figure 5. Our implementation is currently being optimized.

4 Discussions

We present the first heuristic method and the first software for computing acute (and non-obtuse) triangulations of two dimensional domains. We leave finding a theoretical support for the proposed approach an open problem. We should note that boundary handling strategy we use is inherited from the standard Delaunay refinement algorithms [11]. We believe the performance of our algorithm will improve once a special boundary handling rule suited for this problem is integrated into our software.

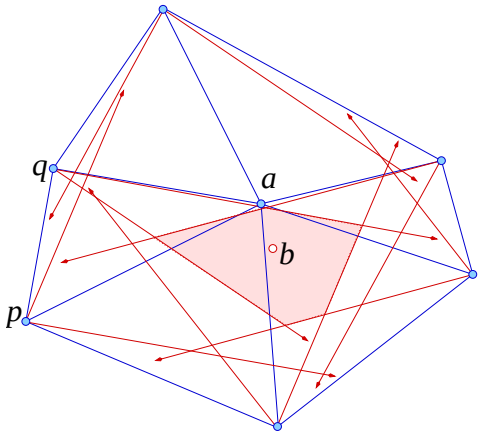

Figure 3: A non-acute (obtuse) triangle pqa might be fixed by relocating one of its vertices a. Such a relocation point, if exists, must be in the intersection region (shaded) of the slabs of edges on the link of a.

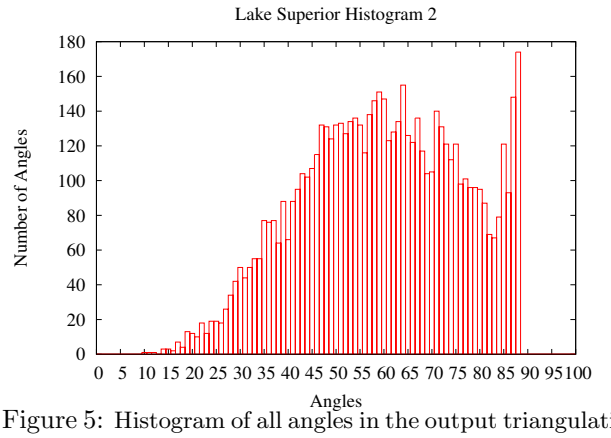

Figure 5: Histogram of all angles in the output triangulation for input set Lake Superior.

References

[1] I. Babuška and A. Aziz. On the angle condition in the finite element method. *SIAM J. Numer. Analysis 13:214–227, 1976.*

[2] B. S. Baker, E. Grosse, and C. S. Rafferty. Nonobtuse triangulation of polygons. Disc. & Comp. Geometry 3:147–168, 1988.

[3] M. Bern and D. Eppstein. Polynomial-size nonobtuse triangulation of polygons. Int. J. Comp. Geometry and Applications 2:241–255, 1992.

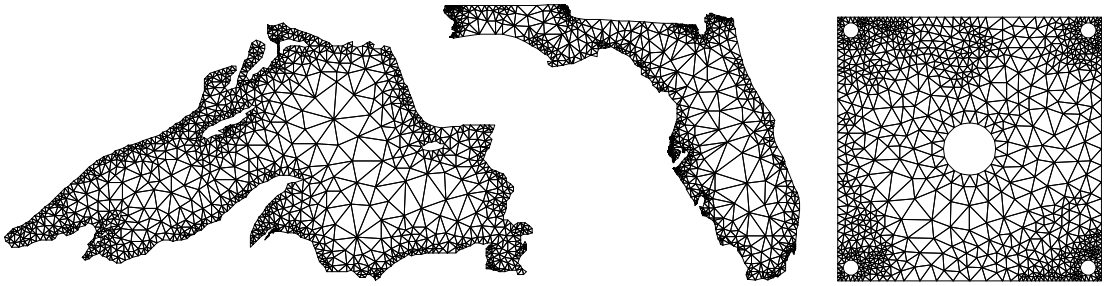

Figure 4: Acute triangulations for various input domains.

Data Set			Output			Time
name	# points	# edges	# points	# triangles	max angle	millisec
Acute	153	153	742	1086	89.597°	761
			332	420	90°	168
Superior	522	522	1649	2688	87.994°	5395
			1267	1983	90°	3095
Turkey	216	216	2296	4027	89.995°	7771
			1880	3314	90°	6687
Florida	304	304	1807	3043	89.998°	9817
			1248	1875	90°	1286
Boeing	30	30	188	315	88.978°	945
			109	175	90°	577
Random	1004	4	2482	4793	89.89°	5818
			2424	4683	90°	5650
plate	65	65	236	357	89.186°	495
			216	322	90°	190

Table 1: Performance of the algorithm on different data sets.

[4] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. J. Comp. System Sciences 48:384–409, 1994.

[5] M. Bern, S. Mitchell, and J. Ruppert. Linear-size nonobtuse triangulation of polygons. Proc. 10th ACM Symp. Comp. Geometry, 221–230, 1994.

[6] C. Cassidy and G. Lord. A square acutely triangulated. J. Rec. Math. 13(4):263–268, 1980.

[7] H.Q. Dinh, A.J. Yezzi, G. Turk. Texture transfer during shape transformation. ACM Trans. Graph. 24(2): 289–310 2005.

[8] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. Proc. SIGGRAPH, 173–182, 1995.

[9] D. Eppstein. Faster circle packing with applications to nonobtuse triangulation. Int. J. Comput. Geometry Appl. 7(5): 485–492, 1997.

[10] J. Erickson, D. Guoy, J.M. Sullivan, A. Üngör. Building spacetime meshes over arbitrary spatial domains. Eng. Comput. (Lond.) 20(4): 342–353, 2005.

[11] H. Erten and A. Üngör. Triangulations with Locally Optimal Steiner Points. Proc. Eurographics Symp. Geometry Processing, Barcelona Spain, 2007

[12] M. Gardner. Mathematical games. Scientific American 202:177–178, 1960.

[13] S. Har-Peled and A. Üngör. A time-optimal Delaunay refinement algorithm in two dimensions. Proc. ACM Symp. Comp. Geometry, 228–236,2005.

[14] R. Kimmel and J.A. Sethian. Computing geodesic paths on manifolds. Proc. of National Academy of Sciences, 95(15):8431–8435, 1998.

[15] H. Lindgren. Geometric dissections. Van Nostrand, Princeton, N. J., 1964.

[16] H. Maehara. Acute triangulations of polygons. Proc. of Japan Conf. on Disc. & Comp. Geometry, 237–243, 2000. Springer-Verlag, LNCS 2098.

[17] W. Manheimer. Dissecting an obtuse triangle into acute triangles. American Math. Monthly, 67, 1960.

[18] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. Exper. Math. 2(1):15–36, 1993.

[19] J. Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. Proc. 4th ACM-SIAM Symp. on Disc. Algorithms, 83–92, 1993.

[20] A. Üngör and A. Sheffer. Pitching tents in space-time: Mesh generation for discontinuous Galerkin method. J. of Found. of Computer Science 13(2):201–221, 2002.