Computational Geometry 47 (2014) 741–778

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Reprint of: Delaunay refinement algorithms for triangular mesh generation ☆

Jonathan Richard Shewchuk [1]

Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720, USA

ARTICLE INFO

Article history:
Available online 26 February 2014
Communicated by S. Fortune

Keywords:
Triangular mesh generation
Delaunay triangulation
Constrained Delaunay triangulation
Delaunay refinement
Computational geometry

ABSTRACT

Delaunay refinement is a technique for generating unstructured meshes of triangles for use in interpolation, the finite element method, and the finite volume method. In theory and practice, meshes produced by Delaunay refinement satisfy guaranteed bounds on angles, edge lengths, the number of triangles, and the grading of triangles from small to large sizes. This article presents an intuitive framework for analyzing Delaunay refinement algorithms that unifies the pioneering mesh generation algorithms of L. Paul Chew and Jim Ruppert, improves the algorithms in several minor ways, and most importantly, helps to solve the difficult problem of meshing nonmanifold domains with small angles.

Although small angles inherent in the input geometry cannot be removed, one would like to triangulate a domain without creating any *new* small angles. Unfortunately, this problem is not always soluble. A compromise is necessary. A Delaunay refinement algorithm is presented that can create a mesh in which most angles are 30° or greater and no angle is smaller than $\arcsin[(\sqrt{3}/2)\sin(\phi/2)] \sim (\sqrt{3}/4)\phi$, where $\phi \leqslant 60°$ is the smallest angle separating two segments of the input domain. New angles smaller than 30° appear only near input angles smaller than 60°. In practice, the algorithm's performance is better than these bounds suggest.

Another new result is that Ruppert's analysis technique can be used to reanalyze one of Chew's algorithms. Chew proved that his algorithm produces no angle smaller than 30° (barring small input angles), but without any guarantees on grading or number of triangles. He conjectures that his algorithm offers such guarantees. His conjecture is conditionally confirmed here: if the angle bound is relaxed to less than 26.5°, Chew's algorithm produces meshes (of domains without small input angles) that are nicely graded and size-optimal.

1. Introduction

Delaunay refinement is a technique for generating triangular meshes suitable for use in interpolation, the finite element method, and the finite volume method. The problem is to find a triangulation that covers a specified domain, and contains

DOI of original article: http://dx.doi.org/10.1016/S0925-7721(01)00047-5.

☆ The article is reprinted in recognition of Dr. Shewchuk's receipt of the 2012 Test of Time award. For citation purposes, please use the original publication details: J.R. Shewchuk, Computational Geometry 22 (2002) 21–74.

E-mail address: jrs@cs.berkeley.edu.

[1] Supported in part by the National Science Foundation under Awards ACI-9875170, CMS-9980063, CMS-9318163, and EIA-9802069, in part by the Advanced Research Projects Agency and Rome Laboratory, Air Force Materiel Command, USAF under agreement number F30602-96-1-0287, in part by the Natural Sciences and Engineering Research Council of Canada under a 1967 Science and Engineering Scholarship, and in part by gifts from the Okawa Foundation and Intel.

http://dx.doi.org/10.1016/j.comgeo.2014.02.005
0925-7721/

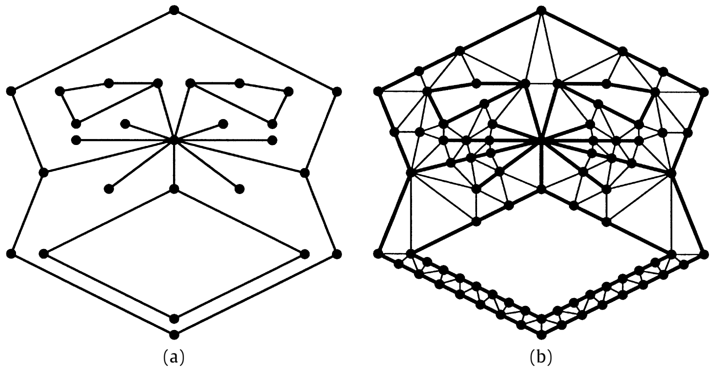

Fig. 1. A PSLG and a mesh generated by Ruppert's Delaunay refinement algorithm.

only triangles whose shapes and sizes satisfy constraints: the angles should not be too small or too large, and the triangles should not be much smaller than necessary, nor larger than desired. Delaunay refinement algorithms offer mathematical guarantees that such constraints can be met. They also perform excellently in practice.

This article has three purposes. First, it offers a theoretical framework for Delaunay refinement algorithms that makes it easy to understand why different variations of Delaunay refinement are successful. This framework is used to clarify the performance of an algorithm by Ruppert, to reanalyze an algorithm by Chew, and to generate several extensions of Delaunay refinement. Second, this article exploits the framework to help find a practical solution to the difficult problem of meshing domains with small angles that Delaunay refinement algorithms proposed to date cannot mesh. Third, it presents almost everything algorithmic a programmer needs to know to implement a state-of-the-art triangular mesh generator for straight-line domains. (Curved boundaries and surfaces, however, are not treated here. Thorough treatments of data structures and Delaunay triangulation algorithms are available elsewhere [8,17,29].)

A full description of the mesh generation problem begins with the domain to be meshed. Most theoretical treatments of meshing take as their input a *planar straight line graph* (PSLG). A PSLG is a set of vertices and segments, like that illustrated in Fig. 1(a). A segment is an edge that must be represented by a sequence of contiguous edges in the final mesh, as Fig. 1(b) shows. By definition, a PSLG is required to contain both endpoints of every segment it contains, and a segment may intersect vertices and other segments only at its endpoints. (A set of segments that does not satisfy this condition can be converted into a set of segments that does. Run a segment intersection algorithm [3,12,28], then divide each segment into smaller segments at the points where it intersects other segments or vertices.)

The process of mesh generation necessarily divides each segment into smaller edges called *subsegments*. The bold edges in Fig. 1(b) are subsegments; other edges are not. The *triangulation domain* is the region that a user wishes to triangulate. For mesh generation, a PSLG must be *segment-bounded*, meaning that segments of the PSLG entirely cover the boundary separating the triangulation domain from its complement, the *exterior domain*. A triangulation domain need not be convex, and it may enclose untriangulated holes, but the holes must also be bounded by segments. A segment must lie anywhere a triangulated region of the plane meets an untriangulated region.

A mesh generator produces a triangulation that attempts to satisfy three goals. First, the union of the triangles is the triangulation domain, and the triangulation *respects* the segments—each segment is a union of triangulation edges.

Second, the triangles should be relatively "round" in shape, because triangles with large or small angles can degrade the quality of the numerical solution to a finite element problem. In interpolation, triangles with large angles can cause large errors in the gradients of the interpolated surface. In the finite element method, large angles can cause a large *discretization error* [1]; the solution may be less accurate than the method would normally promise. Small angles can cause the coupled systems of algebraic equations that the finite element method yields to be ill-conditioned [7].

A lower bound on the smallest angle of a triangulation implicitly bounds the largest angle. If no angle is smaller than θ, no angle is larger than $180° - 2\theta$. Hence, many mesh generation algorithms, including the Delaunay refinement algorithms studied here, take the approach of attempting to bound the smallest angle.

A third goal is to offer as much control as possible over the sizes of triangles in the mesh. Some meshing algorithms, including algorithms by Baker, Grosse, and Rafferty [2] and Chew [9], produce only *uniform meshes*, in which all triangles have roughly the same size. Other algorithms offer rapid *grading*—the ability to grade from small to large triangles over a relatively short distance. Small, densely packed triangles offer more accuracy than larger, sparsely packed triangles; but the computation time required to solve a problem is proportional to the number of triangles. Hence, choosing a triangle size entails trading off speed and accuracy. In the finite element method, the triangle size required to attain a given amount of accuracy depends upon the behavior of the physical phenomena being modeled, and may vary throughout the problem domain.

Given a *coarse* mesh—one with relatively few triangles—it is not difficult to *refine* it to produce another mesh having a larger number of smaller triangles [18]. The reverse process is not so easy [22]. Hence, mesh generation algorithms often set themselves the goal of being able, in principle, to generate a mesh with as few triangles as possible. They typically offer