A Guaranteed Quality Boundary Graded Triangular Meshing Algorithm Backed by a Computer-Assisted Proof

Jing Xu and Andrey N. Chernikov

Department of Computer Science, Old Dominion University, Norfolk, VA, USA,
{jxu,achernik}@cs.odu.edu

Summary. The quality of mesh elements including shape and size influences the performance of finite elements analysis, interpolation, and contouring. The existence of small angles is especially critical for conditioning of the stiffness matrix in finite elements methods. In this paper we present a boundary graded triangular mesh generation algorithm that allows for guaranteed bounds on angles. Our proof program shows that our algorithm bounds the minimum angle above 5.65°.

Key words: mesh generation, angle bounds, quadtree

1 Introduction

There is a large body of algorithms that theoretically guarantee a quality mesh. Mesh generation by Delaunay refinement, which allows for the mathematical proofs of termination and good grading, is one of the push-button algorithms used for constructing guaranteed quality triangular and tetrahedral meshes. Quality is traditionally defined in terms of the bounds on circumradius-to-shortest-edge ratio [2].

Another kind of mesh generation algorithms that provide high quality non-uniform meshes takes advantage of the structure of a quadtree (octree in three-dimension). Quadtrees were introduced for domain decomposition to generate non-uniform meshes by Yerry and Shephard [7], however, there is no proven quality or size bounds involved in the paper.

Later, Bern, Eppstein, and Gilbert [5] studied several versions of generating triangular meshes of a planar point set or polygonally bounded domain which guarantee well-shaped elements and small total size simultaneously (Mitchell and Vavasis [6] extended Bern's work in three dimensions). They achieved angle bounds between 18.4° and 153.2° for the minimum and the maximum angle, respectively.

Drawing inspiration from the Marching Cubes algorithm [4] and Bern et al's work [5], Labelle and Shewchuk [3] adopted the idea of *warping* and proposed the Isosurface Stuffing tetrahedral meshing algorithm on geometric

domains represented by a continuous cut function. The algorithm chooses *body centered cubic* (*BCC*) *lattice* as background grid of excellent quality, computes or approximates *cut points*, warps the vertices of background grid to *cut points*, or inserts *cut points* from a set of predetermined stencils. The algorithm is accompanied by guaranteed dihedral angle bounds. A variant of the algorithm creates meshes with interior grading based on octree, however, the authors didn't report details on surface grading on the reason that they can not make guarantees on dihedral angles better than 1.66° for min angle or 174.72° for max angle.

Our work is based on the Isosurface Stuffing algorithm: we want to expand the algorithm to more general cases, in which it provides strong angle bounds for the meshes that allow for grading on the surface. We present preliminary results in this paper on the proof of the 2-dimensional non-uniform boundary grading algorithm, with minimum angle bound of 5.65°. By placing a lower bound on the smallest angle of a triangulation, one is also bounding the largest angle, since in two dimensions, if no angle is smaller than θ, then no angle is larger than $180° - 2\theta$. The Our algorithm also relies on a balanced quadtree subdivision that offers boundary grading. We can specify any leaf side length, which means the element size can be defined by a user. Compared with Bern's work, our algorithm does not suffer strict restriction on grading.

The rest of the note is organized as follows. In Section 2 we describe the boundary grading quality mesh generation algorithm. In Section 3 we present our angle bounds based on a computer-assisted proof. Section 4 concludes the paper.

2 Boundary Graded Mesh Generation Algorithm

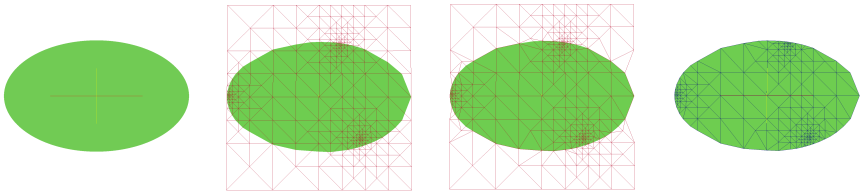
Our algorithm starts with constructing a balanced quadtree that covers the model, then fills the quadtree leaves with high quality template elements (isosceles right triangles), and finally warps the mesh vertices onto the model surface, or inserts *cut points* from a predetermined table. Fig. 1 illustrates the main steps performed by our algorithm. The meshes whose input is an image are showed in Fig. 2.

2.1 Construction of the Balanced Quadtree

There are two conditions that control the quadtree splitting:

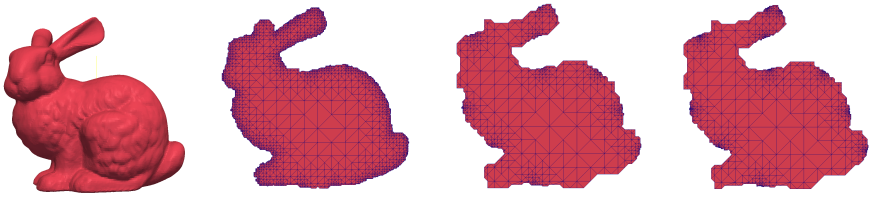
(i) the current square contains one or more selected points;
(ii) the side length of the current square is longer than the side length of user-defined minimum leaf square.

After the quadtree subdivision, we balance it by requiring that any two neighboring squares differ at most by a factor of two in size. To fill the leaves in the quadtree to generate the background grid, we use the following strategy [1]: we say that the side of a leaf square is *split* if either of the neighboring leaf squares sharing it is split. If at least one of its side is *split* by a midpoint,



(a) (b) (c) (d)

Fig. 1. An illustration of the main steps of our algorithm. (a) The input is a continuous function: an ellipse. (b) The quadtree with leaves refined by 3 selected points and filled with high quality template elements. (c) The background grids after *warping*. (d) The final mesh.



(a) (b) (c) (d)

Fig. 2. Mesh results of the use of our algorithm. All the angles are bounded from below by 5.65°. (a) The original image. (b) The final mesh refined with points everywhere on the boundary. (c) The final mesh refined with fewer points. (d) The final mesh refined with finer elements.

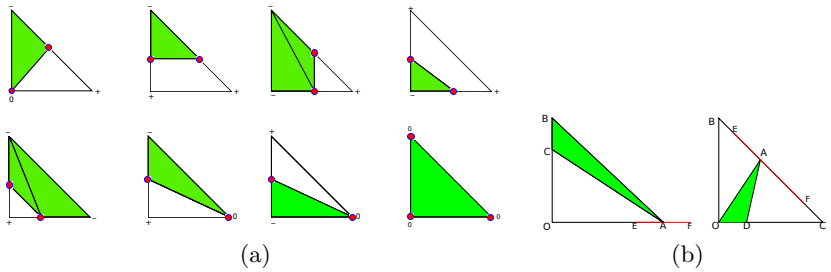
introduce the center of the leaf square and connect the center to the midpoint and the four corners of the leaf square. If none of the sides was split, we only add one diagonal of the leaf square. After this procedure, all the leaves are filled with isosceles right triangles.

2.2 Warping

We use two rules to decide if a background grid vertex needs to be warped: if the distance between a *cut point* and a grid vertex is less than α (a parameter in range from 0 to 0.5) times the length of grid edge; and if there are several *cut points* adjoining to the grid vertex, we always choose the nearest one to warp to, so that small elements can not be severely distorted by larger ones.

3 Proof of the Angle Bounds

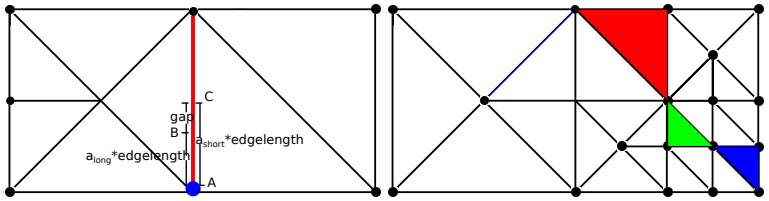
We offer the angle bounds through a computer-assisted proof. We wrote a program to find the most suitable α by which the best minimum angle can be obtained.



(a) (b)

Fig. 3. (a) Predetermined table we use in our algorithm with 8 stencils. (b) Two cases in which the angle is not monotonically increasing or decreasing. Vertex A can move on segment EF, the green angles at A are those we concern about.

The analysis begins with the observation that in most of the stencils of our table (Fig. 3(a)), we can get the angle extrema by only considering the endpoints of segments: if we fix one vertex of the changeable two and move the other along the segment, the angle is monotonically increasing or decreasing, so we can get the angle extrema by comparing the angles obtained at the endpoints of all segment. But there are a few exception cases, in which angles are not monotonically changing. These cases can be grouped into only two cases, illustrated in Fig. 3(b). We can easily prove that in these cases, the angle extrema are also obtained at the endpoints of segments.



(a) (b)

Fig. 4. (a) An illustration of the reason of the equality of α for long edges and short edges. (b) Case in which the minimum angle appears.

In some cases, certain edge (illustrated in Fig. 4(a), the red edge) can be the long edge of one triangle and at the same time be the short edge of the other. For the blue grid vertex in the figure, the warping distance ranges from A to B for the left small triangle. For the right big triangle, the warping can range from A to C. There exists a gap between B and C. If the *cut point* lies in the range of B to C, for the left one, the blue point doesn't warp; while for the right one, it warps. This equals to forcing the α for the long edge to become the same as α for the short edge. From this analysis, we conclude that the value of α for the long edge and for the short edge must be set equally.

For the uniform case, the warping distance for each stencil is the same, but this is not the case for boundary grading, because bigger neighbors can make smaller triangles warp their vertices a longer distance. We show the case in which the minimum angle appears (illustrated in Fig. 4(b)). The vertex of the green triangle shared with the red triangle can be warped a long distance along the hypotenuse of the red triangle; the vertex shared by the green triangle and the blue triangle can be warped the same distance as in the uniform case.

We compute the value of α within a margin of 0.000001. Our proof program shows that when α equals to 0.309017, our algorithm bounds the minimum angle within 5.65°.

4 Conclusion

In this paper, we presented our 2D boundary grading mesh generation algorithm. We provide the angle bounds that make the resulting meshes suitable for FE simulation. For the future work, we want to construct a formal proof using the Coq proof assistant to verify the angle bounds. The language used in Coq called Gallina is more restrictive than the imperative programming languages, and therefore provides sound logical reasoning. The results in this paper are only a 2D preliminary work; we are planning to extend it to the 3D tetrahedral surface grading mesh generation and its quality proof.

References

1. Andrey Chernikov and Nikos Chrisochoides. Multitissue tetrahedral image-to-mesh conversion with guaranteed quality and fidelity. *SIAM Journal on Scientific Computing*, 33:3491–3508, 2011.
2. L. P. Chew. Guaranteed-Quality Triangular Meshes. Tech. Rep. pages TR–89–983, Department of Computer Science, Cornell University, 1989.
3. Franois Labelle and Jonathan Richard Shewchuk. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. In *ACM Transactions on Graphics, special issue on Proceedings of SIGGRAPH 2007*, volume 26(3), pages 57.1–57.10, August 2007.
4. W. E. Lorensen and H. E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Computer Graphics (SIGGRAPH87 Proceedings)*, volume 29, 4, pages 163–170, 1987.
5. D. Eppstein M. Bern and J. Gilbert. provably good mesh generation. In *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science*, pages 231–241, New York, 1990.
6. S. A. Mitchell and S. A. Vavasis. Quality mesh generation in three dimensions. In *the ACM Computational Geometry Conference*, pages 212–221, 1992.
7. Mark A. Yerry and Mark S. Shephard. A modified quadtree approach to finite element mesh generation. *Computer Graphics and Applications*, 3:39–46, 1983.