11-2009

# Fast Meshing for Acute PLCs

Gary L. Miller
*Carnegie Mellon University*

Todd Phillips
*Carnegie Mellon University*

# Fast Meshing for Acute PLCs

Gary L. Miller Todd Phillips
Carnegie Mellon University

November 6, 2009


## Abstract

We present a new algorithm to mesh an arbitrary piecewise linear complex in three dimensions. The algorithm achieves an $O(n \log \Delta + m)$ runtime where $n$, $m$, and $\Delta$ are the input size, the output size, and spread respectively. This represents the first non-trivial runtime guarantee for this class of input. The new algorithm extends prior work on runtime-efficient meshing by allowing the input to have acute input angles (called creases). Features meeting at creases are handled with protective collars. A new procedure is given for creating these collars in an unstructured fashion, without the need for expensive sizing precomputation as in prior work. The collar surface dividing these two regions is represented implicitly using surface reconstruction techniques. This new approach allows the collar to be dynamically generated , allowing the whole algorithm to run in a single pass. For inputs with $\Delta$ bounded by a polynomial in $n$, this runtime is optimal.


## 1 Motivation

Conventional wisdom has often regarded the meshing problem as a pre-process to a series of heavy computations on the output mesh. However, state of the art problems now rely on evolving geometries, particularly for simulation problems involving fluid-solid interactions. In such simulations, a new mesh must be generated at every simulation timestep, either dynamically or from scratch. Additionally, the linear system solves which traditionally dominate compute time are increasingly fast, with modern solvers running at least sub-quadratic in the mesh size.

This creates a need for advanced algorithms that are efficient enough to avoid dominating the overall runtime. The Sparse Voronoi Refinement (SVR) algorithm [3] represented the first algorithm for meshing non-acute PLCs in more than two dimensions with a non-trivial runtime bound. We have extended this algorithm to handle acute PLCs in three dimensions with the same overall runtime guarantees.

## 2 Algorithm Overview

We follow a standard iterative Delaunay (Voronoi) refinement paradigm, wherein we continually add new vertices to a mesh with the two goals of increasing element quality, and conforming to an input PLC. The algorithmic difficulties for handling acute PLCs arise due to a conflict between these two goals. It is impossible to fit quality tetrahedra inside creases of the PLC. This is circumvented by giving two different guarantees on element quality.

Adjacent to the creases of the input, we generate tetrahedra with no large dihedral angles. In regions disjoint from the creases, we generate tetrahedra with a good ratio of circumradius to shortest edge. These are almost all good aspect ratio tetrahedra, with the exception of slivers. (The sliver removal post-process of [4] could be easily modified to improve the aspect ratio without additional runtime cost.)

Unfortunately, a proper choice for these two regions is not known in advance, and the problem of distinguishing these regions is fundamentally the same as generating the entire mesh. We observe that the regions that need to be specially treated are precisely those where a traditional meshing algorithm will not terminate, refining forever based on the conflicting goals of quality and conforming.

Ruppert [6] was the first to address the issue of small input angles in a refinement based algorithms. His idea for 2D meshing was to add “small” protective balls around vertices with small input angles thus removing the small input angles from the mesh. In a series of papers, this idea has been extended to 3D by adding a set of balls (collars) around input points and edges that are common to a small input angles [5, 1]. Their constructions are nontrivial. Unfortunately, none of these algorithms have sub-quadratic run times even for inputs with polynomially bounded spread.

As with all refinement algorithms, SVR does not terminate when the input features contain small angles. Even though such algorithms do not terminate, they do converge to a fixed mesh, though possibly infinite.

That is, if we pick any point $p$ in the input domain other than the vertex of a small input angle, the algorithm

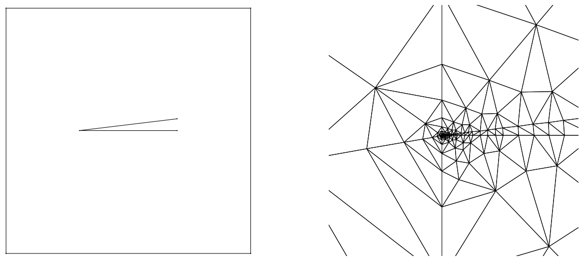

Figure 1: **Left:** An simple input consisting of two lines meeting at a crease. **Right:** An infinite quality mesh for this input.

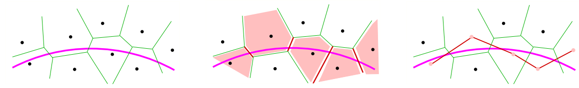

Figure 2: *Left:* A Voronoi diagram with a collar surface cutting through. *Center:* The restricted Voronoi cells. *Right:* The restricted Delaunay triangulation approximates the surface.

will eventually include the point in a fixed, finite triangle/simplex or Voronoi cell of the output. Our idea is to simulate the refinement algorithm generating an infinite mesh and use this mesh to specify our final finite mesh. Of coarse we will not let the mesh become of unbounded size.

In keeping with traditional approaches, we too create a *collar region* around the creases, and the mesh refinement within is handled as a special case to ensure termination. We achieve better runtime by creating this special region dynamically. Our on the fly calculations allow the region to be computed in a single pass with the rest of the algorithm, avoiding any expensive pre-processes.

When the mesh elements in the neighborhood of a crease are approximately small enough, a proper size is computed and the collar region is augmented. Previous approaches to collar regions either required expensive $O(n^2)$ pre-computations or assumed a lower-bound on size and create constant-sized collar regions. We use careful predicates and show that the collar region is always augmented before refinement progresses too far, and that the collar-sizing calculations can always be performed efficiently. In three dimensions, the creases may be corners or whole segments. The collar region will consist of a union of balls centered on these creases that will eventually cover all of the creases.

# 3 Collar Surface Construction

The infinite mesh will be generated fine enough so that it has a good approximation to the boundary of the collar region. It is necessary to have some approximation to this boundary surface so that the two meshes (exterior and interior the collar region) can be properly stitched together.

To extract from the infinite mesh an approximation to this surface, we use the notion of the *restricted Delaunay triangulation* [2]. Most prior work has proposed meshing so that all the vertices in the restricted Delaunay of the surface are explicitly on the surface. Forcing all these points to be on the boundary of the ball is not always necessary and may generate more vertices than needed. When the collar region is dynamically augmented and the surface is updated, we allow the existing mesh vertices to appear in the restricted Delaunay, and only require that any new vertices be placed on the surface. This relaxed approach makes it easier to analyze the runtime, and we believe will generate fewer vertices in practice.

A proper approximation to the surface of the collar region will need the appropriate topology to ensure a proper mesh when the interior and exterior are stitched together. Since the collar region is a union of balls, the interior will be meshed with stars of tetrahedra emanating from the crease to the surface of the collar region. To ensure that these tetrahedra have no large dihedral angles, the facets of the restricted Delaunay must approximate the surface normals.

To meet these two goals of topology and normal approximation, additional low-priority refinements are added to the meshing process whenever there are violations. Violations can be detected efficiently, and $\varepsilon$-sampling arguments bound the number of additional vertices to be inserted.

# References


[1] S.-W. Cheng and S.-H. Poon. Three-dimensional delaunay mesh generation. *Discrete Comput. Geom*, 36:419–456, 2006.

[2] T. K. Dey. *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press, Oct. 2006.

[3] B. Hudson, G. Miller, and T. Phillips. Sparse Voronoi Refinement. In *Proceedings of the 15th International Meshing Roundtable*, pages 339–356, Birmingham, Alabama, 2006. Long version available as Carnegie Mellon University Technical Report CMU-CS-06-132.

[4] X.-Y. Li and S.-H. Teng. Generating well-shaped Delaunay meshed in 3D. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 28–37. ACM Press, 2001.

[5] S. E. Pav and N. J. Walkington. Robust Three Dimensional Delaunay Refinement. In *Thirteenth International Meshing Roundtable*, pages 145–156, Williamsburg, Virginia, Sept. 2004. Sandia National Laboratories.

[6] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995. Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (Austin, TX, 1993).