# A Pliant Method for Anisotropic Mesh Generation

Frank J. Bossen* Paul S. Heckbert†
Computer Science Dept.
Carnegie Mellon University

**Abstract.** A new algorithm for the generation of anisotropic, unstructured triangular meshes in two dimensions is described. Inputs to the algorithm are the boundary geometry and a metric that specifies the desired element size and shape as a function of position. The algorithm is an example of what we call *pliant mesh generation*. It first constructs the constrained Delaunay triangulation of the domain, then iteratively smooths, refines, and retriangulates. On each iteration, a node is selected at random, it is repositioned according to attraction/repulsion with its neighbors, the neighborhood is retriangulated, and nodes are inserted or deleted as necessary. All operations are done relative to the metric tensor. This simple method generates high quality meshes whose elements conform well to the requested shape metric. The method appears particularly well suited to surface meshing and viscous flow simulations, where stretched triangles are desirable, and to time-dependent remeshing problems.

**Keywords:** unstructured mesh, Delaunay triangulation, element size function, Riemannian metric tensor, smoothing.

## 1 Introduction

Our goal is the development of a mesh generator that:

**is application-independent and modular.** The mesh generator should meet the needs of problems in structural analysis, heat transfer, fluid flow, electromagnetics, computer graphics, and other applications. It should be flexible enough to support these varied demands while remaining modular, with a general subroutine interface that allows it to be used inside an adaptive finite element solver or in other contexts.

**allows complex geometry.** The meshes produced should conform to complex boundary geometry.

**generates anisotropic and graded meshes.** It should be possible to generate meshes with elements whose size and shape vary with position. For some applications, equilateral triangles are desired, while for others, such as viscous flow simulations, extreme aspect ratios are wanted. The result should closely match the desired element sizes.

**is automatic and robust.** The mesh generator should require no user interaction beyond specification of the geometry and desired element size, and it should always produce a valid mesh.

In order to meet these goals, we use the following approach.

To support complex boundary geometries without user interaction, unstructured triangular meshes are employed, since structured meshes often require human intervention.

To accommodate the generation of anisotropic and graded meshes, the desired element size and shape are specified by an *element size function*. For graded, isotropic meshes, this function gives edge length as a function of position, while for anisotropic meshes, this function specifies edge length as a function of position and direction. This function can be represented by a $2 \times 2$ metric tensor which defines a field of ellipses that can vary in size, eccentricity, and orientation across the domain. This mechanism for controlling anisotropy and grading of the mesh can support both explicit user control, implicit constraints from the boundary geometry, and adaptivity.

In order to generate a quality mesh that conforms to the constraints of the boundary and the element size function, we employ a combination of techniques including smoothing, refinement and coarsening, local topological optimization, and Delaunay triangulation.

*Frank J. Bossen is currently with the Signal Processing Laboratory, Electrical Engineering Dept., EPFL, 1015 Lausanne, Switzerland. bossen@ltssg4.epfl.ch, http://ltswww.epfl.ch/~bossen

†Paul S. Heckbert, Computer Science Dept., Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh PA 15213-3891, USA. ph@cs.cmu.edu, http://www.cs.cmu.edu/~ph


Figure 1: Collective triangulation.

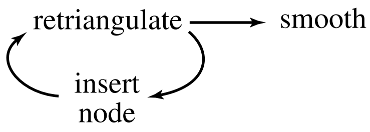



Figure 2: Incremental triangulation.

The remainder of the paper is organized as follows: After reviewing previous work, we give an overview of our basic method for the generation of graded, isotropic meshes. Next we generalize the method to anisotropic mesh generation. Finally, we give results and draw some conclusions.

## 2 Previous Work

In this section we review common mesh generation techniques and those that are closest to our own approach. More complete surveys are available [14, 19, 2].

### 2.1 Taxonomy

Although many hybrids exist, most mesh generation algorithms can be categorized into one of the following four classes. The first two are the most common.

- collective triangulation
- incremental triangulation
- pliant mesh generation with post-triangulation
- pliant mesh generation with retriangulation

*Collective triangulation* methods (Figure 1) choose the initial node positions, triangulate them as a whole, typically using Delaunay triangulation [11, 9], and then optionally adjust node positions using Laplacian smoothing. The principal difficulty with the collective approach is that poor choice of initial node positions can constrain the mesh to a poor topology.

*Incremental triangulation* methods (Figure 2) insert nodes one at a time, updating the triangulation during insertion (e.g. [8, 18, 4]). The advancing front method [16, 12] is one such technique. On the boundary, it produces well shaped elements, but where fronts collide in the interior, the elements can be distorted. Post-process smoothing can alleviate this problem.

An emerging approach is what we call *pliant mesh generation*, which we define to be methods in which smoothing, insertion, and deletion take place in a loop. Triangulation can be done in this pliant loop as well, or it can be done as a post-process. Pliant methods are often physically-based, in that they simulate physical behaviors such as attraction and repulsion, but they need not be[1]. Inserting and deleting nodes in this loop gives pliant methods population control, which allows them to make rapid changes to the topology when the local density of nodes and elements is too low or too high. Pliant methods tend to produce more uniform distributions of points than Laplacian smoothing.

One type of pliant mesh generation uses post-triangulation (Figure 3). Shimada and Gossard developed an algorithm that chooses initial node locations, smooths using attraction/repulsion forces between nearby nodes while occasionally inserting or deleting nodes, and finally performs Delaunay triangulation [20, 19]. In related work, Witkin and Heckbert used repelling particles to sample and polygonize curved surfaces [25].

Another form of pliant mesh generation uses retriangulation (Figure 4). Welch used a combination of Laplacian smoothing and surface Delaunay triangulation to maintain a quality mesh on a dynamic surface [24]. The method of Fortin *et al.* is also pliant [7].

[1] It is debatable whether a pliant method using Laplacian smoothing is "physically-based", for example.

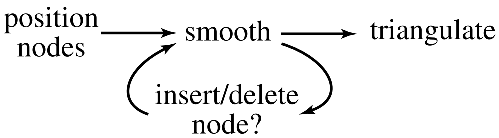



Figure 3: Pliant mesh generation with post-triangulation.

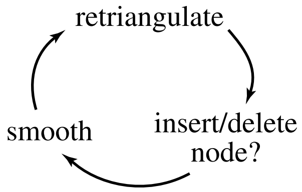



Figure 4: Pliant mesh generation with retriangulation.

### 2.2 Smoothing

The most popular repositioning technique, Laplacian smoothing, iteratively relocates each node to the centroid of its neighbors [6]. This method improves element shape in most cases, but on some concave domains, nodes are pulled outside the boundary. A more sophisticated variant is Laplace-Delaunay smoothing [6, 12], which performs both Delaunay edge swaps [11] and Laplacian smoothing in a loop, adjusting the topology and geometry of a mesh simultaneously. We do not regard Laplace-Delaunay smoothing by itself as a pliant method because it does not insert or delete nodes.

### 2.3 Graded and Anisotropic Meshes

In many cases, it is desirable to create graded meshes, where node spacing is a function of position, or anisotropic meshes, where node spacing is a function of position and direction. The desired size and shape of elements is application-dependent [21, 14]. For finite element approximation of many elliptic partial differential equations, equilateral triangles are ideal, but to simulate anisotropic diffusion or viscous fluid flow, elongated elements with extreme aspect ratio are often desirable. In structural simulations, smaller elements are needed in regions of stress concentration, while larger elements suffice in other regions. In the boundary layer around an aircraft wing, or along the shock fronts in supersonic flight, aspect ratios as high as several thousand are sometimes desirable.

Research in approximation theory has studied the effect of triangle size and shape on approximation error. One can ask: what triangulation with a given number of triangles minimizes the error of piecewise linear approximation to a known, smooth function? The answer is that as the number of triangles goes to infinity, the optimal triangles' orientation is given by the eigenvectors of the Hessian of the function at each point, and their size in each direction is given by the reciprocal square root of the absolute value of the corresponding eigenvalue [5, 21]. Thus, large angles can be optimal for minimizing approximation error [17].

Viscous flow simulations demanding extreme aspect ratios have traditionally been done using structured meshes exclusively, or with a combination of structured meshes along boundaries and unstructured meshes in the remainder of the domain. These approaches are difficult to automate for complex geometries, so researchers have sought to extend existing unstructured mesh generators to create highly anisotropic meshes.

Element size can be controlled by an *element size function* [8]. For graded, isotropic meshes in 2-D, element size is a scalar function of position, $r(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2)$. Ruppert defines an element size function in terms of distance to boundary features [18]. More explicit schemes often employ a *background mesh* [8, 16], which can be given by the user or can be the mesh from the previous iteration in an adaptive solver. The background mesh specifies the element size at each vertex, and sizes are then interpolated over its elements. For anisotropic mesh generation, the element size function is generalized to describe shape as well as size, with a $2 \times 2$ matrix $\mathbf{M}(\mathbf{x})$. In differential geometry, this is called a Riemannian metric tensor [22]. The element size function can be visualized as a field of ellipses whose major radius, minor radius, and angle are functions of position (see Figure 10b). The metric is isotropic iff $\mathbf{M}$ has equal eigenvalues, in which case the ellipses are circles. In the context of adaptive finite element methods, the metric tensor can be derived from the Hessian of a previous solution [16, 26].

For anisotropic mesh generation, distances are often defined using a local metric tensor. The advancing front method can remesh a domain anisotropically by positioning new nodes according to the metric [16]. Alternatively, Castro-Díaz *et al.* refine and coarsen the mesh by splitting and collapsing edges that are too long or too short according to the metric, swapping edges to maintain a quality triangulation [4]. Mavriplis places nodes using a structured mesh [13] or an advancing front [14], and then applies Delaunay triangulation in a stretched space. When smoothing is used on anisotropic meshes, it, too, should conform to the metric tensor [10]. Vallet *et al.* use a combination of insertion, deletion, retriangulation, and anisotropic smoothing [23, 7].

Anisotropic meshes can also be generated without reference to metrics or stretched space. Marcum generates meshes that are semi-structured and anisotropic near boundaries, but unstructured and isotropic in the interior [12].

In spite of the progress described above, problems remain. Most existing unstructured anisotropic mesh generators are unable to generate meshes with extreme aspect ratios. Aspect ratios larger than about 10 or 20 are problematic for most 3-D advancing front methods [15]. Marcum's algorithm can generate highly stretched elements, but it does not support general anisotropy, and it occasionally generates sliver tetrahedra [12]. Much work remains to be done to generate unstructured meshes automatically for problems involving complex geometry and extreme aspect ratios.

## 3 Basic Method

The new method we propose for mesh generation is a pliant method with retriangulation (Figure 4). We maintain a triangulation during the pliant iterations as an efficiency measure. This allows us to find neighboring nodes quickly, without the spatial data structures required by some other algorithms.

The inputs to the mesh generator are a polygonal boundary and an element size function. First, a constrained Delaunay triangulation of the domain is built. Then, at each step through the pliant loop, a node is picked at random and relocated. The mesh is retriangulated, and if necessary a node is inserted or deleted. This process is illustrated in Figure 5. Note that each iteration is very fast.

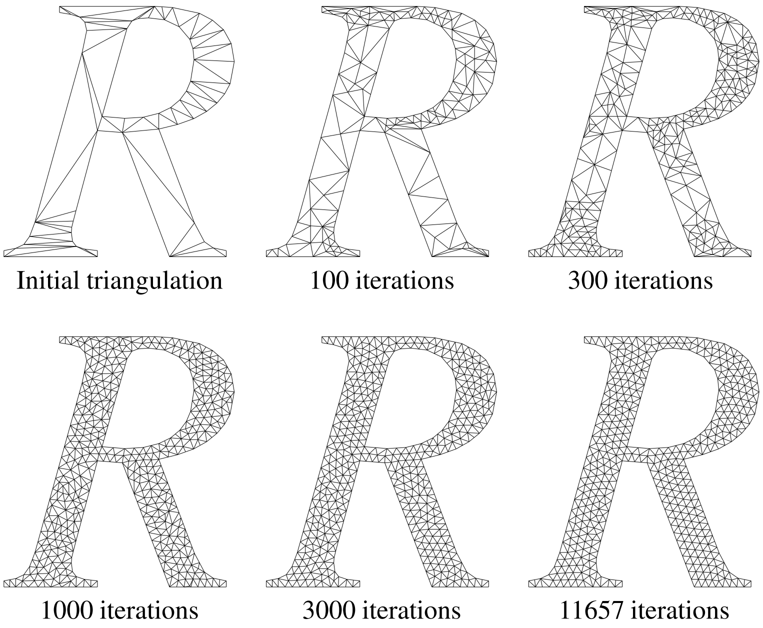



Figure 5: Mesh Evolution

A more precise description of the algorithm is given below:

```
create initial constrained Delaunay triangulation
until all nodes inactive
   randomly pick an active node i
   reposition node i according to positions of neighbors
   retriangulate to satisfy Delaunay criterion
   if extent of node i is too low, delete it and retriangulate
   else
      find extents of adjacent edges
      if largest edge extent is too high, split edge and retriangulate
   update active/inactive flags of node i and its neighbors
```

The next paragraphs describe in more detail each component of the mesh generation algorithm.

### 3.1 Smoothing

For graded, isotropic mesh generation, we define the distance $d(\mathbf{x}, \mathbf{y})$ between two points $\mathbf{x}$ and $\mathbf{y}$ as

$$d(\mathbf{x}, \mathbf{y}) = \frac{||\mathbf{x} - \mathbf{y}||}{r} \tag{1}$$

where $||\cdot||$ is the Euclidean norm operator, and $r$ the desired edge length, which could vary as a function of position. The terms "distance" and "normalized length" will mean distance defined this way. Our goal is to create meshes with normalized edge length of 1.

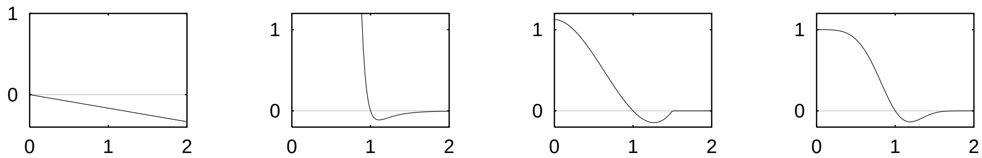
Given a definition of normalized distance, general smoothing can be expressed in terms of a first order[2] motion equation:

$$\mathbf{p}'_i = \mathbf{p}_i + \alpha_i \sum_{j \in \mathcal{N}_i} f(d(\mathbf{p}_i, \mathbf{p}_j))\, \mathbf{u}_{ij} \tag{2}$$

where $\mathbf{p}_i$ is the position of node $i$, $\mathbf{u}_{ij}$ is the unit vector $(\mathbf{p}_i - \mathbf{p}_j)/d(\mathbf{p}_i, \mathbf{p}_j)$, $\mathcal{N}_i$ is a neighborhood of $i$, $f$ is a smoothing function, and $\alpha_i$ is a constant.

The most common smoothing method is Laplacian smoothing [6], in which each node is moved to the centroid of its neighbors. Laplacian smoothing is thus defined by $\mathcal{N}_i = \{j | j \text{ shares an edge with } i\}$, $\alpha_i = 1/|\mathcal{N}_i|$, and $f(d) = -d$.

The Lennard–Jones potential from chemistry describes attraction/repulsion behavior [1]. Its smoothing function is $f(d) = d^{-13} - d^{-7}$. For mesh generation purposes, this model suffers from numerical instabilities, since $f(0^+) \to \infty$, so several variants have been proposed [19, 3].



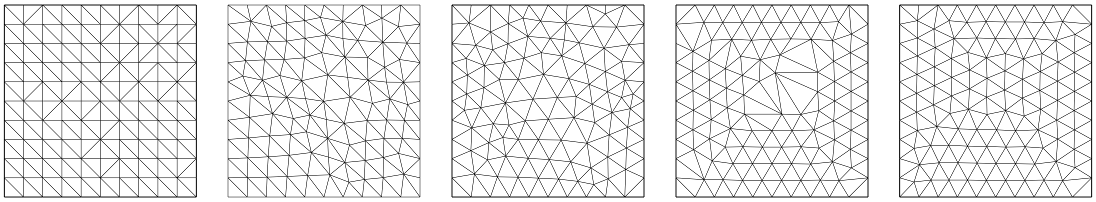

a) Laplacian b) Lennard–Jones c) Shimada d) Ours

Figure 6: Several smoothing functions $f(d)$

After testing several alternatives, we have chosen $f(d) = (1 - d^4) \cdot \exp(-d^4)$, pictured in Figure 6d. We have found that smoothing functions with such attraction/repulsion behavior yield better results [3]. If two nodes are too close to each other ($d < 1$), they *repel*, and if too distant ($d > 1$), they *attract*. This is different from Laplacian smoothing where nodes always attract each other, regardless of distance. Our neighborhood $\mathcal{N}_i$ is also defined as the set of nodes which share an edge with $i$, and $\alpha_i$ is set to 0.2.

[2] Second order is also possible [3, 19] but is not considered here.

The new position for the vertex is only accepted if it does not violate the triangulation. If the line segment between $\mathbf{p}_i$ and $\mathbf{p}_i'$ does not cross any edges, then the move is accepted. If it crosses an edge, then if the first edge crossed is a boundary edge, the node is positioned on that edge, otherwise the node is not moved.



a) Initial mesh b) Laplacian c) Laplace-Delaunay d) Our smoothing without insertion e) Our smoothing & insertion (pliant)

Figure 7: Smoothing techniques applied to the mesh at left

Figure 7 shows the effect of Laplacian smoothing, Laplace-Delaunay smoothing [6] versus the new method. Our smoothing technique generates regular meshes with some voids unless node insertion is used, so we have found that the two should be used together.

### 3.2 Node Insertion and Deletion

When the mesh is locally too sparse, a node is inserted, and when it is too dense, a node is deleted. We quantify sparseness using *extent*, which has the units of normalized area. Normalized area is simply area divided by $r^2$. The extent of a node on the boundary is defined to be 0.5 times the squared sum of the normalized lengths of the two boundary edges adjacent to it, and the extent of an internal node is $2.3/\sqrt{n(n-2)}$ times the sum of the normalized areas of the $n$ triangles that are adjacent to it[3]. The extent of a boundary edge is defined to be 0.5 times the square of its normalized length, and the extent of an internal edge is 0.8 times the sum of the normalized areas of the two triangles adjacent to it.

Let $i$ be the node picked in the main loop. If the extent of node $i$ is below 1, then it is deleted, and the mesh is locally retriangulated according to the Delaunay criterion. If node $i$ is not deleted, the extent of each edge adjacent to node $i$ is computed. If the largest of these extents is greater than 1 then the corresponding edge is split at its midpoint and the mesh is retriangulated. Boundary nodes present in the input are never repositioned or deleted.

Retriangulation after insertion is done using incremental Delaunay triangulation [11, 9]. In our context, point location is unnecessary because we know the edge being split. After deletion, we construct an initial triangulation of the resulting simple polygon and apply edge swapping to the edges internal to the polygon [3].

### 3.3 Node Tagging and Convergence

This combination of smoothing, retriangulation, and insertion/deletion tends to drive all the edges to a normalized length of 1. To monitor convergence, nodes are marked as active or inactive. Initially all nodes are active. Only active nodes can be picked for smoothing. After a node is relaxed it is kept active or not according to its speed $||\mathbf{p}_i' - \mathbf{p}_i||$. If its speed is above a threshold, it is kept active and its neighbors are marked active, but if below the threshold, the node is marked inactive. When a node is inserted or deleted, its neighbors become active. The algorithm stops when no more nodes are active.

So far, we have described an algorithm that produces high quality graded, isotropic meshes.

## 4 Anisotropy

In this section, we generalize the method to create anisotropic meshes. A $2 \times 2$ symmetric, positive definite tensor $\mathbf{M}(\mathbf{x})$ is used to quantify the desired element size as a function of position. For a graded, isotropic mesh with element size $r(\mathbf{x})$, $\mathbf{M}$ would be a diagonal matrix with $m_{11} = m_{22} = 1/r^2(\mathbf{x})$.

[3] These constants are derived in [3].

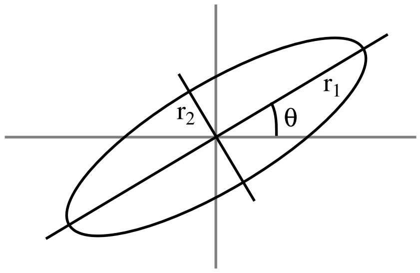



Figure 8: Ellipse

The specification of anisotropy requires three parameters: a major radius $r_1$, minor radius $r_2$, and angle $\theta$ (Figure 8), defining the desired edge length as a function of orientation and position. The corresponding tensor is:

$$\mathbf{M} = \mathbf{R}\boldsymbol{\Lambda}\mathbf{R}^T = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1/r_1^2 & 0 \\ 0 & 1/r_2^2 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{3}$$

Inversely, given a metric tensor, its eigenvalues and eigenvectors define the inverse squares of the major and minor radii and the directions of the axes, respectively.

Using this tensor, the distance between two points $\mathbf{x}$ and $\mathbf{y}$ is computed as

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T \mathbf{M}_{\text{avg}} (\mathbf{x}-\mathbf{y})} \tag{4}$$

where $\mathbf{M}_{\text{avg}} = \big(\mathbf{M}(\mathbf{x}) + \mathbf{M}(\mathbf{y})\big)/2$. This is equivalent to measuring distance in a *normalized space* where the ellipse is mapped to a unit circle using the rotation and scaling transformation $\boldsymbol{\Lambda}^{1/2}\mathbf{R}^T$. A more rigorous definition of distance would require integration [22, p. 30]; by assuming the metric to be locally constant, we achieve the less expensive formula above [4, 3]. In a mesh conforming to this metric, the neighbors of node $i$ lie close to the ellipse $d(\mathbf{p}_i, \mathbf{y}) = 1$. They lie exactly on the ellipse if the metric is locally constant.

The normalized area of a triangle defined by three points $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ is computed as

$$A(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \frac{1}{2}\sqrt{\det(\mathbf{M}_{\text{avg}})}\ \ (\mathbf{y}-\mathbf{x}) \times (\mathbf{z}-\mathbf{x}) \tag{5}$$

where $\mathbf{M}_{\text{avg}} = \big(\mathbf{M}(\mathbf{x}) + \mathbf{M}(\mathbf{y}) + \mathbf{M}(\mathbf{z})\big)/3$, and $\mathbf{u} \times \mathbf{v}$ is the 2-D cross product with scalar value $u_1 v_2 - u_2 v_1$.

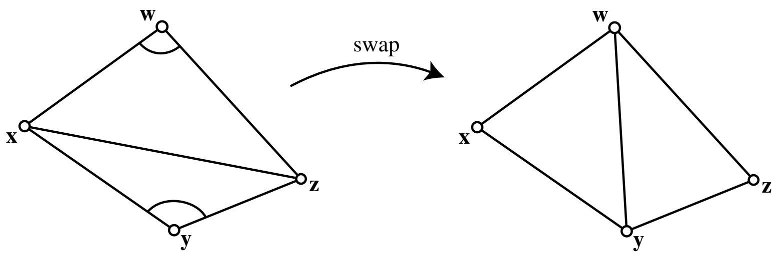



Figure 9: Edge swapping

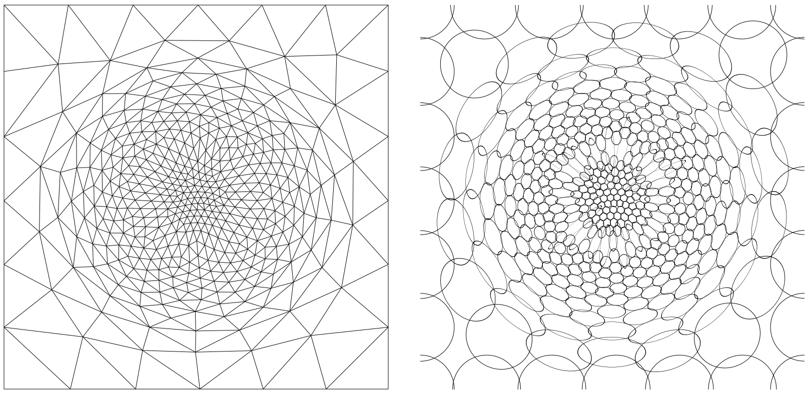
### 4.1 Modified Delaunay Criterion

We modify the Delaunay criterion to take anisotropy into account. Let **xyz** and **zwx** be two adjacent triangles (Figure 9). Delaunay retriangulation swaps edges to maximize the minimum angle. Equivalently, if $\angle\mathbf{zyx} + \angle\mathbf{xwz} > 180^\circ$ then swap edge **xz** for edge **yw** [2]. For anisotropic Delaunay triangulation, we measure angles in the normalized space defined by the metric. The generalized rule is to swap edge **xz** for **yw** if

$$\big[(\mathbf{z}-\mathbf{y}) \times (\mathbf{x}-\mathbf{y})\big]\ (\mathbf{x}-\mathbf{w})^T \mathbf{M}_{\text{avg}} (\mathbf{z}-\mathbf{w})\ +\ (\mathbf{z}-\mathbf{y})^T \mathbf{M}_{\text{avg}} (\mathbf{x}-\mathbf{y})\ \big[(\mathbf{x}-\mathbf{w}) \times (\mathbf{z}-\mathbf{w})\big]\ <\ 0 \tag{6}$$

This test is equivalent to performing the well known circumcircle test in the normalized space, but requires fewer arithmetic operations [3]. When inserting, $\mathbf{M}_{\text{avg}} = \big(\mathbf{M}(\mathbf{w}) + \mathbf{M}(\mathbf{x}) + \mathbf{M}(\mathbf{y}) + \mathbf{M}(\mathbf{z})\big)/4$, where $\mathbf{w}$, $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are the four vertices of the quadrilateral being checked, and when deleting node $i$, $\mathbf{M}_{\text{avg}} = \mathbf{M}(\mathbf{p}_i)$ for all swap tests.

When the metric tensor is constant, the edge swapping process finds a global optimum [11]. When it is not constant, there is potential for infinite looping if the metric is evaluated at several points, because these metrics could be inconsistent. We avoid this possibility simply by using the same tensor for all of the swap tests following a deletion.



a) Mesh, 430 nodes b) Ellipse around each node

Figure 10: Anisotropic approximation of a Gaussian

Figure 10a shows an anistropic mesh for approximating the Gaussian $e^{-\mathbf{x}^2/2}$. It was constructed using a metric tensor equal to the Hessian of the Gaussian, with eigenvalues replaced by their absolute values. In Figure 10b, the ellipses $d(\mathbf{p}_i, \mathbf{y}) = 1/2$ around each node $i$ are drawn, as a visualization of the element size function. Note than in regions where the metric tensor is changing rapidly, ellipses may overlap.

## 5 Results

The algorithm described has been implemented in C++. Given a 2-dimensional domain bounded by line segments, and a function specifying the desired element size, our program generates an unstructured triangular mesh. In this section, we present some results.

Figure 11 depicts the result when the element size function specifies small elements near the diagonal. The angle and normalized edge length histograms show that a majority of elements are close to optimality (i.e. 60 degree angles and unit normalized length).

A more complex mesh representing an idealized hyperbolic shock front is shown in figures 13 and 14. The aspect ratio of the triangles grows towards the center of the mesh, where it reaches 100:1. The histograms demonstrate that the normalized angles [22, eq. (8.19)] are close to 60 degrees and the elements are close to the desired size. Figure 14 shows that even in the region of maximum aspect ratio, the mesh is very regular and the elements are well shaped.

Figure 12 shows the time cost as a function of the number of nodes for the mesh of Figure 10, running on an SGI Indigo2 with 250 MHz MIPS R4400 processor. From empirical tests, the behavior is approximately linear. Although convergence is not guaranteed, it is achieved in the vast majority of cases. Problematic cases are generated by inconsistencies between the desired edge size and the geometry (small features in the geometry and large desired edge size), and abrupt changes in the metric. In these situations, our assumption that the metric is locally constant breaks down. Further study of the conditions on validity of element size functions is needed.

Figure 15 illustrates a more complex domain created from outline fonts. The element size function, chosen for illustration purposes, controls grading according to distance from input boundary nodes, and anisotropy according to distance from a curve.

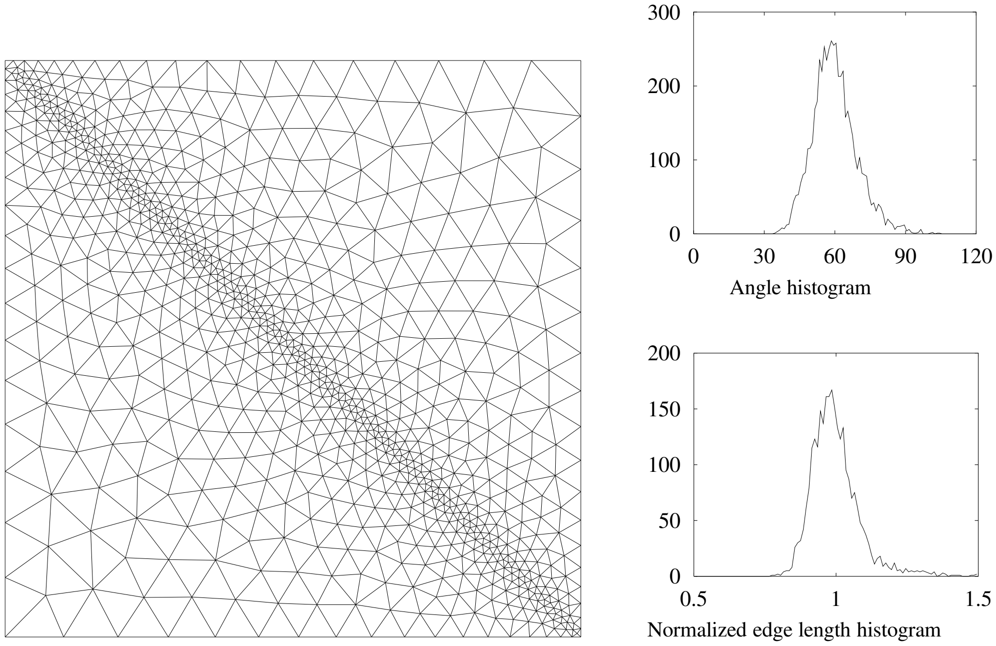



Figure 11: Isotropic mesh, 925 nodes

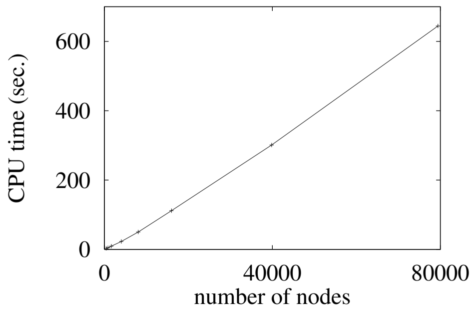



Figure 12: Computation time to generate meshes

## 6 Conclusions

We have presented a new method for two dimensional, unstructured triangular mesh generation that is capable of producing quality meshes on complex domains. Depending on the application, meshes that are uniform or graded, and isotropic or anisotropic can be generated. The pliant approach, in which nodes are smoothed, inserted, and deleted in a loop, gives the algorithm the flexibility to fit an appropriate number of elements in each region of the domain, resulting in well shaped elements. In contrast to hybrid structured/unstructured mesh generation techniques, our method can generate complex meshes with specified element shape using a single, simple algorithm.

The inputs to the algorithm are general enough to permit application-independent modularization of the mesh generator. The desired edge length is specified through an element size function that is provided as part of the input. This element size function can come from the boundary geometry, from an adaptive FEM solver, or from the user. It can change over time.

The method appears particularly well suited for several applications. In viscous flow simulations, the anisotropic capabilities of the algorithm are very useful. For adaptive FEM simulations, the flexible refinement and coarsening provided by the algorithm can obviate remeshing. These properties make the method well suited to shape optimization and large deformation simulations as well.

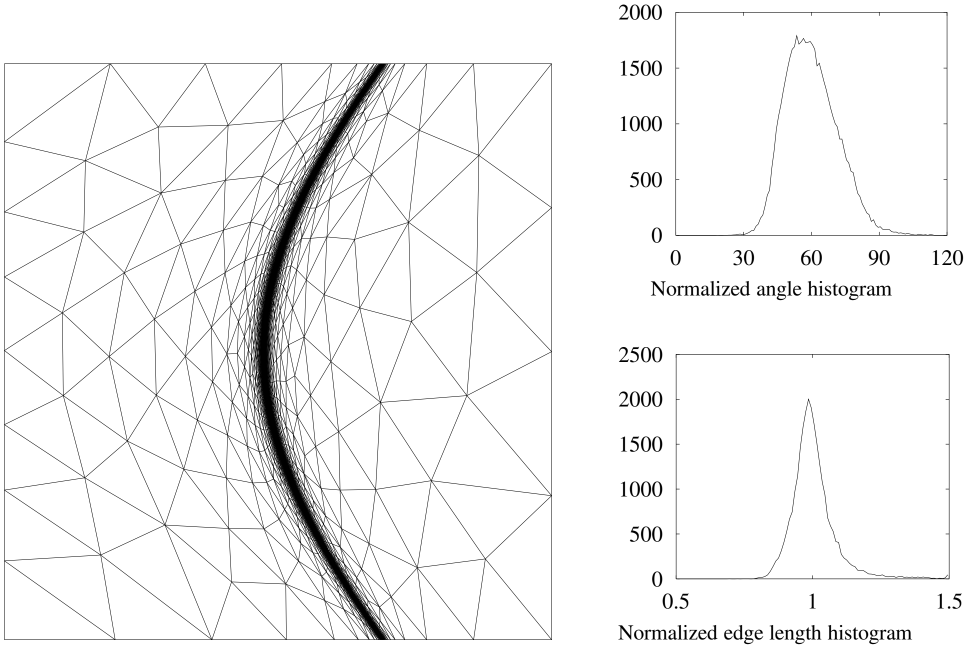



Figure 13: Idealized shock front, 8712 nodes

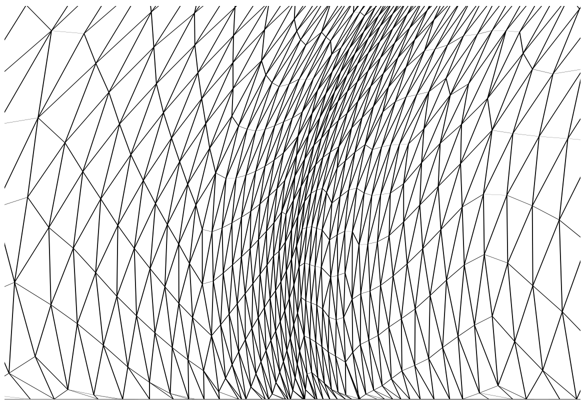


Figure 14: Idealized shock front closeup (150× horizontal and 15× vertical zoom)

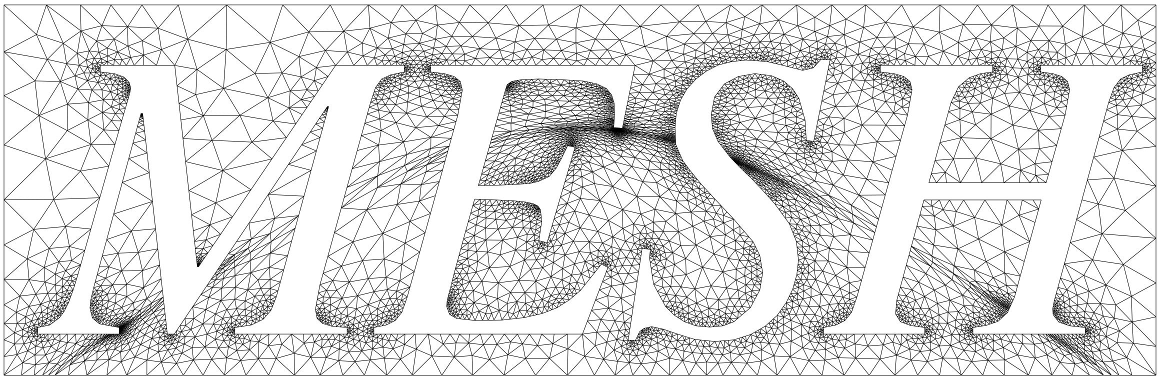


Figure 15: Anisotropic mesh for a complex domain, 5523 nodes

Although we have not done side-by-side empirical tests, we make some preliminary comparisons with previous mesh generators. Relative to the methods of Shimada-Gossard and Ruppert [20, 18], our method is more general by virtue of supporting anisotropy, and it appears to be faster than the former. Compared to advancing front methods: our algorithm may be better able to generate highly stretched meshes than the method of Peraire and Peiró [16], it is simpler than the anisotropic techniques of Mavriplis [14], and it is simpler than, and it supports more general element size functions than, Marcum's algorithm [12]. Our technique is probably most similar to those of Castro-Díaz *et al.* [4] and Vallet *et al.* [23, 7]. A subjective visual comparison suggests that our meshes are more "attractive" than all of the unstructured methods cited with the exception of Shimada and Gossard's, with which it is on par. (But this does not imply that our meshes are better numerically, of course.) On the negative side, our algorithm is currently limited to 2-D triangles, and it is currently slower than some previous methods (e.g. [18, 12]).

There are several areas for future work. The algorithm could probably be sped up. For example, when lower mesh quality is acceptable, the algorithm could do less smoothing. We would like to find a provably convergent variant of the algorithm, possibly using modified rules for smoothing, insertion, and deletion. The method should be tested empirically with an FEM solver and its results compared to other mesh generators. The technique appears to generalize naturally to surfaces and volumes in three dimensions, and we suspect that it will generate high quality meshes there, as well, even though tetrahedrization is topologically much more complex than triangulation.

We plan to release source code for our algorithm on the Web at some future date.

## Acknowledgments

We thank Scott Canann, Omar Ghattas, Marshall Bern, and Jim Ruppert for their comments, and Paul Haeberli for font software. This work was supported by NSF Young Investigator award CCR-9357763.

## References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Clarendon Press, Oxford, 1987.

[2] Marshall Bern and Paul Plassmann. Mesh generation. In Jörg Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*. Elsevier Science, to appear.

[3] Frank J. Bossen. Anisotropic mesh generation with particles. Technical Report CMU-CS-96-134, CS Dept., Carnegie Mellon University, May 1996. http://ltswww.epfl.ch/∼bossen/.

[4] M.J. Castro-Díaz, F. Hecht, and B. Mohammadi. New progress in anisotropic grid adaptation for inviscid and viscous flows simulations. In *4th Annual Intl. Meshing Roundtable*, Oct. 1995. http://www.ce.cmu.edu/∼sowen/Roundtable.agenda.html.

[5] Eduardo F. D'Azevedo. Optimal triangular mesh generation by coordinate transformation. *SIAM J. Sci. Stat. Comput.*, 12(4):755–786, July 1991.

[6] David A. Field. Laplacian smoothing and Delaunay triangulations. *Comm. Applied Numer. Meth.*, 4:709–712, 1988.

[7] Michel Fortin, Marie-Gabrielle Vallet, Julien Dompierre, Yves Bourgault, and Wagdi G. Habashi. Anisotropic mesh adaptation: Theory, validation and applications. In *Third ECCOMAS Computational Fluid Dynamics Conf.*, Paris, Sept. 1996. To appear.

[8] William H. Frey. Selective refinement: a new strategy for automatic node placement in graded triangular meshes. *Intl. J. Numer. Meth. Eng.*, 24:2183–2200, 1987.

[9] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. on Graphics*, 4(2):74–123, April 1985.

[10] Peter Hansbo. Generalized Laplacian smoothing of unstructured grids. *Comm. Numer. Meth. Eng.*, 11:455–464, 1995.

[11] Charles L. Lawson. Software for $C^1$ surface interpolation. In John R. Rice, editor, *Mathematical Software III*, pages 161–194. Academic Press, 1977.

[12] David L. Marcum. Generation of unstructured grids for viscous flow applications. In *33rd AIAA Aerospace Sciences Mtg.*, Reno, NV, Jan. 1995. AIAA paper 95-0212.

[13] Dimitri J. Mavriplis. Adaptive mesh generation for viscous flows using Delaunay triangulation. *J. of Computational Physics*, 90(2):271–291, Oct. 1990.

[14] Dimitri J. Mavriplis. Unstructured mesh generation and adaptivity. Technical Report ICASE 95-26, NASA Langley, Hampton VA, Apr. 1995. Abstract at http://techreports.larc.nasa.gov/cgi-bin/NTRS.

[15] Peter Möller and Peter Hansbo. On advancing front mesh generation in three dimensions. *Intl. J. Numer. Meth. Eng.*, 38:3551–3569, 1995.

[16] J. Peraire and J. Peiró. Adaptive remeshing for three-dimensional compressible flow computations. *J. of Computational Physics*, 103:269–285, 1992.

[17] Shmuel Rippa. Long and thin triangles can be good for linear interpolation. *SIAM J. Numer. Anal.*, 29(1):257–270, Feb. 1992.

[18] Jim Ruppert. A new and simple algorithm for quality 2-dimensional mesh generation. In *4th ACM-SIAM Symp. on Discrete Algorithms*, pages 83–92, 1993.

[19] Kenji Shimada. *Physically-Based Mesh Generation: Automated Triangulation of Surfaces and Volumes via Bubble Packing*. PhD thesis, ME Dept., MIT, 1993.

[20] Kenji Shimada and David C. Gossard. Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing. In *Third Symp. on Solid Modeling and Appls.*, pages 409–419, May 1995.

[21] R. Bruce Simpson. Anisotropic mesh transformations and optimal error control. *Applied Numer. Math.*, 14(1-3):183–198, 1994.

[22] Tracy Y. Thomas. *Concepts from Tensor Analysis and Differential Geometry*. Academic, New York, 1965.

[23] Marie-Gabrielle Vallet, Julien Dompierre, Yves Bourgault, Michel Fortin, and Wagdi G. Habashi. Coupling flow solvers and grids through an edge-based adaptive grid method. In *ASME Fluids Eng. Conf.*, San Diego, CA, July 1996. Also CERCA report R96-1, http://www.cerca.umontreal.ca/∼dompiere/postscript/postscript.html.

[24] William Welch. *Serious Putty: Topological Design for Variational Curves and Surfaces*. PhD thesis, CS Dept, Carnegie Mellon University, Dec. 1995. CMU-CS-95-217, ftp://reports.adm.cs.cmu.edu/usr/anon/1995/CMU-CS-95-217A.ps, 217B.ps, 217C.ps.

[25] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *SIGGRAPH 94 Proceedings*, pages 269–277, July 1994. http://www.cs.cmu.edu/∼ph.

[26] O.C. Zienkiewicz and J.Z. Zhu. Adaptivity and mesh generation. *Intl. J. Numer. Meth. Eng.*, 32:783–810, 1991.