International Journal of Computational Geometry & Applications
Vol. 10, No. 4 (2000) 417–440

ANISOTROPIC TRIANGULATION OF PARAMETRIC SURFACES VIA CLOSE PACKING OF ELLIPSOIDS

KENJI SHIMADA*
Mechanical Engineering, Carnegie Mellon University
Pittsburgh, PA 15213, USA
E-mail: shimada@cmu.edu
http://ciel.me.cmu.edu/shimada/

and

ATSUSHI YAMADA and TAKAYUKI ITOH
IBM Research, Tokyo Research Laboratory, 1623-14, Shimo-tsuruma
Yamato-shi, Kanagawa-ken 242-8502, Japan

Received 15 June 1998
Revised 1 February 1999
Communicated by S. H. Teng, Guest Editor

ABSTRACT

This paper describes a new computational method of fully automated anisotropic triangulation of a trimmed parametric surface. Given as input: (1) a domain geometry and (2) a 3×3 tensor field that specifies a desired anisotropic node-spacing, this new approach first packs ellipsoids closely in the domain by defining proximity-based interacting forces among the ellipsoids and finding a force-balancing configuration using dynamic simulation. The centers of the ellipsoids are then connected by anisotropic Delaunay triangulation for a complete mesh topology. Since a specified tensor field controls the directions and the lengths of the ellipsoids' principal axes, the method generates a graded anisotropic mesh whose elements conform precisely to the given tensor field.

Keywords: Unstructured mesh, anisotropy, parametric surface, metric tensor, Delaunay triangulation.

1. Introduction

Triangulating parametric surfaces is an essential task in many computer applications, including finite element method (FEM) and computer graphics. The original challenge in surface triangulation was to create a well-shaped graded mesh, that is, to generate triangles as equilateral as possible with the element size distribution conforming to a given metric. There are many algorithms proposed for this isotropic graded meshing problem, and some of them are commercially available.

The isotropic meshing problem has thus been mostly solved, but a new challenge now is to triangulate a surface into an anisotropic mesh, a mesh stretched in a given

*Author for correspondence.

direction. An anisotropic mesh is often advantageous in terms of computational cost and solution accuracy when physical phenomena on a surface or a geometric property of a surface has strong directionality.

For example, in fluid dynamics simulation using FEM, we prefer an anisotropic mesh stretched along shock/boundary layers and stream lines because a fewer mesh elements are required than in isotropic meshing to obtain the same level of solution accuracy. When simulated phenomena have strong directionality, as in fluid dynamics, an anisotropic mesh is more efficient in terms of computational time and solution accuracy than an isotropic mesh. The desired anisotropy of a mesh, however, is usually unknown prior to the analysis, and thus we have to perform adaptive remeshing (refinement and coarsening) based on the analysis result.

Another situation where we prefer an anisotropic mesh is when we approximate a curved surface by piecewise linear triangular elements. It is efficient to use an anisotropic mesh whose edge sizes are adjusted according to the directions of the principal curvatures. In order to equidistribute an approximation error, the edge length should be longer in a low curvature direction, and shorter in a high curvature direction.

In this paper we propose a versatile computational method of automatically generating an anisotropic graded triangular mesh of a trimmed parametric surface, applicable to FEM analyses and surface approximations.

Let us first define our problem of anisotropic triangulation of a parametric surface. We assume that an anisotropy is given as a 3×3 tensor field defined over a domain to be meshed. The surface triangulation problem is then stated as follows:

Given:

- a geometric domain on a parametric surface $\mathbf{S}(u, v)$ trimmed by trimming curves $\mathbf{C}_t(s)$
- inside curves $\mathbf{C}_i(s)$ and vertices $\mathbf{V}$ on which nodes are exactly located
- a desired anisotropic node spacing distribution, given as a 3×3 tensor field $\mathbf{M}(\mathbf{x})$

Generate:

- an anisotropic graded triangular mesh that is compatible with trimming curves, inside curves, and inside vertices

In the above problem statement, each surface patch is defined as a mapping, denoted as $\mathbf{S}(u, v) = \big(x(u, v), y(u, v), z(u, v)\big)$, from a rectangular region called *parametric space* into a 3D coordinate system called *object space*. A surface patch can be trimmed by restricting the rectangular region to a subset called the *trimmed region*, and its boundary curves are called *trimming curves*, denoted as $\mathbf{C}_t(s) = \big(x(s), y(s), z(s)\big)$. Occasionally we need to define extra curves and vertices inside the trimmed region so that some nodes are exactly located on those geometric elements. These curves and vertices are referred to as *inside curves* and *inside vertices*, denoted as $\mathbf{C}_i(s) = \big(x(s), y(s), z(s)\big)$ and $\mathbf{V} = (x, y, z)$ respectively. The actual curve

and surface representations can be of any form, as long as they are continuous and tangent vectors can be calculated anywhere on the curves and surfaces.

In order to generate an anisotropic triangular mesh over a given trimmed parametric surface, we modified and extended the *bubble mesh* method that we previously proposed for isotropic meshing.[9,11,12] The original bubble meshing procedure consists of two steps: (1) pack an appropriate number of spheres, or bubbles, closely in the domain, while the sizes of the spheres are adjusted based on a specified node spacing scalar field, and (2) connect the bubbles' centers by constrained Delaunay triangulation to generate node connectivity. The novelty of this method is that the close packing of bubbles mimics a pattern of Voronoi regions that yield well-shaped triangles and tetrahedra. Although the original bubble mesh using sphere packing creates a well-shaped, graded triangular or tetrahedral mesh, its application is limited to isotropic meshing because the close packing of spheres, or isotropic cells, naturally generates an isotropic node distribution.

In this paper, to apply the bubble mesh concept to anisotropic meshing of parametric surfaces, we assume as input a 3×3 tensor field that specifies the desired anisotropy of a mesh. With this tensor field, a spherical bubble is deformed to an ellipsoid whose directions and lengths of the principal axes are calculated from the eigenvectors and eigenvalues of the tensor respectively. By packing ellipsoidal bubbles closely in the domain, a set of nodes is distributed so that a graded, anisotropic triangular mesh is formed when the nodes are connected by anisotropic Delaunay triangulation.

The remainder of the paper is organized as follows. After reviewing related previous work (Section 2), we give an overview of our triangulation procedures (Section 3). We then give the derivation of a metric tensor for parametric surfaces (Section 4), physically-based bubble packing (Section 5), and anisotropic Delaunay triangulation (Section 6). Finally we show generated anisotropic meshes (Section 7) and close with discussion and conclusions (Section 8).

2. Related Work

In this section we review related work on three technical issues: (1) the usage of a tensor matrix for representing anisotropy; (2) interacting particles; and (3) the original bubble mesh for isotropic meshing.

2.1. Metric Tensor for Representing Anisotropy

One common way to represent an anisotropy is to define a metric tensor field, $\mathbf{M}$, over the domain.[4,3,2,1] $\mathbf{M}$ is a symmetric positive-definite 2×2 matrix in two dimensional problems, and a symmetric positive-definite 3×3 matrix in three dimensional problems. Castro-Díaz et al. show how a metric tensor can be defined so that it improves the quality of the adapted meshes in flow computations with multi-physical interactions and boundary layers.[4] Bossen and Heckbert use as input a 2×2 metric tensor in generating a 2D planar anisotropic triangular mesh using a system of interacting particles.[3] Borouchaki et al. demonstrate how a metric tensor

can be used to generate an anisotropic triangular mesh on a surface and to convert it to a quadrilateral mesh.[2] Shimada uses a 2D vector field equivalent to a 2×2 tensor for 2D anisotropic meshing.[10]

In this paper we also use the same metric tensor to control the size and the shape of an ellipsoid to be packed in a domain. Note that this tensor matrix is used not only to specify anisotropy of a mesh but also to dictate the element size distribution of the mesh.

2.2. Interacting Particles

A particle system is a collection of particles that moves over time according to either a deterministic or a stochastic set of rules or equation of motion. In computer graphics, a particle system was originally used to model and render natural fuzzy phenomena such as fog, smoke, and fire.[8] While early particle systems had little or no interparticle interaction, particle systems with proximity-based force interaction are recently used for different purposes, including Turk's re-tiling of a polygonal surface,[15] Szeliski's surface modeling,[14] de Figueiredo et al.'s polygonization of implicit surfaces, [5] Witkin and Heckbert's sampling and controlling of implicit surfaces,[16] and Fleischer et al.'s texture generation.[6]

These interacting particle systems use either repelling only or repelling and attracting forces among particles. If the magnitude and range of the force are uniform, the system creates a uniform distribution of particles yielding a hexagonal arrangement. Uneven, or graded, distribution can also be obtained by adjusting the magnitude and the range of the interparticle forces.

Bossen and Heckbert apply an interacting particle system to 2D anisotropic FEM mesh generation.[3] The method uses a 2×2 metric tensor to specify an anisotropy in a planar region, similar to Castro-Diaz's,[4] and generates a anisotropic node distribution using a proximity-based force similar to Shimada's.[9,11,12] This approach seems to be successful and to create a high-quality anisotropic 2D mesh. In terms of the definition of the interacting force, it is most similar to our bubble mesh in the 2D case.

2.3. Bubble Mesh

The bubble system is similar to the particle systems used in computer graphics in the sense that discrete bodies interact in 3D space as a result of the application of pairwise, repulsive/attractive forces. However, there are several unique characteristics that make this method particularly suitable for FEM mesh generation:

- A bubble system can triangulate a curved domain, a planar domain, a surface domain, a volumetric domain, and a hybrid of these domains (a non-manifold geometry) in a consistent manner. Bubbles are packed in order of dimension, i.e., vertices, edges, faces, and volumes, easily identified in CAD data. (See Figure 1.)
- Unlike some early particle systems for rendering, particle motion and its dynamic simulation themselves are not the focus. The model and the numerical

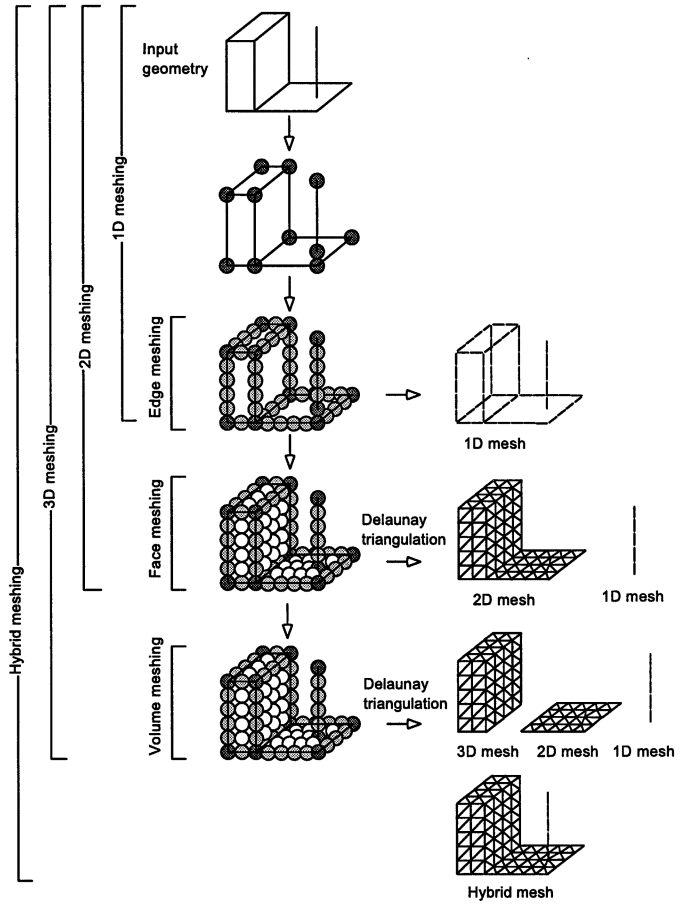


Fig. 1. Bubble meshing procedures.

solution of a bubble system are devised specifically to minimize the computational time necessary for reaching a force-balancing configuration.

- A quick initial guess at the final bubble configuration is obtained by using hierarchical spatial subdivision. This reduces the computational time necessary for the normally time-consuming process of dynamic simulation, or physically based relaxation.

- Unlike in a system of uniform particles, bubble diameters are adjusted individually by the node-spacing function. This makes precise control of triangle size possible throughout the mesh.

- A population control mechanism is used during relaxation to remove any superfluous bubble that is largely overlapped by its neighbors, and to subdivide any lone bubble missing some neighbors, so that a given domain is filled with an appropriate number of bubbles. This automatic feature drastically reduces the time necessary for the system to converge to a force-balancing configura-

tion.

In the original bubble mesh, spheres are closely packed to create isotropic meshes in 1D, 2D, surface, and 3D domains.[9,11,12] The method was later extended to generate a 2D planar anisotropic mesh by packing ellipsoids instead of spheres and modifying a circumcircle test in Delaunay triangulation.[10] In this paper, we demonstrate that the same idea of packing ellipsoids can be applied to anisotropic meshing of a trimmed parametric surface. The proposed surface meshing can be used as a subprocess of 3D and non-manifold meshing, and anisotropic meshing of such volumetric domains can be performed by the same ellipsoidal bubble packing. (Simply replace the spheres in Figure 1 by ellipsoids.)

3. Basic Approach

The novelty of our anisotropic meshing lies in the process of packing ellipsoidal bubbles closely in a domain. We achieve this close packing configuration by defining a proximity-based interbubble force and then solving the equation of motion numerically to yield a force-balancing configuration.

The first part of this section discusses the order of packing ellipsoidal bubbles, and the second part describes how the shapes and sizes of the bubbles are specified using 3×3 tensor matrices.

3.1. Triangulation Procedures

In order to obtain anisotropic node locations, ellipsoidal bubbles are packed on geometric entities in order of dimension as shown in Figure 2, that is:

1. Bubbles are placed on all the vertices $\mathbf{V}$, including inside vertices, as well as the endpoints of trimming curves and inside curves.
2. Bubbles are packed on all the trimming curves $\mathbf{C}_t$ and inside curves $\mathbf{C}_t$.
3. Bubbles are placed inside the trimmed region of the surface $\mathbf{S}$.

In the above process it is essential that we first place bubbles on geometric entities of lower dimension. In this way two fixed bubbles are already placed at the two endpoints of a curve when bubbles are packed on the curve, and these two bubbles are stable throughout the packing process, preventing moving bubbles from escaping the range of the curve. Similarly, when bubbles are packed inside the trimmed region of a surface, the trimming curves are already filled with fixed bubbles, preventing moving bubbles from escaping the trimmed region. In other words we put higher priority on the bubble placement of lower dimensional elements, i.e., vertex bubbles over edge bubbles, and edge bubbles over face bubbles. This strategy makes sense because lower order geometric elements are often more critical in FEM analysis.

Once all the bubbles are packed so that they cover the entire surface domain without significant gaps and overlaps, their centers are connected by *anisotropic Delaunay triangulation* for a complete mesh topology. The anisotropic Delaunay triangulation is detailed in Section 6.

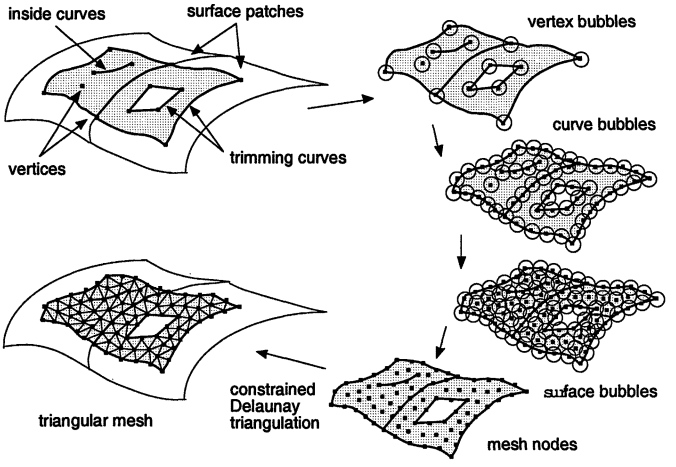


Fig. 2. Ellipsoidal bubble packing procedure.

3.2. Ellipsoidal Bubbles

As in previous work of anisotropic mesh generation,[4,3,2] we assume as input a symmetric positive-definite 3×3 metric tensor field $\mathbf{M}(\mathbf{x})$ that represents a desired anisotropy. We then use this metric tensor to specify the shapes and sizes of the ellipsoidal bubbles packed in 3D space. Such a 3×3 tensor matrix can be characterized by three eigenvalues λ_i, $i = 1, 2, 3$ and three eigenvectors $\mathbf{v}_i$, $i = 1, 2, 3$.

The eigenvalues define the inverse squares of the radii of the major, medium, and minor radii of the ellipsoidal bubble, and they are calculated by solving the equation

$$\det|\mathbf{M} - \lambda\mathbf{I}| = 0. \tag{1}$$

After the eigenvalues λ_i are determined, the eigenvectors $\mathbf{v}_i$ can be found by solving the equation

$$\mathbf{M}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, 2, 3. \tag{2}$$

The three eigenvectors are thus expressed as

$$\mathbf{v}_i = \lambda_i \mathbf{e}_i, \quad i = 1, 2, 3, \tag{3}$$

where $\mathbf{e}_i$, $i = 1, 2, 3$ are unit vectors in the directions of the eigenvectors $\mathbf{v}_i$, $i = 1, 2, 3$. These unit vectors are mutually orthogonal, and they are used to define the directions of the major, medium, and minor axes[a] of an ellipsoidal bubble.

Given unit vectors $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$, of the major, medium, and minor axes of an ellipsoid, and the diameters d_1, d_2, and d_3, along these axes, a 3×3 metric tensor

[a]In some computational mechanics applications, particularly in the study of materials, these axes are referred to as the principal axes of the tensor and they are physically important. For example, if the tensor is a stress tensor, the principal axes are the directions of normal stress with no shear stress.

is written as

$$\mathbf{M} = \mathbf{R}\begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}\mathbf{R}^T = \mathbf{R}\begin{pmatrix} (d_1/2)^{-2} & 0 & 0 \\ 0 & (d_2/2)^{-2} & 0 \\ 0 & 0 & (d_3/2)^{-2} \end{pmatrix}\mathbf{R}^T, \tag{4}$$

where

$$\mathbf{R} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{pmatrix} = \begin{pmatrix} e_{1x} & e_{2x} & e_{3x} \\ e_{1y} & e_{2y} & e_{3y} \\ e_{1z} & e_{2z} & e_{3z} \end{pmatrix}, \tag{5}$$

and d_i, $i = 1, 2, 3$ are the ellipsoid's diameters along the principal axes. The size and the shape of an ellipsoidal bubble is thus given as a function of its center position using the above 3×3 tensor field.

4. Metric Tensor for Parametric Surfaces

Although we need a 3×3 metric tensor field $\mathbf{M}(\mathbf{x})$ to specify the size and shape of an ellipsoid, a desired anisotropy is often given by a 2×2 metric tensor field defined in either parametric space or object space. A good example of such a case is when a surface is triangulated based on its curvature. Hence it is important that we discuss the following two issues in this section:

- How to find a corresponding ellipse, or a 2×2 tensor in parametric space, when an anisotropy is defined by a 2×2 tensor in object space. This is necessary for anisotropic Delaunay triangulation in parametric space.
- How to define an ellipsoid, or a 3×3 tensor, in object space, necessary for force calculation, when only a 2×2 tensor in object space is given as input.

The two issues are discussed in the remaining of this section, Section 4.1 and Section 4.2 respectively.

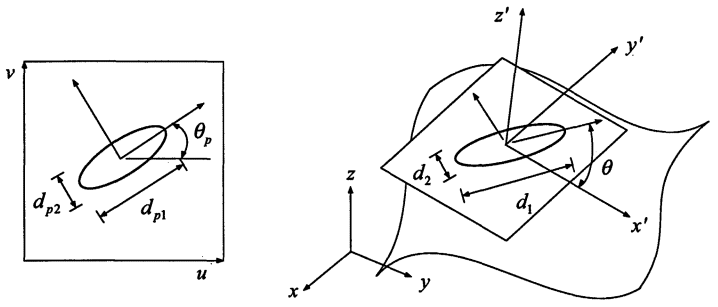
4.1. Finding Ellipses in Parametric Space

Given a point on a surface, we can calculate two tangent vectors in the u direction and v direction, $\frac{\partial \mathbf{S}}{\partial u}$ and $\frac{\partial \mathbf{S}}{\partial v}$ respectively. We then define a local coordinate sytem $x'y'z'$ in such a way that: (1) the x'-axis is parallel to $\frac{\partial \mathbf{S}}{\partial u}$; (2) the z'-axis is parallel to the normal direction $\frac{\partial \mathbf{S}}{\partial u} \times \frac{\partial \mathbf{S}}{\partial v}$; and (3) the y'-axis is parallel to the cross product of the z'-axis and the x'-axis (See Figure 3(b)).

A 2×2 metric tensor $\mathbf{M}_{x'y'}$ represents an ellipse in object space lying on the tangent plane $x'y'$, and this tensor can be expressed as

$$\mathbf{M}_{x'y'} = \mathbf{R}_2(\theta)\begin{pmatrix} (d_1/2)^{-2} & 0 \\ 0 & (d_2/2)^{-2} \end{pmatrix}\mathbf{R}_2(\theta)^T, \tag{6}$$

where θ measures an angle between the x'-axis and the major axis of the ellipse, and d_1 and d_2 are diameters along the major axis and the minor axis.



(a) uv parametric space (b) xyz object space

Fig. 3. Tensor ellipse on a tangent plane.

To find a corresponding ellipse in parametric space, we first find the following 2×2 matrix $\mathbf{A}$ that transforms the $x'y'$ coordinate system to the uv coordinate system,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{A}\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \left\|\frac{\partial \mathbf{S}}{\partial u}\right\| & \left\|\frac{\partial \mathbf{S}}{\partial v}\right\| \cos(\theta_w) \\ 0 & \left\|\frac{\partial \mathbf{S}}{\partial v}\right\| \sin(\theta_w) \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix}, \tag{7}$$

where θ_w measures an angle between the x'-axis and $\frac{\partial \mathbf{S}}{\partial v}$.

Using this coordinate transformation matrix $\mathbf{A}$ and the 2×2 metric tensor in object space $\mathbf{M}_{x'y'}$, the 2×2 metric tensor in parametric space $\mathbf{M}_{uv}$ can be obtained as

$$\mathbf{M}_{uv} = \mathbf{A}\mathbf{M}_{x'y'}\mathbf{A}^T = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}. \tag{8}$$

This is a 2×2 symmetric positive-definite matrix and hence $m_{12} = m_{21}$.

By calculating eigenvalues and eigenvectors, we can also express $\mathbf{M}_{uv}$ in the form

$$\mathbf{M}_{uv} = \mathbf{R}_2(\theta_p) \begin{pmatrix} (d_{p1}/2)^{-2} & 0 \\ 0 & (d_{p2}/2)^{-2} \end{pmatrix} \mathbf{R}_2(\theta_p)^T, \tag{9}$$

where θ_p measures an angle between the u-axis and the major axis of the ellipse in parametric space, and d_{p1} and d_{p2} diameters along the major axis and the minor axis in parametric space. The implicit form of this ellipse is

$$m_{11}u^2 + m_{22}v^2 + 2m_{12}uv = 1. \tag{10}$$

Now we have found how to calculate a corresponding 2×2 metric tensor, or an ellipse, in parametric space when a 2×2 tensor field is given on the surface in object space.

A particularly useful 2×2 metric tensor is one based on the curvature of a surface; a surface region of high curvature is meshed with fine triangles, and a region of low curvature with coarse triangles. The curvature changes depending on a cross-sectional plane perpendicular to the tangent plane, and two principal curvature directions can be identified. These two principal axes are orthogonal,

and they represent the directions of the maximum radius of curvature and the minimum radius of curvature. In order to equidistribute the approximation error, one can define d_1 and d_2 in Equation 6 as follows [9,12]

$$\begin{aligned} d_1 &= min\Big(2\sqrt{2e\rho_{max} - e^2}, D_{max}\Big), \\ d_2 &= min\Big(2\sqrt{2e\rho_{min} - e^2}, D_{max}\Big), \end{aligned} \tag{11}$$

where ρ_{min} and ρ_{max} denote the minimum radius of curvature and the maximum radius of curvature respectively, e a target constant error between the original surface and the mesh, and D_{max} the allowable maximum size of the diameter of an ellipsoid. Setting this maximum size is necessary because, when a surface is nearly flat in one direction, ρ_{max} approaches infinity, yielding an oversized mesh element.

4.2. 2 × 2 Metric Tensor in Parametric Space

As mentioned earlier in this section we also need to find how to define a 3 × 3 metric tensor, or a tensor ellipsoid, when only a 2 × 2 metric tensor is given on the surface. This is the process of expanding ellipses to ellipsoids by adding a diameter along the third axis. The process is essential because, as detailed later in Section 5, interbubble forces are calculated using ellipsoids defined by a 3 × 3 metric tensor.

To decide the diameter along the third axis, parallel to the normal to the surface, we compare the two diameters d_1 and d_2 along the two principal axes on the tangent plane $x'y'$ and give the smaller value to the diameter along the third axis. The 3 × 3 metric tensor is thus defined as

$$\mathbf{M}_{xyz} = \mathbf{R} \begin{pmatrix} (d_1/2)^{-2} & 0 & 0 \\ 0 & (d_2/2)^{-2} & 0 \\ 0 & 0 & \big(min(d_1, d_2)/2\big)^{-2} \end{pmatrix} \mathbf{R}^T. \tag{12}$$

Although there are other ways to define the ellipsoid diameter along the third axis, for example by taking the average of d_1 and d_2, we choose to use the minimum of d_1 and d_2 because a smaller diameter gives a smaller element size in the surface normal direction, which is advisable if the surface triangulation is used as a subprocess of three dimensional meshing.

5. Bubble Packing by Proximity-Based Forces

5.1. Interbubble Forces

In isotropic meshing the ideal node configuration is a regular hexagonal arrangement, a repeating pattern often observed in nature. One such example of a regular hexagonal arrangement is a molecular structure; the pattern is created by the van der Waals force, which exerts a repelling force when two molecules are located closer together than the stable distance and exerts an attracting force when two molecules

are located farther apart than the stable distance. One of the mathematical representations of this van der Waals force is

$$f(r) = 12Ar^{-13} - 6Br^{-7}, \tag{13}$$

where A and B are positive constants, and r is the distance between two points. The first term describes the repulsion force, and the second the attraction force.

Since the van der Waals force creates a regular layout of points, as observed in metal bonding, we could simply take one of the standard mathematical models of this force and implement it as the interbubble force field. This is not a good approach, however, because our goal is not the realistic simulation of the behavior of molecules, but is to find a force-balancing configuration efficiently. This is why we have devised the following simplified force model using a single piecewise cubic polynomial function.

Let the positions of adjacent bubbles i and j be $\mathbf{x}_i$ and $\mathbf{x}_j$; the current distance between the two bubbles $l(\mathbf{x}_i, \mathbf{x}_j)$; the target stable distance $l_0(\mathbf{x}_i, \mathbf{x}_j)$; the ratio of the current distance and the target distance $w(\mathbf{x}_i, \mathbf{x}_j) = \frac{l(\mathbf{x}_i, \mathbf{x}_j)}{l_0(\mathbf{x}_i, \mathbf{x}_j)}$; and the corresponding linear spring constant at the target distance k_0. Our simplified force model can then be written as

$$f(w) = \begin{cases} \frac{k_0}{l_0}\left(1.25w^3 - 2.375w^2 + 1.125\right) & 0 \le w \le 1.5 \\ 0 & 1.5 < w. \end{cases} \tag{14}$$

As shown in Figure 4, this force model applies either a repelling or attracting force between two bubbles based on the following distance comparison. Assuming that two bubbles are adjacent to each other, a repelling force is applied if l is smaller than l_0, or if $w < 1.0$. An attracting force is applied if l is larger than l_0, or if $1.0 < w < 1.5$. No force is applied if the two bubbles are located exactly at the stable distance or if they are located much farther apart, the cases where $w = 1.0$ or $1.5 < w$.

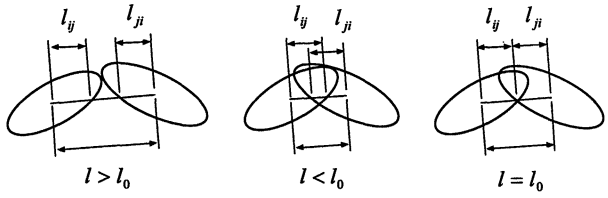
In the original isotropic bubble meshing, where two bubbles are spherical, the stable distance can be calculated simply as the sum of the radii of the two bubbles [9,11,12]

$$l_0(\mathbf{x}_i, \mathbf{x}_j) = \frac{d(\mathbf{x}_i)}{2} + \frac{d(\mathbf{x}_j)}{2}, \tag{15}$$

where $d(\mathbf{x}_i)$ and $d(\mathbf{x}_j)$ are the diameters of bubble i and bubble j respectively. If the two bubbles are ellipsoidal, however, this target stable distance should be calculated as the sum of the two lengths, measured along the line segment that connects the centers of the two ellipsoids, from the center to boundary of each ellipsoid (See Figure 4). Let these two lengths be l_{ij} and l_{ji}; the target stable distance l_0 is then given as

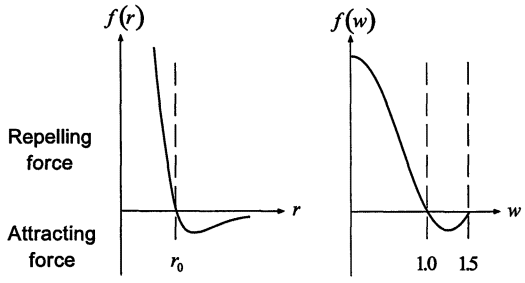
$$l_0(\mathbf{x}_i, \mathbf{x}_j) = l_{ij} + l_{ji}, \tag{16}$$

where l_{ij} is calculated with a relatively low computational cost by multiplying the tensor matrix $\mathbf{M}(\mathbf{x}_i)$ and a unit vector from $\mathbf{x}_i$ to $\mathbf{x}_j$, and l_{ji} is calculated similarly. Note that Equation 15 is a special case of Equation 16.



(a) Attracting (b) Repelling (c) Stable

Fig. 4. Target stable distance.



(a) van der Waals force (b) Implemented simplified force

Fig. 5. Interbubble proximity-based force.

Compared to the van der Waals force, our force, as shown in Figure 5, has the following two characteristics that make it suitable for our physically-based relaxation:

- The force is saturated near w=0, where two bubbles are located extremely close together. This prevents the interbubble force from growing infinitely large and causing numerical instability in dynamic simulation.
- The force interaction is active only within a specified distance and only when two bubbles are adjacent.

The second point is particularly important to reduce the single most time consuming process in physically-based node placement: the calculation of pairwise interaction forces. In our implementation, we run the anisotropic Delaunay triangulation, detailed in Section 3.5, every certain number of iterations in order to identify adjacent pairs of bubbles. Force is exerted, consequently, only on adjacent bubbles.

5.2. Physically-Based Mesh Relaxation

Given the proximity-based interbubble force, our goal of physically-based relaxation is to find a bubble configuration that yields a static force balance in a direction tangential to the surface. In other words, we want the summation of interbubble force vectors applied to a bubble to be parallel to the surface normal direction. This condition can be written as

$$\mathbf{f}_i \cdot \mathbf{n}_i = 0, \qquad i = 1, \ldots, n, \tag{17}$$

where $\mathbf{f}_i$ represents the total force on bubble i from all its adjacent bubbles, $\mathbf{n}_i$ the surface normal $\frac{\partial \mathbf{S}}{\partial u} \times \frac{\partial \mathbf{S}}{\partial v}$ at the location of the bubble center $\mathbf{x}_i$, and n the number of mobile bubbles.

Due to an arbitrarily defined tensor field and geometric constraints on bubble locations, Equation 17 is highly nonlinear, and thus it is difficult to solve the equation directly by a multidimensional root-finding technique such as the Newton-Raphson method. Our alternative approach is to assume a point mass m at the center of each bubble and the effect of viscous damping c, and to solve the following equation of motion[b] by using a standard numerical integration scheme, the fourth-order Runge-Kutta method.

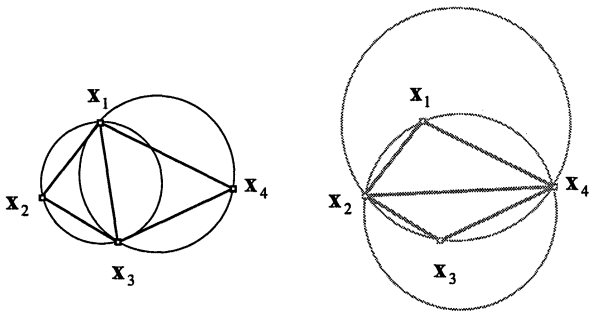
$$m\ddot{\mathbf{x}}_i(t) + c\dot{\mathbf{x}}_i(t) = \mathbf{f}_i(t), \qquad i = 1, \ldots, n. \tag{18}$$

There are two subprocesses that we need to perform in solving Equation 18 numerically: (1) imposing geometric constraints on bubble movements so that bubbles do not pop out of a curve or a surface; and (2) adjusting the bubble population so that a given surface domain is filled with an appropriate number of bubbles. The two subprocesses are briefly described below.

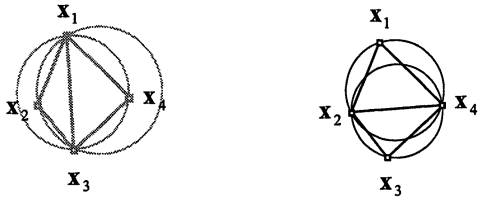
Imposing geometric constraints on bubble movements is important. Because a new bubble location obtained by the numerical integration of the equation of motion is not constrained to a curve or surface, we need to move the bubbles back onto the curve or the surface. We accomplish this in the following two steps: (1) calculate a corresponding displacement in parametric space by projecting the unconstrained displacement onto unit tangent vectors of the curve or the surface; and (2) map the displacement onto the curve or the surface. The basic approach to impose geometric constraints on bubble movements is common to both anisotropic meshing and isotropic meshing, and more details can be found elsewhere.[9,11,12]

Another process incorporated in solving the equation of motion is adaptive bubble population control. This is essential because we do not know beforehand an appropriate number of bubbles that is necessary and sufficient to fill a region. Although our initial bubble configuration generator gives a reasonably good guess, to produce an optimum number of bubbles in the domain we implemented a procedure of adaptively adding and deleting bubbles based on local population density. To identify areas that are either too sparse or too crowded we first perform triangulation when the system of bubbles becomes stable, and then we check the lengths of the triangle edges to see if they are close to the desired lengths specified by a given tensor field. An edge much longer than the desired length indicates that there is a significant gap around it and thus a new bubble is added at the middle point of the edge. Conversely, an edge much shorter than the desired length means that there is a significant overlap and thus a bubble needs to be deleted.

[b]The first order equation can also be used.[3] In either case, the essential point is that after a certain number of iterations the system reaches a virtual equilibrium, where both the velocity term $\dot{x}$ and the acceleration term $\ddot{x}$ approach zero, leaving a static force balance.



(a) Original circumcircle test will choose $\triangle x_1x_2x_3$ and $\triangle x_1x_3x_4$



(b) Anisotropic circumcircle test with $\widetilde{\mathbf{M}}_{uv} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$ will choose $\triangle x_1x_2x_4$ and $\triangle x_2x_3x_4$

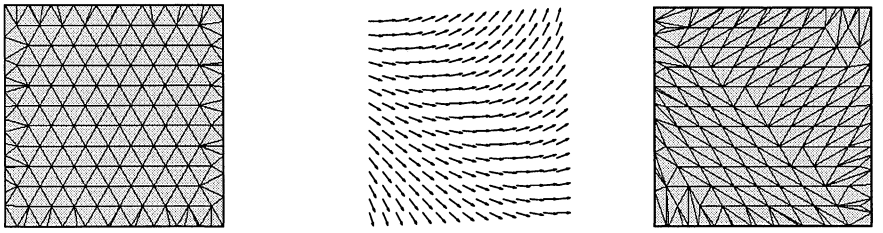
Fig. 6. Anisotropic circumcircle test.

6. Anisotropic Delaunay Triangulation

Once a force-balancing configuration of ellipsoidal bubbles is obtained, the bubbles' centers must be connected to form a complete triangular mesh. In connecting nodes, Delaunay triangulation is considered suitable for finite element analysis, as the triangulation maximizes the sum of the smallest angles of the triangles. It creates triangles as equilateral, or isotropic, as possible for the given set of points; thus thin, or anisotropic triangles are avoided whenever possible.

One important property of Delaunay triangulation is that a circumscribing circle of a Delaunay triangle, called a *circumcircle*, must not contain other points inside. To check this, many Delaunay triangulation algorithms use the so-called *circumcircle test.* This test is also used in Sloan's algorithm,[13] which we implemented in the original 2D isotropic bubble mesh. As shown in Figure 6, the circumcircle test is performed on a pair of adjacent triangles that forms a convex quadrilateral. Given such a set of four points, the circumcircle test checks one of the triangles to see whether the fourth point is inside the circumcircle. If it is, the four points are then reconnected into the other possible configuration of two triangles.

Obviously the original Delaunay triangulation with this circumcircle test is not suitable for our anisotropic meshing. We therefore modified the original Delaunay triangulation slightly to incorporate anisotropy in the circumcircle test. Assuming the metric tensor is locally constant, we perform the same circumcircle test in parametric space, but only after the four nodes' coordinate values have been transformed so that an ellipse is mapped back to a circle. A local average tensor for four nodes in parametric space can be determined by first calculating the barycenter of



(a) Isotropic triangulation (b) d_1 axis direction (c) Anisotropic triangulation, d_1=2 and d_2=1

Fig. 7. An example of anisotropic Delaunay triangulation.

the four nodes and then finding the metric tensor at this barycenter[c]

$$\widetilde{\mathbf{M}}_{uv} = \mathbf{M}_{uv}\left(\frac{\mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3 + \mathbf{x}_4}{4}\right). \tag{19}$$

Figure 6 shows a case where a different pair of triangles is selected when the circumcircle test is performed after the positions of the four nodes are transformed.

To demonstrate the effectiveness of this anisotropic Delaunay triangulation, Figure 7(a) and Figure 7(c) compare the original Delaunay triangulation and the anisotropic Delaunay triangulation. Given the same set of triangular grid nodes, the anisotropic Delaunay triangulation creates an anisotropic mesh that is stretched and "flows" along the direction of the major eigenvectors shown in Figure 7(b).

7. Results

The anisotropic meshing described above has been implemented in C and C++. Three meshing results are shown in Figures 9, 11, and 13, and their quality measures are shown in Figures 10, 12, and 14 respectively. Figure 8 summarizes the statistics of these three meshes, including: (1) the numbers of mesh nodes and elements; (2) CPU times for the initial meshing, intermediate meshing after 30 iterations of dynamic simulation, and the final meshing after 100 iterations of dynamic simulation; and (3) mesh irregularity after 100 iterations. The CPU time was measured on an IBM Unix workstation (PowerPC 604e, 133MHz).

To measure the mesh irregularity shown in Figure 10, Figure 12, Figure 14, and Figure 8, we used two types of irregularity measure, *topological irregularity* and *geometric irregularity.*

For topological irregularity, we defined the following measure, similar to that defined by Frey and Field:[7]

$$\varepsilon_t = \frac{1}{n}\sum_{i=0}^{n} |\delta_i - 6| \tag{20}$$

where δ_i represents the *degree*, or the number of neighboring nodes, connected to the ith interior node, and n represents the total number of interior nodes in the

[c]Slightly different anisotropic Delaunay triangulation schemes are used by other researchers.[4,3,2] For example, an alternative way to take an average of four metric tensors is: $\widetilde{\mathbf{M}}_{uv} = \frac{1}{4}\Big(\mathbf{M}_{uv}(\mathbf{x}_1) + \mathbf{M}_{uv}(\mathbf{x}_2) + \mathbf{M}_{uv}(\mathbf{x}_3) + \mathbf{M}_{uv}(\mathbf{x}_4)\Big)$.

Mesh	Nodes	Elements	CPU time initial mesh	CPU time 30 iterations*	CPU time 100 iterations	Mesh irregularity after 100 iterations	
Mesh 1	1468	2872	3 sec.	13 sec.	45 sec.	$\varepsilon_t = 0.2689$	$\varepsilon_g = 0.0197$
Mesh 2	442	782	0.4 sec.	4 sec.	12 sec.	$\varepsilon_t = 0.2472$	$\varepsilon_g = 0.0243$
Mesh 3	415	732	0.2 sec.	2 sec.	8 sec	$\varepsilon_t = 0.2555$	$\varepsilon_g = 0.0321$

* Approximately 30 iterations are sufficient to generate a reasonably good anisotropic mesh.

Fig. 8. Mesh statistics.

mesh. As elements become more equilateral, this topological irregularity approaches 0, but vanishes only when all the nodes have exactly 6 neighbors, a rare situation. Otherwise, it has a positive value that measures how much the mesh topologically differs from a perfectly regular triangular lattice.

For geometric irregularity we define the following measure, ε_g, using the ratio of the inscribed circle radius to the circumcircle radius

$$\varepsilon_g = \frac{1}{m}\sum_{i=0}^{m}(0.5 - \frac{r_i}{R_i}) \tag{21}$$

where m is the number of triangles, and r_i the inscribed circle radius of the ith triangle, and R_i the circumcircle radius of the ith triangle. Since a resultant mesh is anisotropic and stretched according to a given tensor field, radii of inscribed circles and circumcircles should be calculated after the triangles' three node locations are transformed so that an ellipsoid is mapped back to a circle, a process similar to that of the anisotropic Delaunay triangulation described in Section 6. An average tensor for each triangle is calculated at the barycenter of the triangle. Since the ratio r_i/R_i is at maximum 0.5 for an equilateral triangle, an ideal element, the smaller the value of ε_g, the more geometrically regular the mesh.

Figure 9 shows an example of graded isotropic meshing of a single bicubic parametric surface. The diameters of the packed ellipsoids are adjusted by the minimum radius of curvature as follows

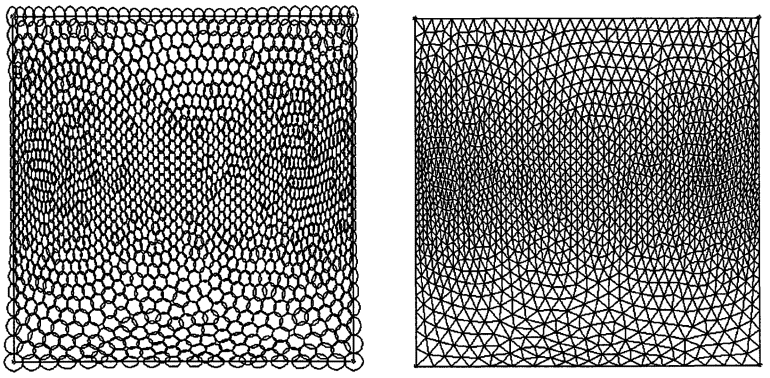
$$d_1 = d_2 = d_3 = min\Big(2\sqrt{2e\rho_{min} - e^2}, D_{max}\Big) \tag{22}$$

where ρ_{min} denotes the minimum radius of curvature, e a target constant error between the original surface and the mesh, and D_{max} the allowable maximum diameter of an ellipsoid. With this metric tensor definition all the bubbles become spheres, yielding a graded isotropic triangular mesh.

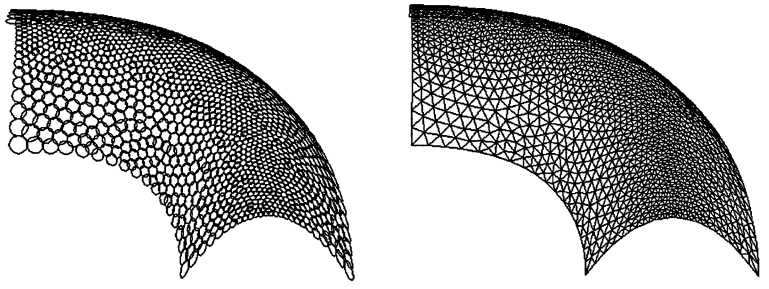
In addition to the minimum radius of curvature we can also calculate the maximum radius of curvature and use both radii to shape the ellipsoids to be packed, as shown in Figure 11. In this case the metric tensor is defined with

$$\begin{aligned} d_1 &= \quad min\Big(2\sqrt{2e\rho_{max} - e^2}, D_{max}\Big), \\ d_2 = d_3 &= \quad min\Big(2\sqrt{2e\rho_{min} - e^2}, D_{max}\Big), \end{aligned} \tag{23}$$

where ρ_{max} denotes the maximum radius of curvature, and D_{max} the allowable maximum value of the major diameter of an ellipsoid.



(a) Packed bubbles and a triangular mesh in parametric space



(b) Packed bubbles and a triangular mesh in object space

Fig. 9. Mesh 1: graded isotropic mesh based on the maximum curvature (1468 nodes, 2872 elements).

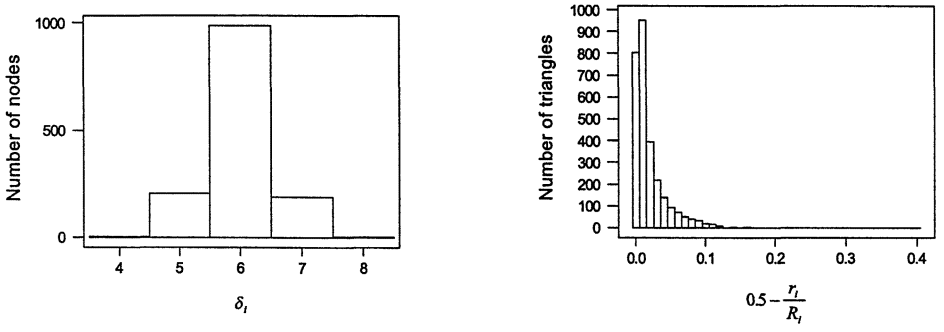
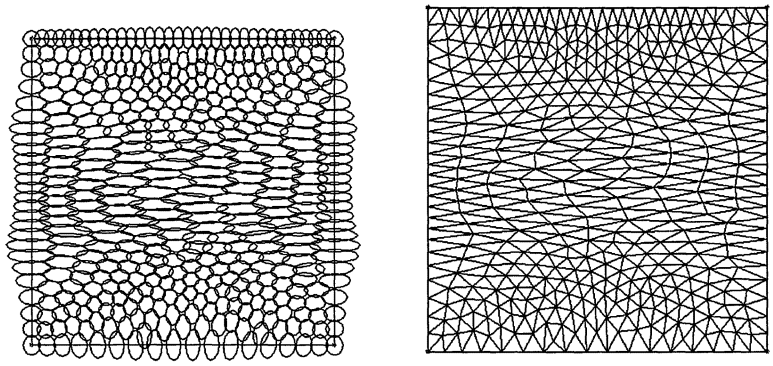
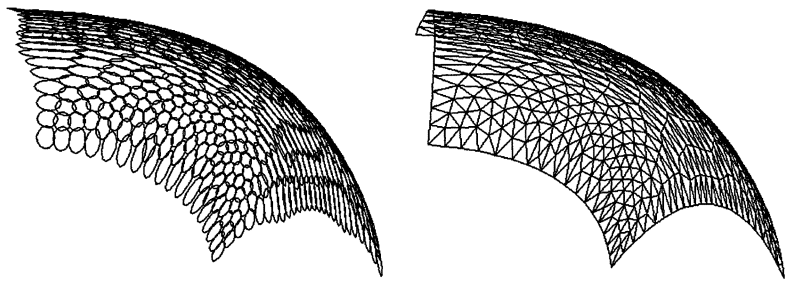


Fig. 10. Mesh 1: mesh quality histogram after 100 iterations.



(a) Packed bubbles and a triangular mesh in parametric space



(b) Packed bubbles and a triangular mesh in object space

Fig. 11. Mesh 2: graded anisotropic mesh based on the principal curvatures (442 nodes, 782 elements).

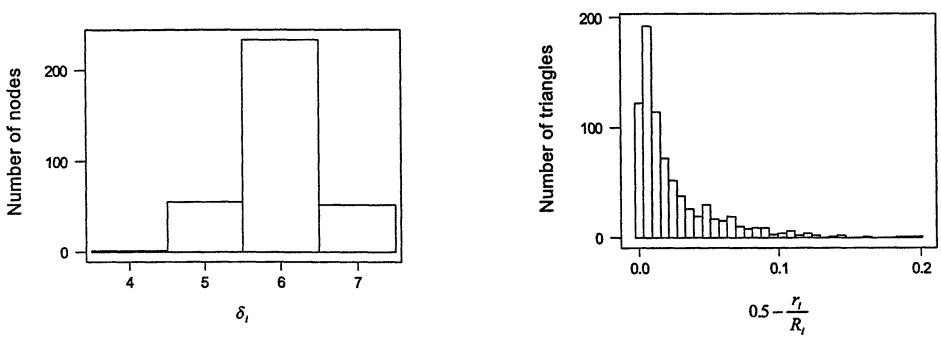
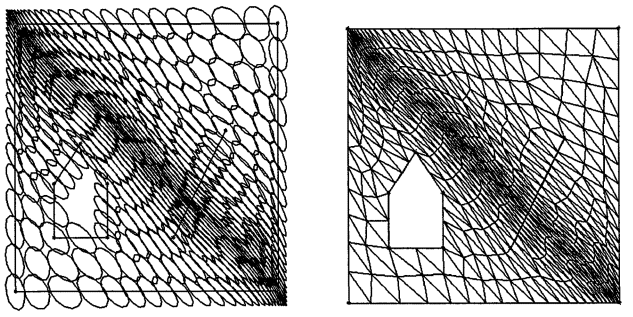
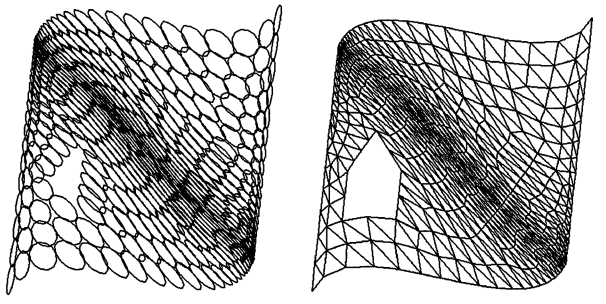


Fig. 12. Mesh 2: mesh quality histogram after 100 iterations.



(a) Packed bubbles and a triangular mesh in parametric space



(b) Packed bubbles and a triangular mesh in object space

Fig. 13. Mesh 3: mesh quality based on an arbitrarily defined metric tensor (415 nodes, 732 elements).

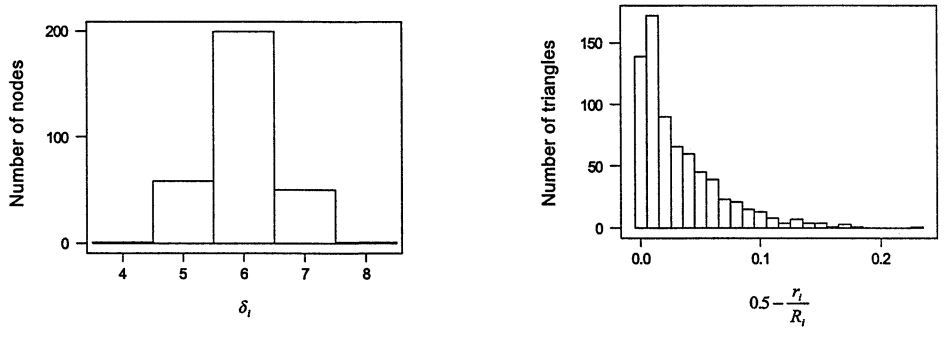
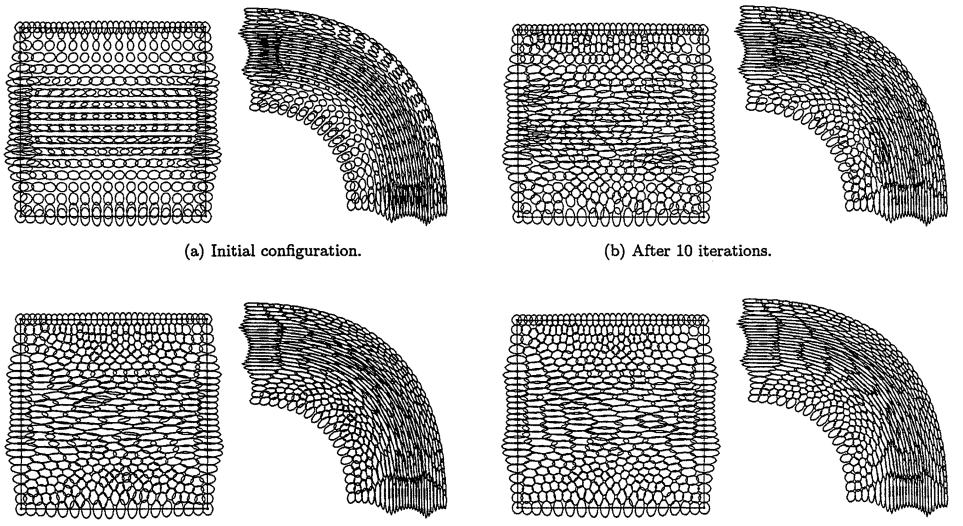


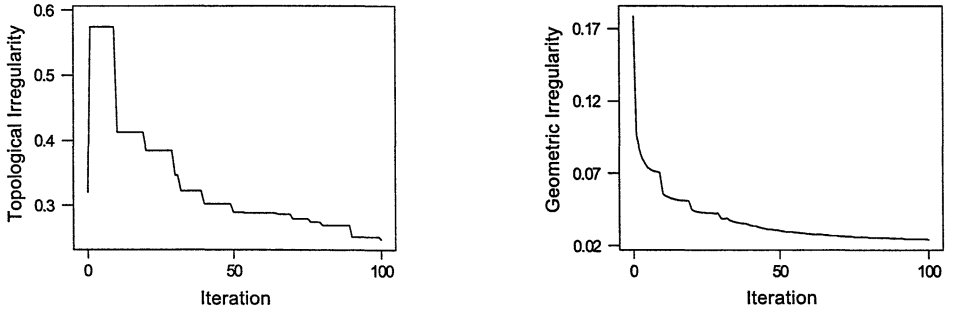
Fig. 14. Mesh 3: mesh quality histogram after 100 iterations.



(a) Initial configuration. (b) After 10 iterations.

(c) After 30 iterations. (d) After 100 iterations.

Fig. 15. Dynamic simulation of bubble movement (Mesh 2).



(a) Topological irregularity ε_t. (b) Geometric irregularity ε_g.

Fig. 16. Irregularity reduced during mesh relaxation (Mesh 2).

Figure 13 shows the anisotropic triangulation of a trimmed parametric surface with five trimming curves $\mathbf{C}_t$ and one inside curve $\mathbf{C}_i$ as shown in Figure 13(a). Because we pack bubbles on these curves before packing bubbles inside the trimmed region, mesh nodes are placed exactly on these curves in the final mesh shown in the right of Figure 13(b).

Figure 15 shows how bubbles are moved to a force-balancing configuration during dynamic simulation, yielding the mesh shown in Figure 11. During the mesh relaxation process both topological irregularity and geometric irregularity are reduced as shown in Figure 16. Although we can get a reasonably good mesh after about 30 iterations, mesh quality can still be improved after 100 iterations. The actual termination criteria of iterations should be decided based on analysis requirements.

Finally Figure 17 and 18 show more meshing results. In Figure 17 a trimmed parametric surface is defined, and the metric tensor is calculated based on the maximum radius of curvature and the minimum radius of curvature, identical to the one used for Mesh 2 shown in Figure 11.

Figure 18 shows a graded anisotropic triangulation of a simple 2D domain. We have defined a 2×2 tensor field so that the mesh is stretched along parabolic stream lines and so that element sizes are adjusted based on their proximity to three points defined inside the domain. Elements located closer to three points are smaller than those located farther away.

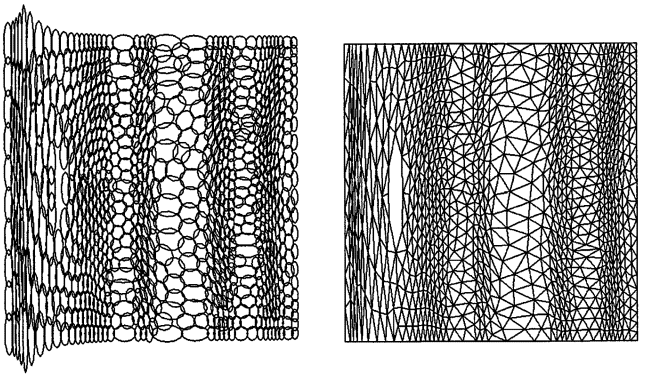
8. Discussion and Conclusion

We have presented a new physically-based method for anisotropic triangulation of a trimmed parametric surface. Our central idea was to pack ellipsoids (and ellipses in parametric space) closely in a domain to create a well-shaped mesh that conforms to a given 3×3 metric tensor field that specifies a desired anisotropy. The application is not limited to surface meshing as previous techniques are; in fact the method is designed so that it can be used as a subprocess in anisotropic meshing of 3D and non-manifold domains.

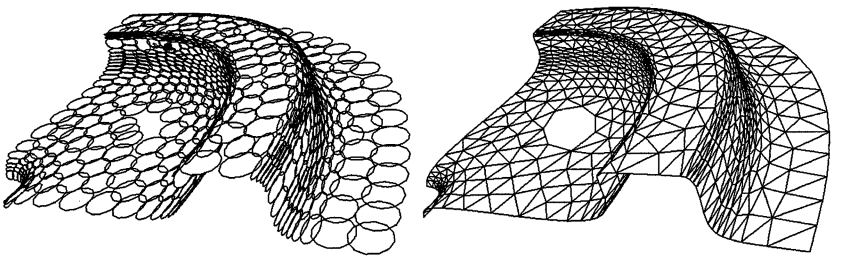
In our original sphere packing method for isotropic meshing, the hexagonal pattern created by the close packing of spheres mimics a Voronoi diagram corresponding to a well-shaped isotropic Delaunay triangulation. In our new method of packing ellipsoids for anisotropic meshing, the same concept applies, except the space is stretched, or deformed, by an anisotropic metric tensor. Consequently if an anisotropic mesh generated by our method is transformed by the inverse of the metric tensor, the node arrangement will be close to a regular hexagonal pattern.

Providing a good initial node distribution is essential in physically-based meshing approaches like ours. Although it is possible to start with a minimum number of "seed nodes" or "seed triangles" and wait until more nodes or triangles are added adaptively during the relaxation process, starting from a good initial configuration helps to reduce convergence time significantly. Also, when speed is critical this initial node distribution can itself be used for a quick triangulation solution.

In this paper we assumed that a desired anisotropy is given by a 3×3 metric tensor, which decides the shape and the size of an ellipsoid to be packed. This is



(a) Packed bubbles and a triangular mesh in parametric space



(b) Packed bubbles and a triangular mesh in object space

Fig. 17. Mesh 4: graded anisotropic mesh based on the principal curvatures (901 nodes, 1686 elements).

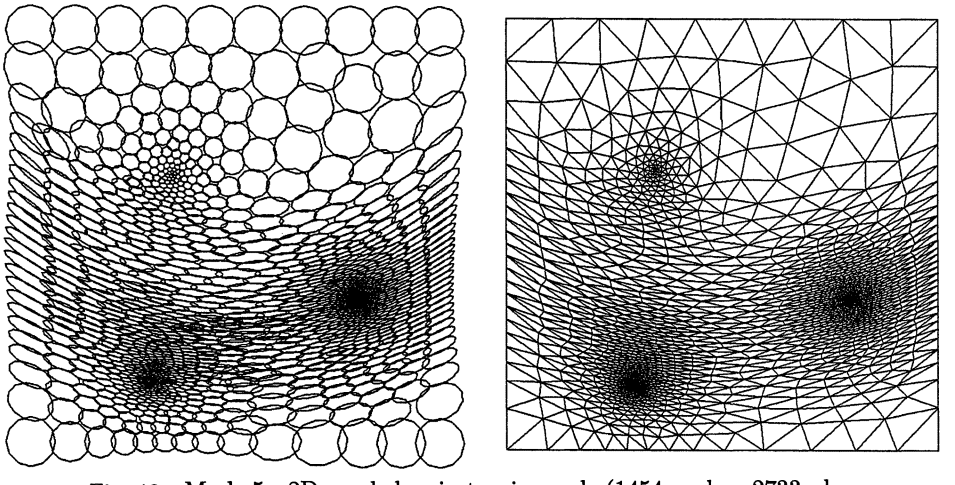


Fig. 18. Mesh 5: 2D graded anisotropic mesh (1454 nodes, 2733 elements).

because we wanted to make our method consistently applicable to 1D, 2D, surface, 3D, and non-manifold domains. In some cases, however, a desired anisotropy is naturally given by a 2×2 metric tensor in parametric space or on the tangent plane in object space; all of the curvature-based meshing examples in Section 7 are such cases. To deal with this situation, we proposed a simple rule to "expand" a 2×2 metric tensor to a 3×3 metric tensor by adding a third eigenvalue and eigenvector based on the first two.

The proposed method appears particularly suitable to FEM mesh generation, especially when the phenomena is anisotropic, as well as to approximating of a trimmed parametric surface to with linear triangular elements. The method is also well suited to adaptive mesh refinement when a geometry and/or a desired anisotropy changes over time.

References

1. Houman Borouchaki and Pascal J. Frey. Adaptive triangular-quadrilateral mesh generation. *Intl. J. Numer. Meth. Eng.*, 41:915–934, 1998.
2. Houman Borouchaki, Pascal J. Frey, and Paul Louis George. Unstructured triangular-quadrilateral mesh generation. application to surface meshing. In *Proc. of 5th Intl. Meshing Roundtable*, pages 229–242, 1996.
3. Frank J. Bossen and Paul S. Heckbert. A pliant method for anisotropic mesh generation. In *Proc. of 5th Intl. Meshing Roundtable*, pages 63–74, 1996.
4. M. J. Castro-Díaz, F. Hecht, and B. Mohammadi. New progress in anisotropic grid adaptation for inviscid and viscous flows simulations. In *Proc. of 4th Intl. Meshing Roundtable*, pages 73–85, 1995.
5. Luiz Henrique de Figueiredo, Jonas de Miranda Gomes, Demetri Terzopoulos, and Luiz Velho. Physically-based methods for polygonization of implicit surfaces. In *Proc. of Interface '92*, pages 250–257, 1992.
6. Kurt W. Fleischer, David H. Laidlaw, Bena L. Currin, and Alan H. Barr. Cellular texture generation. In *Proc. of SIGGRAPH'95*, pages 239–248, 1995.
7. William H. Frey and David A. Field. Mesh relaxation: A new technique for improving triangulations. *Intl. J. Numer. Meth. Eng.*, 31:1121–1133, 1991.
8. W. T. Reeves. Particle systems–a technique for modeling a class of fuzzy objects. In *Proc. of SIGGRAPH '83*, pages 359–376, 1983.
9. Kenji Shimada. *Physically-Based Mesh Generation: Automated Triangulation of Surfaces and Volumes via Bubble Packing.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, U.S.A., 1993.
10. Kenji Shimada. Automatic anisotropic meshing via packing ellipsoids. In *Proc. of Annual Autumn Meeting of IPSJ*, 1995. (in Japanese).
11. 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, 1995.
12. Kenji Shimada and David C. Gossard. Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis. *Computer Aided Geometric Design*, 15/3:199–222, 1998.
13. S. W. Sloan. A fast algorithm for constructing delaunay triangulations in the plane. *Adv. Eng. Software*, 9(1):34–55, 1987.

14. Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. In *Proc. of SIGGRAPH'92*, pages 185–194, 1992.
15. Greg Turk. Re-tiling polygonal surfaces. In *Proc. of SIGGRAPH '92*, pages 55–64, 1992.
16. Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. of SIGGRAPH '94*, pages 269–277, 1994.