Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles

Kenji Shimada *
Carnegie Mellon University

Atsushi Yamada and Takayuki Itoh †
IBM Research, Tokyo Research Laboratory

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 x 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 high quality anisotropic mesh whose elements conform precisely to the given tensor field.

Keywords: unstructured mesh, anisotropy, parametric surface, metric tensor, Delaunay triangulation

1 Introduction

Although most automatic mesh generators try to create a regular isotropic mesh, in some FEM analysis an anisotropic mesh is more efficient in terms of computational time and solution accuracy. For example, in fluid dynamics simulation an anisotropic mesh stretched along shock/boundary layers and stream lines is preferred.

This paper presents a versatile computational method of automatically generating an anisotropic triangular mesh of a trimmed parametric surface, applicable to various FEM analyses. Assuming that an anisotropy is given as a 3 x 3 tensor field defined over the domain to be meshed, this surface triangulation problem can be 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 x 3 tensor field $\mathbf{M}(\mathbf{x})$

Generate:

- an anisotropic 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

*Kenji Shimada, Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, U.S.A. shimada@cmu.edu, http://www.me.cmu.edu/faculty1/shimada/
†Atsushi Yamada and Takayuki Itoh, IBM Research, Tokyo Research Laboratory, 1623-14, Shimo-tsuruma Yamato-shi, Kanagawa-ken 242, Japan, ayamada@trl.ibm.co.jp, itot@trl.ibm.co.jp

$\mathbf{C}_i(s) = \big(x(s), y(s), z(s)\big)$ 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 a derivative vector 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 [8, 11, 10, 16]. 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 x 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.

2 Related Work

2.1 Anisotropic Meshing

In approximating 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. Similarly, in finite element analysis of a physical phenomenon, when the phenomenon has a strong directionality as in fluid dynamics, an anisotropic mesh is more efficient in terms of computational time and solution accuracy than an isotropic mesh.

One common way to represent an anisotropy is to define a metric tensor field, $\mathbf{M}$, over the domain [3, 2, 1]. $\mathbf{M}$ is a symmetric positive-definite 2 x 2 matrix in two dimensional problems, and a symmetric positive-definite 3 x 3 matrix in three dimensional problems. Castro-Díaz et al. showed 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 [3]. Bossen and Heckbert used as input a 2 x 2 metric tensor in generating a 2D planar anisotropic triangular mesh using a system of interacting particles [2]. Borouchaki et al. showed how a metric tensor can be used to generate an anisotropic triangular mesh on a surface and to convert it to a quadrilateral mesh [1]. Shimada used a 2D vector field equivalent to a 2 x 2 tensor for 2D anisotropic meshing [9].

In this paper we use the same metric tensor to control the size and the shape of an ellipsoid to be packed in the domain.

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 [7]. 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 [14], Szeliski's surface modeling [13], de Figueiredo et al.'s polygonization of implicit surfaces [4], Witkin and Heckbert's sampling and controlling of implicit surfaces [15], and Fleischer et al.'s texture generation [5].

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 applied an interacting particle system to 2D anisotropic FEM mesh generation [2]. The method assumes a 2 x 2 metric tensor to specify an anisotropy in a planar region, similar to Castro-Diaz's [3], and generates

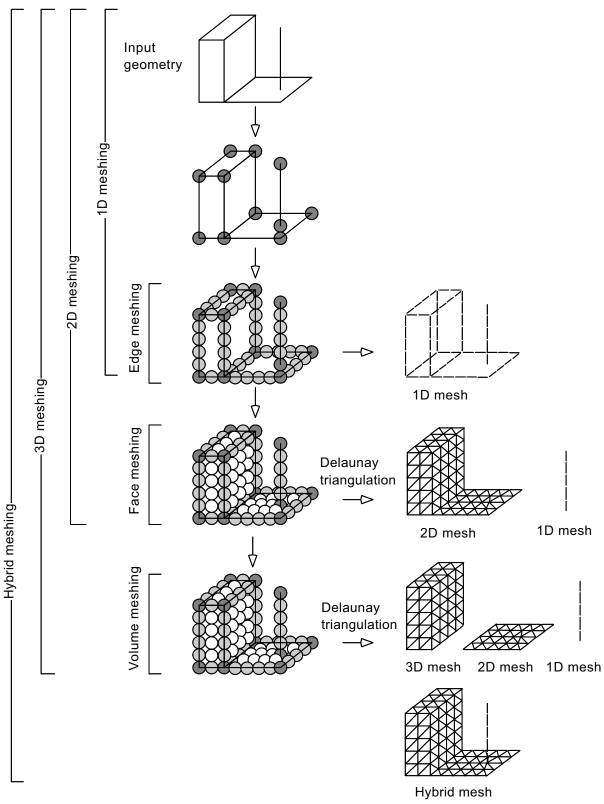


Figure 1: Bubble meshing procedures

a anisotropic node distribution using a proximity-based force similar to Shimada's [8, 11]. 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 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 configuration.

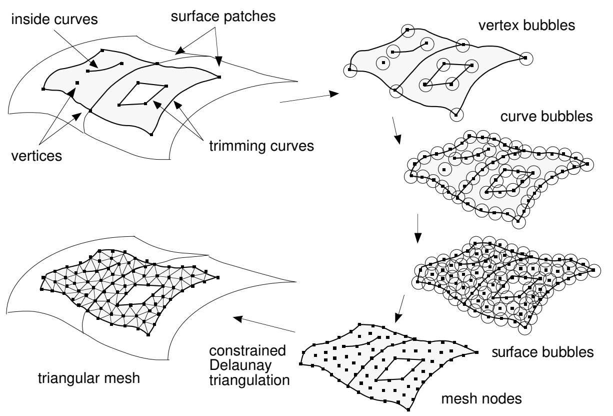


Figure 2: Ellipsoidal bubble packing procedure.

In the original bubble mesh, spheres are closely packed to create isotropic meshes in 1D, 2D, surface, and 3D domains [10, 8, 11, 16]. 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 [9]. 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 Anisotropic Triangulation of Parametric Surfaces

3.1 Triangulation Procedures

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.

This packing process should be performed 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 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 $\mathbf{S}$.

Because bubbles are placed in order of dimension two fixed bubbles are already placed at the two endpoints of the curve when bubbles are packed on a 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 this way 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 region of a trimmed parametric surface, their centers are connected by *anisotropic Delaunay* triangulation, detailed in Section 3.5, for a complete mesh topology.

3.2 Ellipsoidal Bubbles

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

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

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[1] of an ellipsoidal bubble.

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

3.3 Metric Tensor for Parametric Surfaces

Although we need a 3 x 3 metric tensor field $\mathbf{M}(\mathbf{x})$ to specify the size and the shape of an ellipsoid, a desired anisotropy is often given by a 2 x 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:

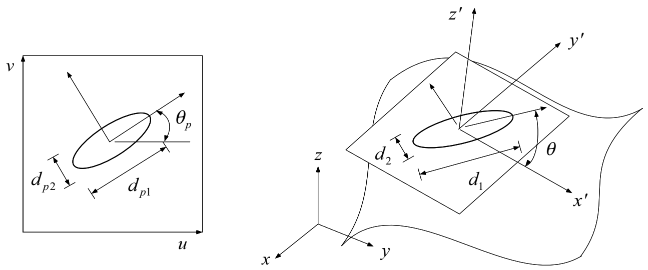
- How to find a corresponding ellipse, or a 2 x 2 tensor in parametric space, when an anisotropy is defined by a 2 x 2 tensor in object space. This is necessary for anisotropic Delaunay triangulation in parametric space.
- How to define an ellipsoid, or a 3 x 3 tensor, in object space, necessary for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

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 x 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

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

[1] 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.



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

Figure 3: Tensor ellipse on a tangent plane.

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.

To find a corresponding ellipse in parametric space, we first find the following 2 x 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 x 2 metric tensor in object space $\mathbf{M}_{x'y'}$, the 2 x 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 x 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 x 2 metric tensor, or an ellipse, in parametric space when a 2 x 2 tensor field is given on the surface in object space.

A particularly useful 2 x 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 [8]

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

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.

As mentioned earlier in this section we also need to find how to define a 3 x 3 metric tensor, or a tensor ellipsoid, when only a 2 x 2 metric tensor is given on the surface. This is essential because, as detailed in Section 3.4, interbubble forces are calculated using ellipsoids defined by a 3 x 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 x 3 metric tensor is thus defined as

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

3.4 Bubble Packing by Proximity-Based 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 a realistic simulation of molecules' behavior, but is to find a force-balancing configuration efficiently. This is why we 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

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

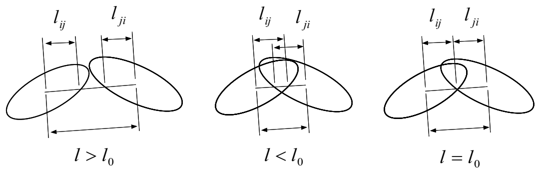
As shown in Figure 4, this force model applies either a repelling or attracting force between two bubbles based on the following distance comparison. Assuming that two bubbles are adjacent to each other, a repelling force is applied if l is smaller than l_0, or if $w < 1.0$. An attracting force is applied if l is longer than l_0, or if $1.0 < w < 1.5$. No force is applied if 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 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 [8, 11, 16]

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

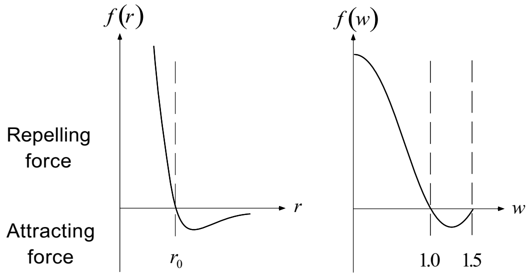
where $d(\mathbf{x}_i)$ and $d(\mathbf{x}_j)$ are the diameters of bubble i and bubble j respectively. If two bubbles are ellipsoidal, however, this target stable distance should be calculated as the summation of the two lengths, measured along the line segment that connects the centers of the two ellipsoids, 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

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



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

Figure 4: Target stable distance.



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

Figure 5: Interbubble proximity-based force.

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.

Compared to the van der Waals force, our force, as also 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: 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.

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

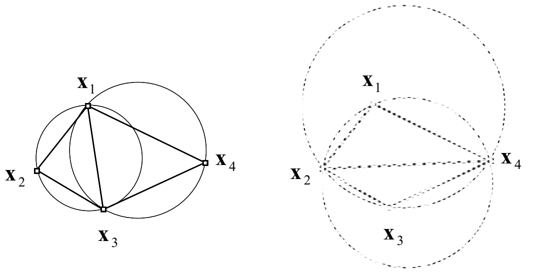
$$\mathbf{f}_i \cdot \mathbf{n}_i = 0, \qquad i = 1, \ldots, n, \tag{17}$$

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.

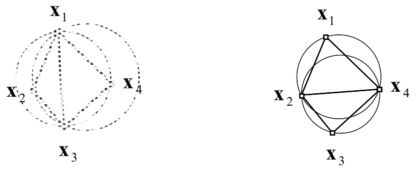
Due to an arbitrarily defined tensor field and geometric constraints on bubble locations, Equation 17 is highly nonlinear, and thus it is difficult to solve the equation directly by a multidimensional root-finding technique such as the Newton-Raphson method. Our alternative approach is to assume a point mass m at the center of each bubble and the effect of viscous damping c, and to solve the following equation of motion[2] by using a standard numerical integration scheme, the fourth-order Runge-Kutta method.

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

[2] The first order equation can also be used [2]. 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 $\triangle \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$ and $\triangle \mathbf{x}_1\mathbf{x}_3\mathbf{x}_4$



(b) Anisotropic circumcircle test with $\widetilde{\mathbf{M}}_{uv} = \begin{pmatrix} 1 & 0 \\ 0 & 4 \end{pmatrix}$ will choose $\triangle \mathbf{x}_1\mathbf{x}_2\mathbf{x}_4$ and $\triangle \mathbf{x}_2\mathbf{x}_3\mathbf{x}_4$

Figure 6: Anisotropic circumcircle test.

In solving Equation 18 numerically, we have to impose geometric constraints so that all the mobile bubbles do not pop out of given parametric curves and surfaces. For this purpose, we perform a process of remapping unconstrained bubble movements onto a curve or a surface by using tangent vectors.

Another process incorporated in solving the equation of motion is adaptive bubble population control. This is important because we do not know beforehand an appropriate number of bubbles that is necessary and sufficient to fill the region. Although our initial bubble configuration generator gives a reasonably good guess, it is still not optimal. To solve this problem, we implemented a procedure to check a local population density and to add more bubbles in sparse areas and delete bubbles in over-packed areas.

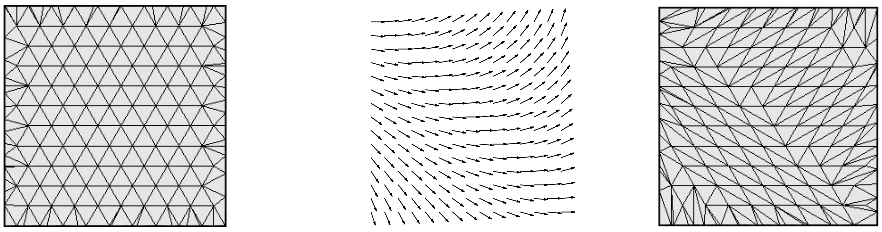
The methods of imposing geometric constraints on the bubble movements and adjusting bubble population are common between anisotropic meshing and isotropic meshing, and the details can be found elsewhere [8, 11].

3.5 Anisotropic Delaunay Triangulation

Once a force-balancing configuration of ellipsoidal bubbles is obtained bubble 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 [12], which we implemented in the original 2D isotropic bubble mesh. As shown in Figure 6, the circumcircle test is perfomed for a pair of adjacent triangles that form a convex quadrilateral. Given a set of such four points, the circumcircle test checks one of the triangles to see whether the forth point is inside the circumcircle. If it is, the diagonal edge is swapped.

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 calculated by first calculating the barycenter of the four



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

Figure 7: An example of anisotropic Delaunay triangulation.

nodes and then finding the metric tensor at this barycenter[3]

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

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 anisotropic mesh that is stretched and "flows" along the direction of the major eigenvectors shown in 7(b).

4 Results

The anisotropic meshing described above has been implemented in C and C++. Three meshing results are shown in Figures 8, 10, and 12, and their quality measures are shown in Figures 9, 11, and 13 respectively. Table 1 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 simulations, and the final meshing after 100 iterations of dynamic simulations; 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 9, Figure 11, Figure 13, and Table 1, 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 [6]:

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

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 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

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

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

Table 1: Mesh statistics.

Mesh	*Nodes*	*Elements*	*CPU time* *for initial mesh*	*CPU time* for 30 iteration	*CPU time* *for 100 iteration*	*Mesh irregularity* *after 100 iteration*	
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	$\varepsilon_t = 0.2555$	$\varepsilon_g = 0.0321$

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 3.5. 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 8 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 ellipsoids to be packed, as shown in Figure 10. 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.

Figure 12 shows an 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 12(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 12(b).

Figure 14 shows how bubbles are moved to a force-balancing configuration during dynamic simulation, yielding the mesh shown in Figure 10. During the mesh relaxation process both topological irregularity and geometric irregularity are reduced as shown in Figure 15. Although we can get a reasonably good mesh after about 30 iterations, mesh quality can be still improved after 100 iterations. The actual termination criteria of iterations should be decided based on analysis requirements.

5 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 x 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 more 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 x 3 metric tensor, which decides the shape and the size of an ellipsoid to be packed. This is 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 x 2 metric tensor in parametric space or on the tangent plane in object space; all of the curvature-based meshing examples in Section 4 are such cases. To deal with this situation, we proposed a simple rule to "expand" a 2 x 2 metric tensor to a 3 x 3 metric tensor by adding a third eigenvalue and eigenvector based on the first two.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] William H. Frey and David A. Field. Mesh relaxation: A new technique for improving triangulations. *Intl. J. Numer. Meth. Eng.*, 31:1121–1133, 1991.

[7] W. T. Reeves. Particle systems–a technique for modeling a class of fuzzy objects. In *Proc. of SIGGRAPH '83*, pages 359–376, 1983.

[8] 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.

[9] Kenji Shimada. Automatic anisotropic meshing via packing ellipsoids. In *Proc. of Annual Autumn Meeting of IPSJ*, 1995. (in Japanese).

[10] Kenji Shimada and David Gossard. Computational methods for physically-based FE mesh generation. In *Proc. of the IFIP TC5/WG5.3 Eight International PROLAMAT Conference*, pages 411–420, 1992.

[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.

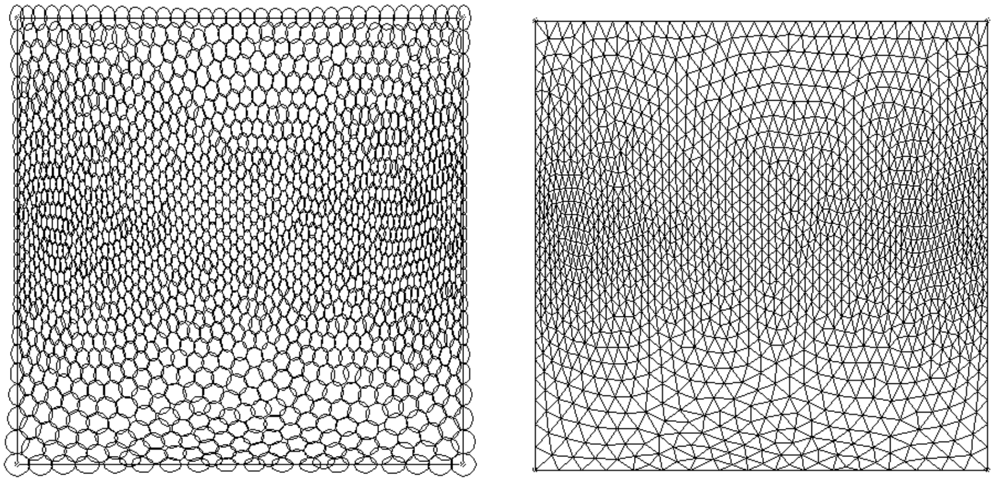
[12] S. W. Sloan. A fast algorithm for constructing delaunay triangulations in the plane. *Adv. Eng. Software*, 9(1):34–55, 1987.

[13] Richard Szeliski and David Tonnesen. Surface modeling with oriented particle systems. In *Proc. of SIGGRAPH'92*, pages 185–194, 1992.

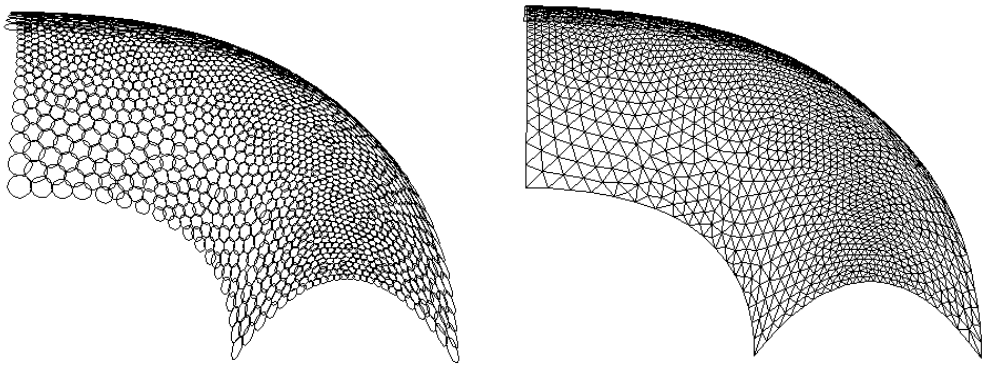
[14] Greg Turk. Re-tiling polygonal surfaces. In *Proc. of SIGGRAPH '92*, pages 55–64, 1992.

[15] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. In *Proc. of SIGGRAPH '94*, pages 269–277, 1994.

[16] Atsushi Yamada, Kenji Shimada, and Takayuki Itoh. Energy-minimizing approach to meshing curved wire-frame models. In *Proc. of 5th Intl. Meshing Roundtable*, pages 179–191, 1996.



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



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

Figure 8: Mesh 1: graded isotropic mesh based on the maximum curvature (1468 nodes, 2872 elements).
$$d_1 = d_2 = d_3 = min\left(2\sqrt{2e\rho_{min} - e^2}, D_{max}\right).$$

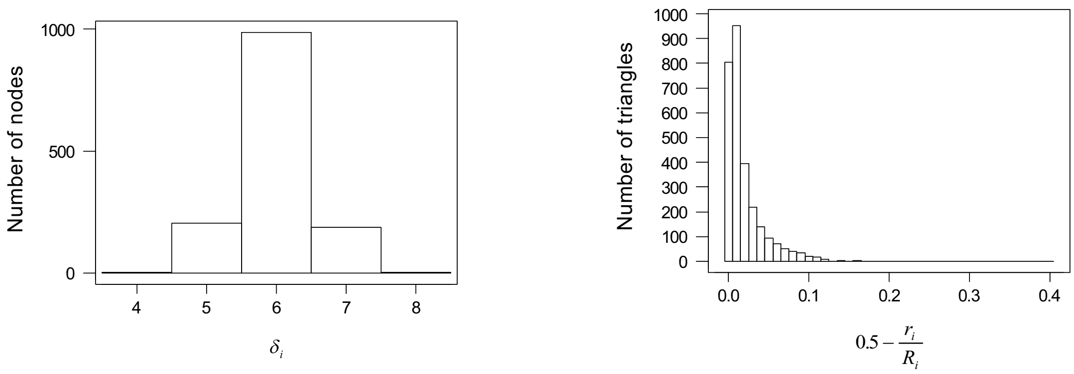
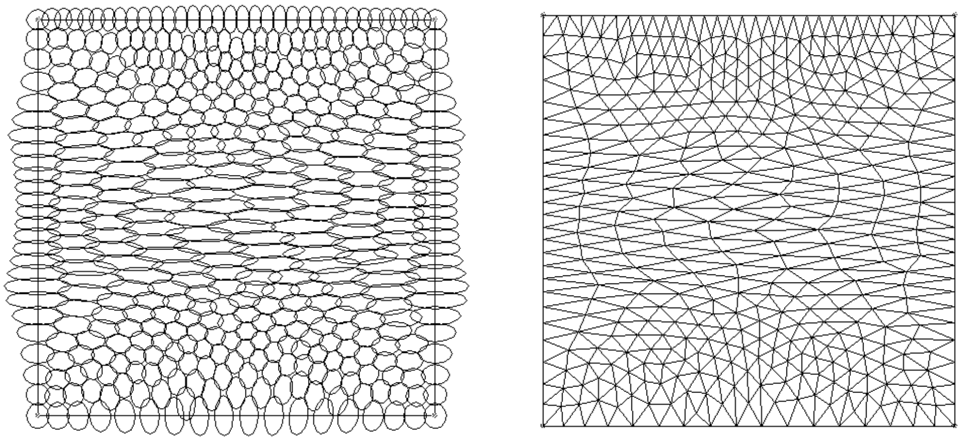
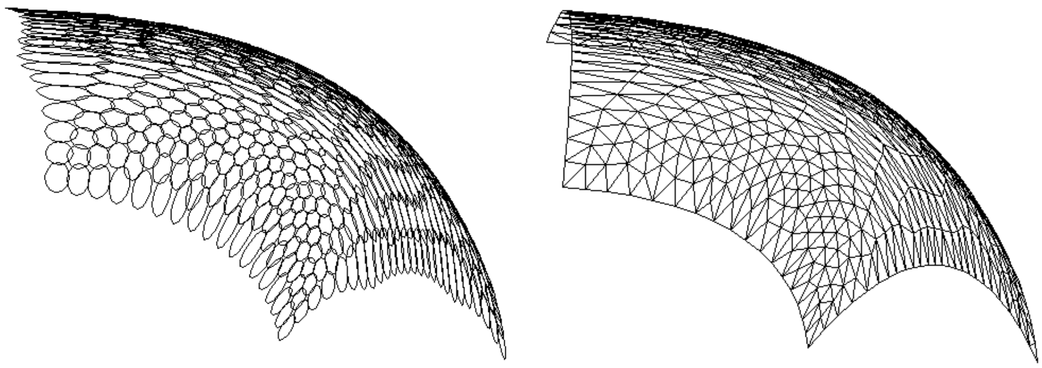


Figure 9: Mesh 1: mesh quality histogram after 100 iterations.



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



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

Figure 10: Mesh 2: graded anisotropic mesh based on the principal curvatures (442 nodes, 782 elements). $d_1 = min\Big(2\sqrt{2e\rho_{max} - e^2}, D_{max}\Big),\quad d_2 = d_3 = min\Big(2\sqrt{2e\rho_{min} - e^2}, D_{max}\Big).$

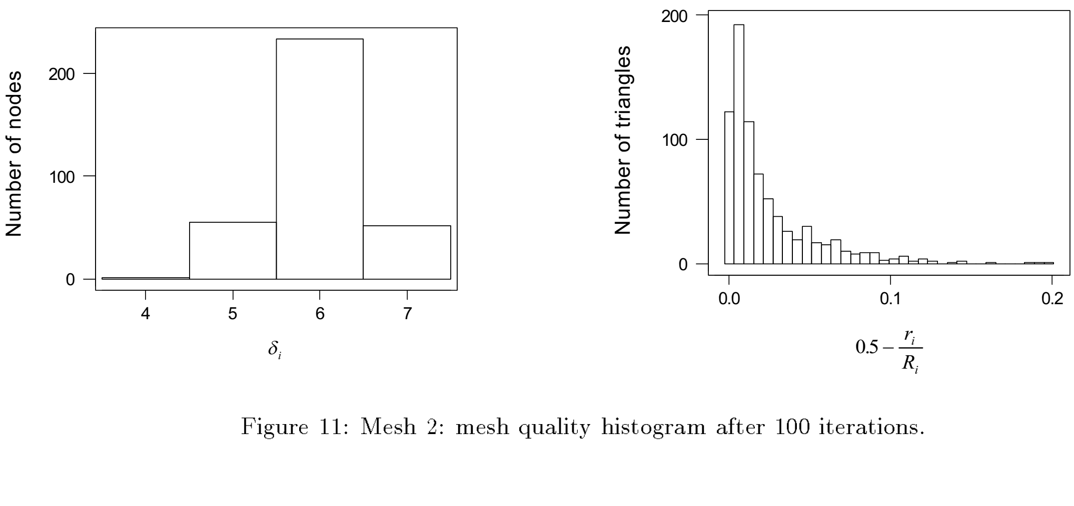
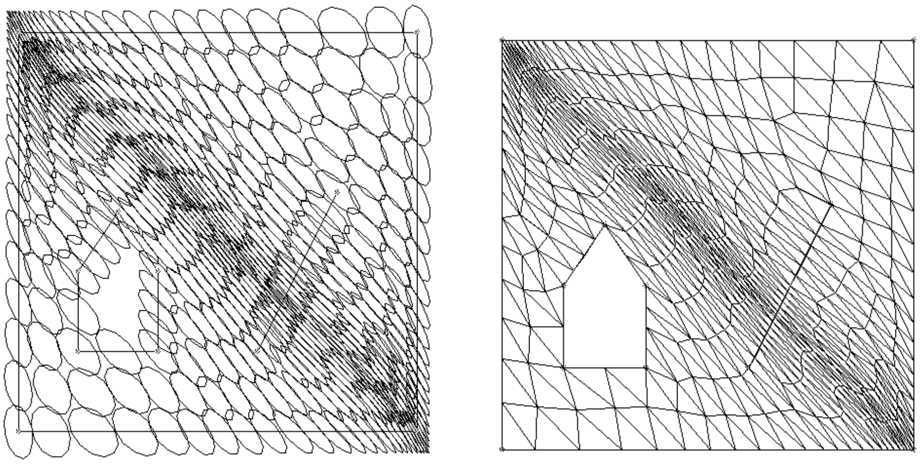
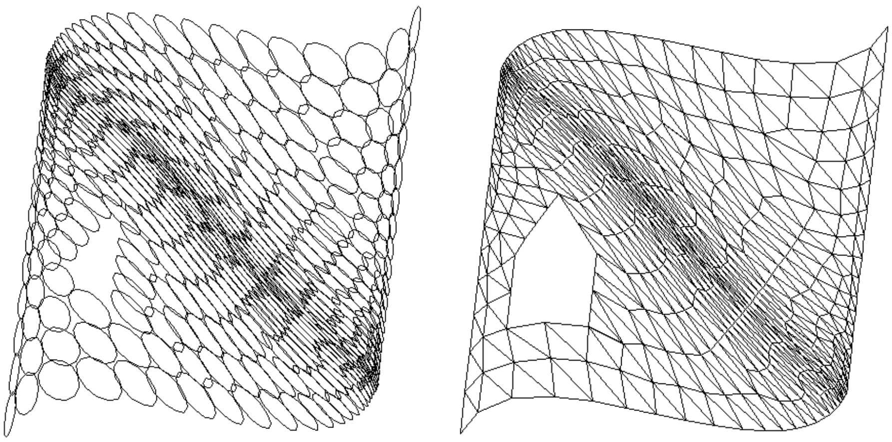


Figure 11: Mesh 2: mesh quality histogram after 100 iterations.



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



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

Figure 12: Mesh 3: mesh quality based on the arbitrarily defined metric tensor (415 nodes, 732 elements).

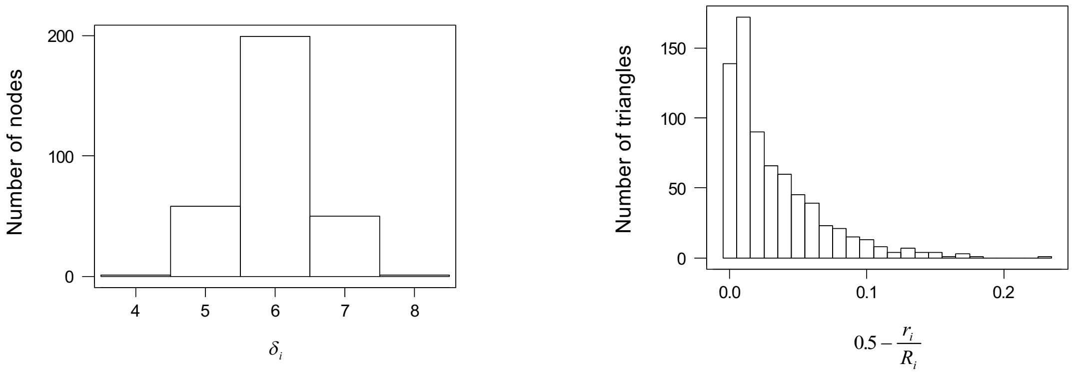
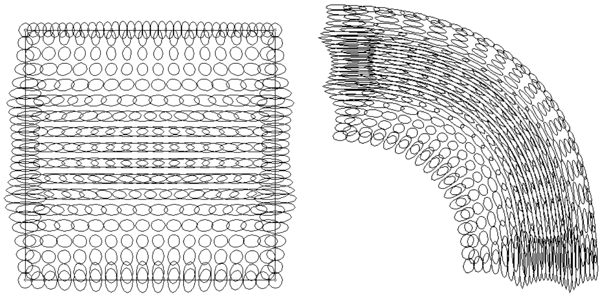
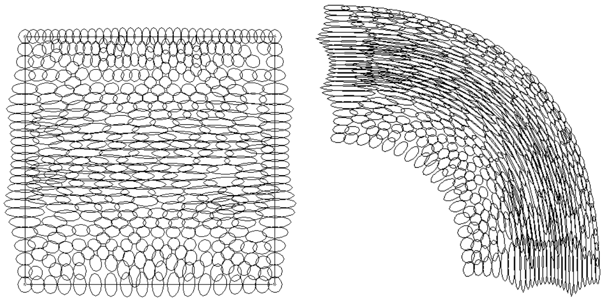


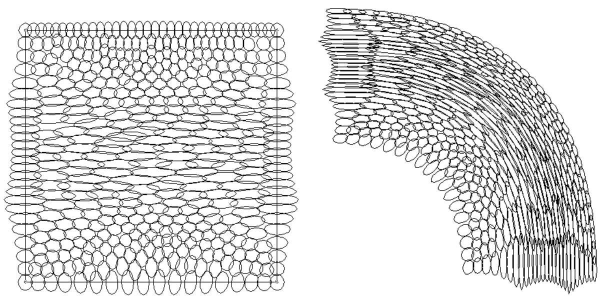
Figure 13: Mesh 3: mesh quality histogram after 100 iterations.



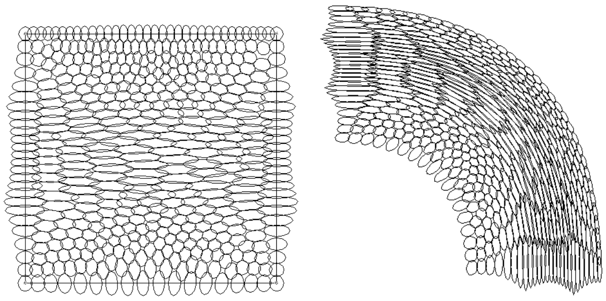
(a) Initial configuration.



(b) After 10 iterations.

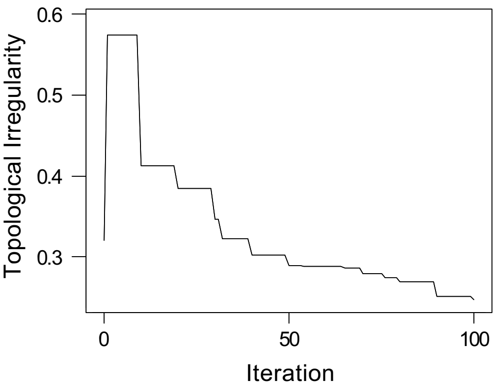


(c) After 30 iterations.

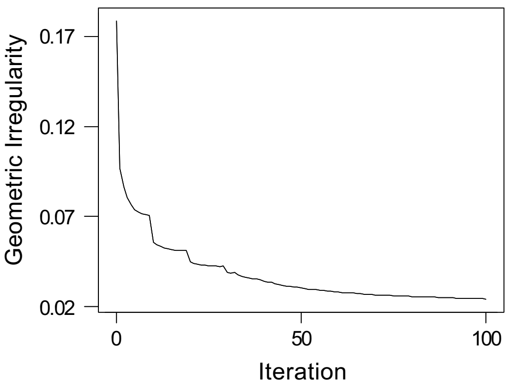


(d) After 100 iterations.

Figure 14: Dynamic simulation of bubble movement (Mesh 2).



(a) Topological irregularity ε_t.



(b) Geometric irreguarlity ε_g.

Figure 15: Irregulartity reduced during mesh relaxation (Mesh 2).