Triangular/Quadrilateral Remeshing of an Arbitrary Polygonal Surface via Packing Bubbles

Soji Yamakawa[1] and Kenji Shimada[2*]
[1]*Carnegie Mellon University, Pittsburgh, PA, U.S.A., soji@andrew.cmu.edu*
[2]*Carnegie Mellon University, Pittsburgh, PA, U.S.A., shimada@cmu.edu*

Abstract

This paper describes a new computational method for creating a triangular and quadrilateral mesh of an arbitrary polygonal surface while controlling the anisotropy and directionality of the mesh. The input polygonal surface may include non-manifold edges and holes, and it can be either a closed polyhedron or an open polygonal surface. The method creates a mesh in two steps. In the first step, volumetric cells are packed – ellipsoidal bubbles for a triangular mesh, or rectangular solid bubbles for a quadrilateral mesh – on the input polygonal surface, then vertices are created at the centers of the bubbles and superimposed onto the input polygonal surface. In the second step, the original vertices of the input polygonal surface are deleted. The method is tolerant to the noise typically introduced by the measurement error of a laser range scanner, it can control element size and anisotropy precisely, and it creates high quality mesh elements. Applications of the proposed scheme include: reducing the number of elements of a mesh created by a laser range scanner, CT scanner, or MRI scanner; creating a surface mesh that can be a starting mesh for a tetrahedral or a hexahedral finite element mesh; and solution-adaptive anisotropic remeshing for finite element analysis.

1. Introduction

This paper describes a new computational method for creating a triangular and quadrilateral mesh of an arbitrary polygonal surface. The remeshing process of the proposed method consists of two major steps. In the first step, volumetric cells – ellipsoidal bubbles for a triangular mesh or rectangular solid bubbles for a quadrilateral mesh – are packed on the input polygonal surface. The new vertices are then created at the centers of the bubbles and superimposed onto the input polygonal surface. In the second step, the original vertices of the input polygonal surface are deleted by using a method similar to Turk's method [1]. An edge-swapping scheme is applied to improve the quality of the elements during and after the deletion of the vertices.

The applications of the proposed method include: reducing the number of elements of a mesh created by a laser range scanner, CT scanner or MRI scanner; creating a surface mesh that can be a starting mesh for a tetrahedral or a hexahedral finite element mesh; and solution-adaptive anisotropic remeshing for the finite element analysis.

Due to these wide varieties of applications, the remeshing algorithm must meet many requirements, and none of the existing remeshing algorithms can meet all of the requirements.

First of all, the remeshing algorithm must be able to deal with wide varieties of input polygonal surfaces. The input polygonal surface may include non-manifold edges and holes, and may be an open polygonal surface or a closed polyhedron. The coordinates of vertices may be adversely affected by noise, yielding unwanted texture on the surface. The polygonal surface may consists of too many polygons to be rendered at a reasonable frame rate, or may consists of too few polygons yielding poor quality and irregularity.

The remeshing algorithm must be able to control element size, anisotropy, and directionality precisely. When the purpose of remeshing is to reduce the number of elements with minimum loss of the appearance of the

* Corresponding author:
Kenji Shimada, Department of Mechanical Engineering, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213-3890, U.S.A.
Tel: (412)268-3614, Fax: (412)268-3348, Email: shimada@cmu.edu

polygonal surface, element size and anisotropy must be controlled by the estimated curvature. On the other hand, when the purpose is solution-adaptive remeshing, element size and anisotropy must be controlled by the preliminary finite element analysis result. When a quadrilateral mesh is needed, the remeshing algorithm must control the directionality of the mesh. Such directionality may be given by the geometry of the input surface, or by the gradient of the preliminary analysis result. Typical curvature-based anisotropic remeshing schemes are insufficient for solution-adaptive remeshing.

Reducing the irregularity is another requirement for the remeshing algorithm. Ideally, a vertex of a triangular mesh should be connected to six edges. This condition is important for creating a subdivision surface or for applying a smoothing scheme. However, a mesh created by a laser-range scanner, CT scanner, and MRI scanner, tends to put vertices in rectangular grid pattern, and many of the vertices are not connected to six edges. A triangular facet data, which is often used for exchanging data between CAD systems, tends to have the minimum number of polygons that keeps the error from the original surface within a certain tolerance, but no care is taken to minimize irregularity when created. As a result, most of the vertices of such a facet data are not connected to six edges.

The remeshing algorithm must be able to create high-quality mesh elements. The quality is particularly important for a finite element mesh. The typical output of a laser range scanner, CT scanner, and MRI scanner, and a typical facet data exported from a CAD system include severely poor-quality elements, which are inappropriate for finite element analysis.

The proposed method addresses all the requirements described above. The advantages of the proposed methods are:

- The proposed method does not require parameterization of a polygonal surface; it can thus create a mesh of a polygon surface even when a reasonable parameterization cannot be found.
- The proposed method has virtually no limitation on the topology of the input polygonal surface. The input polygonal surface can be either a closed polyhedron or an arbitrary open polygonal surface, and it can include non-manifold edges and holes.
- The input polygonal surface is not limited to a triangular or quadrilateral mesh, and it can include polygons consisting of more than four vertices.
- The proposed method is tolerant to the noise typically introduced by a laser range scanner, CT scanner or MRI scanner.
- The output mesh can be either a triangle or a quadrilateral mesh.
- The proposed method can control anisotropy or directionality precisely.
- The proposed method creates high quality mesh elements.

The organization of the paper is as follows. Section 2 reviews related work. Section 3 shows the overview of the proposed method. Section 4 explains a function that specifies desired element shape. Section 5 explains the bubble packing process followed by the vertex deletion process in Section 6. Because the proposed method is an extension of the bubble packing method, Section 5.1 will first introduce the bubble packing method. Next, Sections 5.2, 5.3, and 6 discuss how this method is extended to the remeshing problem. Section 7 presents some examples, and Section 8 offers conclusions.

2. Related work

Many techniques have been proposed for decimating a triangular mesh based on an error estimation scheme. Garland and Heckbert's method is based on quadric error metrics [2, 3], and Rassineux et al.'s method is based on local Hermite surface fitting [4]. These methods are good for reducing the number of triangles with minimum loss of the initial appearance. However, when a noisy polygonal surface is given – such as an output of a laser-range scanner, it is nearly impossible to estimate the curvature of the input surface accurately, and these methods cannot reduce the number of polygons effectively.

Some of the remeshing methods are based on the geometric characteristics. Alliez et al. present a method that relies on the first principal curvature direction [5]. Sifri et al. present a remeshing method based on geodesic distance from a point on the input polygonal surface [6]. This type of method typically is not effective in dealing with a noisy polygonal surface. Although Sifri et al.'s method is relatively tolerant to the noise, it does not preserve constrained edges and thus requires a post-process to recover such constrained edges. The details of such a post-process are not described in their paper.

Other researches propose methods based on the parameterization of the polygonal surface [7, 8]. These methods reduce a remeshing problem into a 2D meshing problem by mapping the polygonal surface to a parametric space; they work only when a good parameterization is found for the input polygonal surface. Inappropriate parameterization yields distorted elements in the output mesh. When the input polygonal surface has a hole, the input surface must be partitioned so that each section can be mapped to a 2D parametric space. However, automatic

partitioning schemes usually create uneven boundaries between the partitions, yielding distorted elements. For an improved result, human intervention is needed for partitioning.

Another group of methods is based on the vertex-locating algorithm. Turk presents a remeshing method called re-tiling [1]. Heckbert presents an algorithm for distributing vertices on surfaces [9]. Surazhsky et al. present a method based on Volonoi tessellation [10]. The method presented in this paper also pertains to this category, and the vertex decimation algorithm is similar to Turk's vertex decimation method. These methods create a high-quality mesh when good vertex locations are acquired.

Many of the existing methods are limited to the triangular mesh and have limited or no control of anisotropy. Existing anisotropic remeshing techniques are based on error estimation or curvature estimation [2, 3, 5]. These methods cannot be used for solution adaptive anisotropic remeshing.

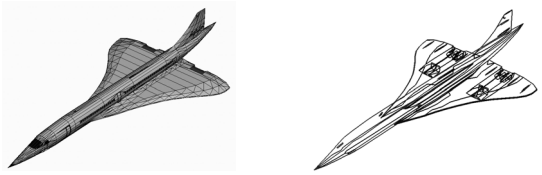
3. Overview

The original idea of the proposed method is found in [11-14]. The proposed method is an expansion of the original to the remeshing problem.

The input to the proposed method is an arbitrary polygonal surface, a type of output mesh, and a function that gives the desired shape of the elements. Detailed explanation of the functions is presented in Section 4. The polygonal surface may include some constrained vertices, which must be retained in the output mesh, and constrained edges can also be specified as a chain of edges of the input polygonal surface. Constrained edges can be automatically created based on face clustering [15], or based on dihedral angles, or manually specified by the user. A non-manifold edge is considered to be a constrained edge. A vertex where more than two constrained edges intersect is considered to be a constrained vertex. Figure 1 shows an example of the input polygonal surface.

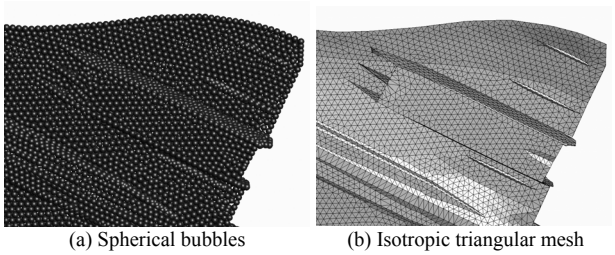
The output can be an isotropic triangular mesh, an anisotropic triangular mesh, an isotropic quadrilateral mesh, or an anisotropic quadrilateral mesh as shown in Figure 2 (b), (d), (f), and (h).

The proposed method creates a mesh in two steps. In the first step, volumetric cells are packed on the polygonal surface, spherical bubbles for an isotropic triangular mesh, ellipsoidal bubbles for an anisotropic triangular mesh, cube bubbles for an isotropic quadrilateral mesh, and rectangular solid bubbles for an anisotropic quadrilateral mesh, as shown in Figure 2 (a), (c), (e), and (g). The bubbles are moved to stable positions by physically-based particle simulation. New vertices are then created at the centers of the cells and are superimposed onto the input polygonal surface. The details of the packing process are explained in Section 5.

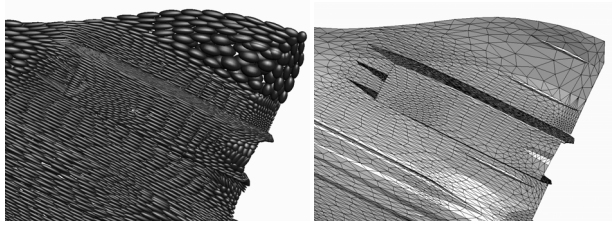


(a) Input polygonal surface (b) Constrained edges

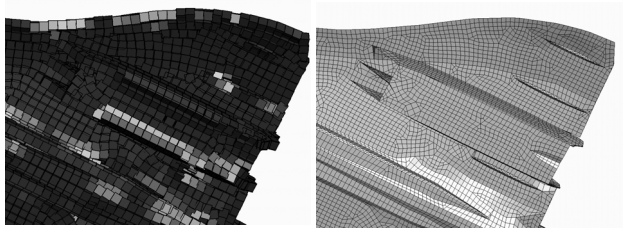
Figure 1 An example of the input polygonal surface



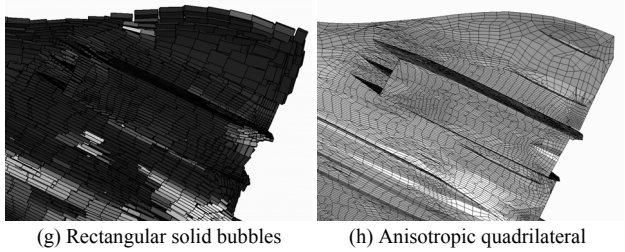
(a) Spherical bubbles (b) Isotropic triangular mesh



(c) Ellipsoidal bubbles (d) Anisotropic triangular mesh



(e) Cube bubbles (f) Isotropic quadrilateral mesh



(g) Rectangular solid bubbles (h) Anisotropic quadrilateral mesh

Figure 2 Types of volumetric cells and output meshes

In the second step, the original vertices of the input polygonal surface are deleted. This process is similar to Turk's vertex decimation algorithm. When a vertex is deleted, the method creates a cavity around the vertex, and the cavity is filled by a triangulation. When a deleted vertex is on a constrained edge, the cavity is first divided by connecting the two vertices of the cavity that are included in the constrained edge before being triangulated. Detailed explanation of this step is described in Section 6.

4. Specifying the shapes of elements

One of the input parameters to the proposed method is a function that gives desired element shape, which is described by anisotropy and element size. Anisotropy and element size can be defined by a 3x3 matrix. Since desired anisotropy and element size may vary over the domain, the function can be written as:

$$\mathbf{M}(\mathbf{x}) = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix},$$

where $\mathbf{x}$ is a three-dimensional coordinate on the input polygonal surface; and $\mathbf{u} = (u_x, u_y, u_z)$, $\mathbf{v} = (v_x, v_y, v_z)$, and $\mathbf{w} = (w_x, w_y, w_z)$ are three orthogonal vectors that define the desired element shape at $\mathbf{x}$. The length of each vector defines the desired element size in each direction. It can be a user-defined function or can be computed based on the curvature estimation or gradient of the preliminary result of the finite element analysis. More detailed discussion of the representation of anisotropy and element size is found in [16].

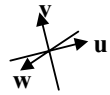
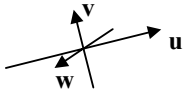
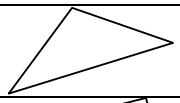
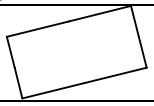
In a special case, when an isotropic triangular mesh is needed, $\mathbf{M}(\mathbf{x})$ is reduced to a scalar function $s = s(\mathbf{x})$, and $\mathbf{M}(\mathbf{x})$ can be re-written as:

$$\mathbf{M}(\mathbf{x}) = \begin{pmatrix} s(\mathbf{x})/2 & 0 & 0 \\ 0 & s(\mathbf{x})/2 & 0 \\ 0 & 0 & s(\mathbf{x})/2 \end{pmatrix},$$

where s is desired edge length.

When an isotropic quadrilateral mesh is needed, the lengths of the three vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ must be equal, and the orientation of $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ must define the orientation, or directionality, of the desired quadrilateral element. Typically, quadrilateral elements must be aligned with the boundary edges, and the method to create a directionality function for such condition from the input geometry is found in [17].

Table 1 Relation between u,v,w and desired element shapes

	Isotropic	Anisotropic
u, v, w		
Triangles		
Quadrilaterals		

The relation between the vectors and the desired element shape is shown in Table 1. For two-dimensional elements, two vectors are sufficient to define a desired element shape [18]. However, three vectors are used because one purposes of this method is to create a boundary mesh for three-dimensional mesh generation [16, 17], and it is convenient to use a form compatible with such three-dimensional mesh generation techniques. When the output mesh is not used as a boundary mesh, one of the three vectors can simply be equal to a normal vector of the boundary, and the other two vectors should define the desired element shape.

5. Packing volumetric cells

5.1. Ellipsoidal bubble and rectangular solid bubble

Since the detailed explanation of the bubble packing process can be found in [11, 13, 14, 16, 17], this section briefly explains the process.

In the first step, the proposed method packs volumetric cells – or bubbles – onto the input polygonal surface. Bubbles are moved to a stable configuration by a physically-based particle simulation, and the centers of the bubbles give good node locations for the output mesh.

The method packs spherical bubbles for an isotropic triangular mesh, ellipsoidal bubbles for an anisotropic quadrilateral mesh, cube bubbles for an isotropic quadrilateral mesh, and rectangular solid bubbles for an anisotropic quadrilateral mesh. Because a spherical bubble is a special case of an ellipsoidal bubble, and a cube bubble is a special case of a rectangular solid bubble, this section only discusses ellipsoidal bubbles and rectangular solid bubbles.

The shape of an ellipsoidal bubble at location $\mathbf{x}$ is defined by an anisotropy function $\mathbf{M}(\mathbf{x})$; it is a unit sphere with radius one transformed by $\mathbf{M}(\mathbf{x})$.

The equation of motion of ellipsoidal bubbles is written as:

$$m\ddot{\mathbf{x}} = \mathbf{F} - c\dot{\mathbf{x}},$$

where m is a mass of a bubble, $\mathbf{x}$ is a location of a bubble, $\mathbf{F}$ is a proximity-based inter-bubble force, and c is a damping coefficient [13, 14, 19].

The inter-bubble force is a function of the ratio of the actual distance l and the neutral distance l_0 between two adjacent bubbles. The distance between two bubbles is neutral when the line segment connecting the centers of the bubbles, and two surfaces of the bubbles intersect at one point as shown in Figure 3.

Bubble $\mathbf{x}_1$ is adjacent to bubble $\mathbf{x}_0$ when l/l_0 is less than 1.5. And the magnitude of the inter-bubble force acting between the two bubbles is computed as:

$$f(w) = k(1.25w^3 - 2.375w^2 + 1.125),$$

where $w = l/l_0$. The force produced by bubble $\mathbf{x}_1$ acting on bubble $\mathbf{x}_0$ is written as:

$$\mathbf{f} = f(w)\frac{\mathbf{x}_0 - \mathbf{x}_1}{\|\mathbf{x}_0 - \mathbf{x}_1\|}.$$

When $f(w)$ is positive (i.e., $0 < l/l_0 < 1$), the two bubbles repel each other, and when it is negative (i.e., $1 < l/l_0 < 1.5$), the two bubbles attract each other. $f(w)$ becomes zero at the equilibrium configuration when $l = l_0$. Hence, bubbles tend to make $l/l_0 = 1$. Since more than one bubble can be adjacent to a bubble, total inter-bubble force acting on a bubble is written as:

$$\mathbf{F} = \sum \mathbf{f}.$$

Damping force $-c\dot{\mathbf{x}}$ is added to make the system stable. The stable positions of the bubbles are obtained by integrating the differential equation by an appropriate numerical integration scheme.

A rectangular solid bubble is implemented as a bubble consisting of nine ellipsoidal bubbles, one main bubble at the center, and eight sub-bubbles at the eight corners of a rectangular solid [17]. The main bubble of a unit rectangular solid bubble is a sphere with a unit radius, and the eight sub-bubbles with radii $\sqrt{3}-1$ as shown in Figure 4. A unit rectangular solid bubble is transformed by $\mathbf{M}(\mathbf{x})$ as shown in Figure 5. A sub-bubble produces force only against main bubbles, and does not interact with another sub-bubble.

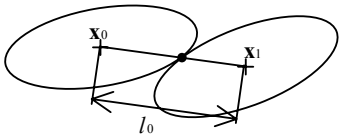


Figure 3 The neutral distance between two bubbles

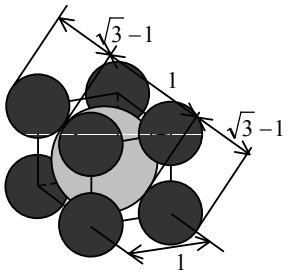


Figure 4 The unit rectangular solid bubble

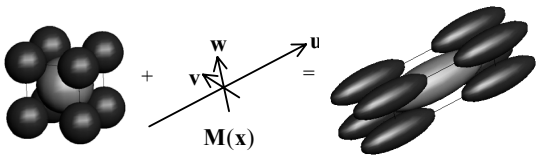


Figure 5 A unit rectangular solid bubble is transformed by $\mathbf{M}(\mathbf{x})$

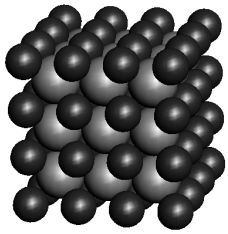


Figure 6 Rectangular solid bubbles form a rectangular crystal pattern

Closely packed rectangular solid bubbles form a crystal pattern as shown in Figure 6, and the centers of the bubbles give suitable node locations for a quadrilateral mesh.

The computational cost of the bubble packing is $O(n \log n)$ where n is the number of bubbles. Because two too-distant bubbles do not interact with each other, computing the inter-bubble force between such bubbles must be avoided to improve computational efficiency. If the bubble size is uniform over the domain, applying a rectangular grid over the domain reduces the total computational cost to $O(n)$ [9]. However, with non-uniformity and/or anisotropy, a rectangular grid becomes inefficient. To overcome this problem, the current implementation uses a range-searching algorithm called kD-tree [20]. With kD-tree, it takes $\log n$ steps to find neighbors of a bubble. Hence each iteration needs $n \log n$

steps of computations. Since the number of iterations needed for stabilizing the system can be assumed to be constant, the overall computational cost of the bubble packing process is $O(n \log n)$.

5.2. Constraining bubbles on the input polygonal surface

The proposed method first puts bubbles on the constrained vertices, the two ends of each constrained edge, and the vertices where more than one constrained edge intersects. These bubbles are called vertex bubbles, and they are fixed and do not move. The method packs bubbles on the constrained edges next, and these bubbles are called edge bubbles. Lastly, the bubbles are packed on the polygonal surface, and these bubbles are called face bubbles.

During the packing, edge bubbles must stay on the constrained edge, and face bubbles must stay on the surface. However, inter-bubble forces may push an edge bubble away from the constrained edge, and a face bubble away from the surface.

To ensure that an edge bubble stays on the constrained edge, the method projects the motion of each edge bubble on the constrained edge, and the motion of each face bubble on the surface in every time step.

Suppose that edge bubble $\mathbf{x}$ is moving on edge $\mathbf{P}_1\ \mathbf{P}_2$, which is a part of a constrained edge. If the motion of bubble $\mathbf{x}$ for the next time step calculated by the equation of motion is $\Delta\mathbf{x}$, the actual distance of motion d is calculated by taking the projection to edge $\mathbf{P}_1\ \mathbf{P}_2$:

$$d = \Delta\mathbf{x} \cdot \frac{\mathbf{P}_2 - \mathbf{P}_1}{\|\mathbf{P}_2 - \mathbf{P}_1\|}.$$

And the bubble is moved along the constrained edge by distance d. Positive d moves the bubble toward $\mathbf{P}_2$, or negative toward $\mathbf{P}_1$. If the bubble moves beyond the range of the segment of edge $\mathbf{P}_1\ \mathbf{P}_2$, the bubble will transit to the next edge segment of the same constrained edge.

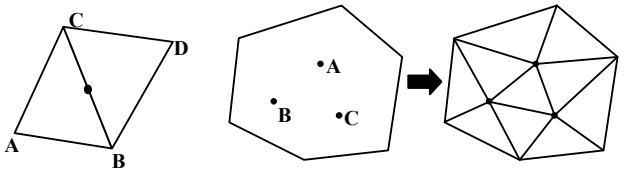
Similarly, an actual distance of motion of a face bubble is calculated by taking the projection to the polygon on which the bubble is located, and the bubble is moved along the surface by the calculated distance.

5.3. Superimposing vertices

After the bubbles are moved to stable positions, a new vertex is created at the center of each bubble so that the center does not coincide with an existing vertex, and the newly created vertices are superimposed onto the input polygonal surface.

First, the method creates vertices at the centers of bubbles that are located on an edge of the input polygonal surface, and the new vertices are superimposed to the polygonal surface. This process is not limited to edge bubbles but it includes face bubbles located on an edge, constrained or non-constrained. For example, when bubble **X** is on edge **BC**, which is used by two polygons **ABC** and **DCB** as shown in Figure 7 (a), two polygons will become **ABXC** and **DCXB**.

Second, new vertices are created at the centers of bubbles that are inside a polygon, and they are superimposed onto the input polygonal surface by a conventional triangular mesh generation technique. For example, if bubbles **A**, **B** and **C** are inside a polygon as shown in Figure 7 (b), the polygon will be triangulated as shown in Figure 7 (c).



(a) Superimposing bubble **X** on edge **BC** (b) Bubbles **A**, **B**, and **C** are inside a polygon (c) After superimposing the new vertices

Figure 7 Superimposing new vertices

6. Deleting vertices not coinciding with a center of a bubble

After new vertices are created and superimposed, the method deletes the vertices except the vertices coinciding with the center of a bubble. All the new vertices created in the first step will be retained because they coincide with the center of a bubble.

When a vertex to be deleted is not on a constrained edge, a cavity is created by deleting the polygons using the vertex, and it is then re-triangulated to fill the cavity.

For example, if vertex **X** shown in Figure 8 (a) is being deleted, a cavity is created by deleting polygons using X as shown in Figure 8 (b), and it is re-triangulated as shown in Figure 8 (c).

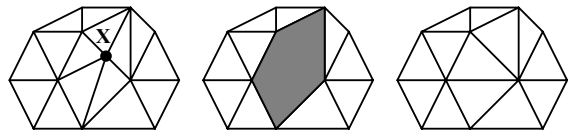
When a vertex to be deleted is on a constrained edge, the method makes some arcs by connecting some edges of the polygons using the vertex. Each arc connects the two neighboring vertices to the target vertex on the constrained edge, and it does not include the target vertex. The polygons using the target vertex are then deleted, and each arc becomes a cavity by connecting the two end points. Finally each arc is re-triangulated independently.

For example, when vertex **X** is to be deleted and **AXB** is a part of a constrained edge as shown in Figure 9 (a),

three arcs are found as shown in Figure 9 (b), and the re-triangulation will become as shown in Figure 9 (c).

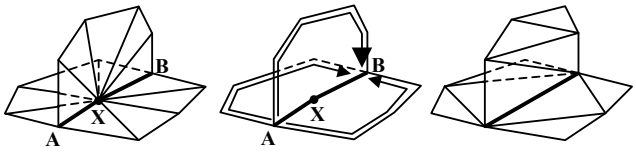
Edge-swapping is applied to improve the quality of the elements during and after this step with anisotropy taken into account.

Figure 10 illustrates the idea. Figure 10 (a) shows an input polygonal surface together with the packed bubbles. Dots are the centers of the bubbles, and stars are the vertices to be deleted. Figure 10 (b) shows an output mesh where the marked vertices are deleted.



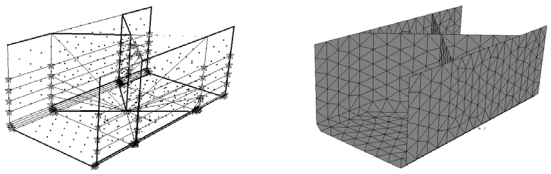
(a) **X** is to be deleted (b) Cavity (c) Re-triangulation

Figure 8 Deleting a vertex not on a constrained edge



(a) Vertex X on a constrained edge is to be deleted (b) Arcs connecting the two neighboring vertices (c) After re-triangulation

Figure 9 Deleting a vertex on a constrained edge



(a) Before deleting vertices (b) After deleting vertices

Figure 10 An example of the vertex deletion

6.1. Converting a triangular mesh to a quadrilateral mesh

When rectangular solid bubbles are packed for creating a quadrilateral mesh, the triangular mesh obtained by the above steps is converted to a quadrilateral mesh by paring some triangles [12].

The paring gives a triangle-quadrilateral mixed mesh. When an all-quadrilateral mesh is needed, each triangular element is subdivided into three quadrilaterals by adding a node on each edge and at the center of the element, and each quadrilateral is subdivided into four quadrilaterals by adding a node on each edge and at the center of the element.

7. Examples

Figure 11 demonstrates the ability of controlling anisotropy. The input polygonal surface shown in Figure 11 (a) includes non-triangular, non-quadrilateral polygons. Constrained edges shown in Figure 11 (b) are extracted by grouping faces into six groups based on the similarity of the normal direction with vectors (1,0,0), (0,1,0), (0,0,1), (-1,0,0), (0,-1,0), and (0,0,-1), and taking boundaries between the groups. Anisotropy is calculated based on curvature estimation. Figure 11 (c) shows packed ellipsoidal bubbles, and the polygonal surface is successfully remeshed into an anisotropic triangular mesh as shown in Figure 11 (d).

Figure 12 shows an example of quadrilateral mesh generation from a mesh created by a laser range scanner. When this polygonal surface was created, the laser did not reach the right side of the nose, and severely irregular elements were created as shown in Figure 12 (a). Directionality for this polygonal surface is calculated so that the elements are aligned with the constrained edges by the method described in [17], and cube bubbles are packed as shown in Figure 12 (b). Finally, the polygonal surface is remeshed into a quadrilateral mesh as shown in Figure 12 (c), and the irregularity shown in Figure 12 (a) does not exist in the output mesh.

Figure 13 shows an example of reducing the number of polygons of a very large polygonal surface. The input polygonal surface shown in Figure 13 (a) was created by taking many snap shots from different angles using a laser range scanner and merging the snap shots, and it consists of 1,211,692 polygons. The sizing function for this example is calculated based on curvature estimation, and bubbles are packed as shown in Figure 13 (b). Finally, the polygonal surface is successfully remeshed into a graded triangular mesh as shown in Figure 13 (c), and it consists of 140,443 triangles (12% of the input.)

Table 2 shows the number of polygons of the input polygonal surfaces and the number of elements of the output meshes of the examples.

	The number of polygons in the input polygonal surface	The number of elements in the output mesh
Concorde (Figure 11)	3,848	25,598
Face (Figure 12)	28,829	23,305
Camera (Figure 13)	1,211,692	140,443

Table 2 The number of polygons of each example

8. Conclusions

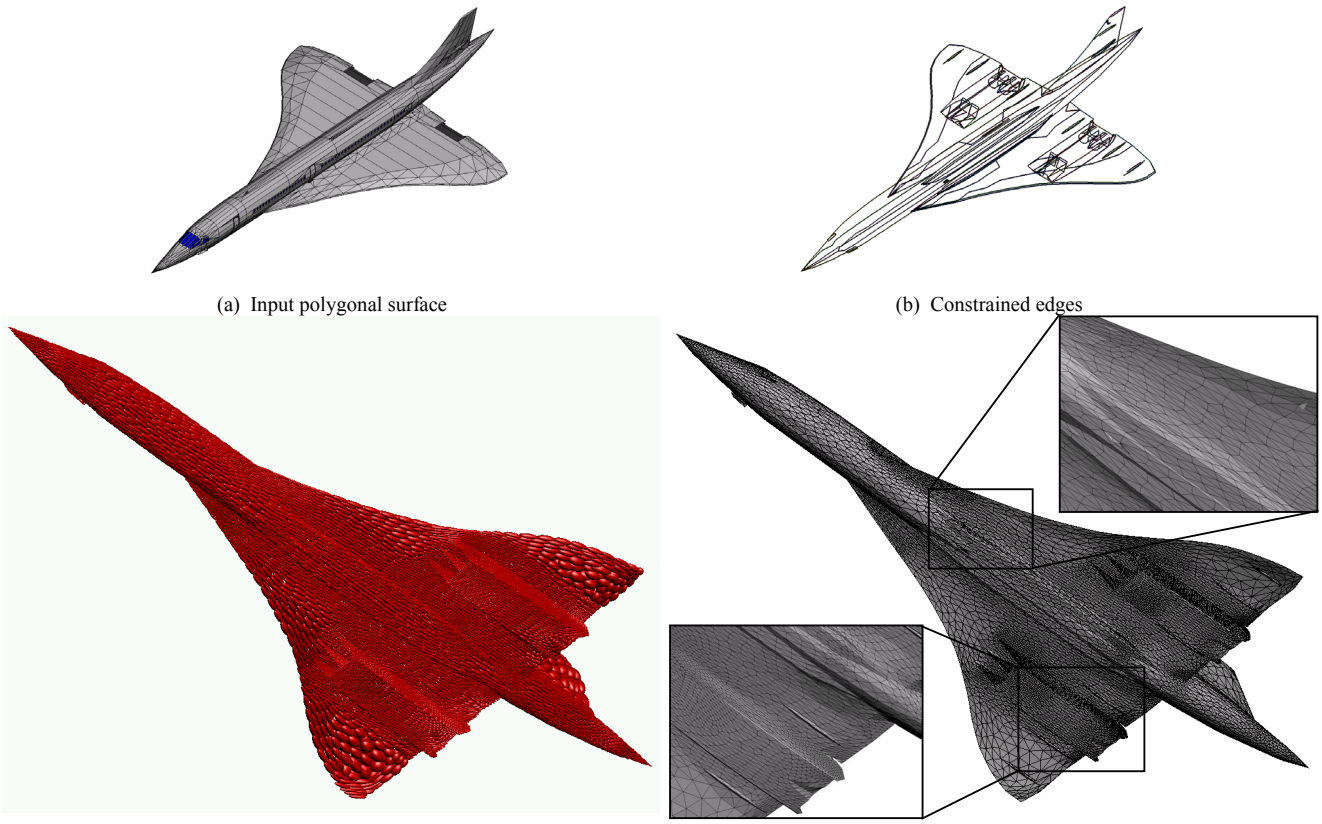
This paper has presented a new method for creating a triangular and quadrilateral mesh of an arbitrary polygonal surface while controlling the anisotropy and directionality of the mesh. The method creates a mesh in two steps, (1) packing bubbles on the input polygonal surface and creating vertices at the centers of the bubbles, and (2) deleting vertices not coinciding with a bubble. The method puts virtually no limitation on the topology of the input polygonal surface, and it is tolerant to the noise typically introduced by the measurement error of a laser range scanner. It can control element size, anisotropy and directionality flexibly and creates high quality elements.

Acknowledgement

This material is based in part on work supported under a NSF CAREER Award (No. 9985288).

References

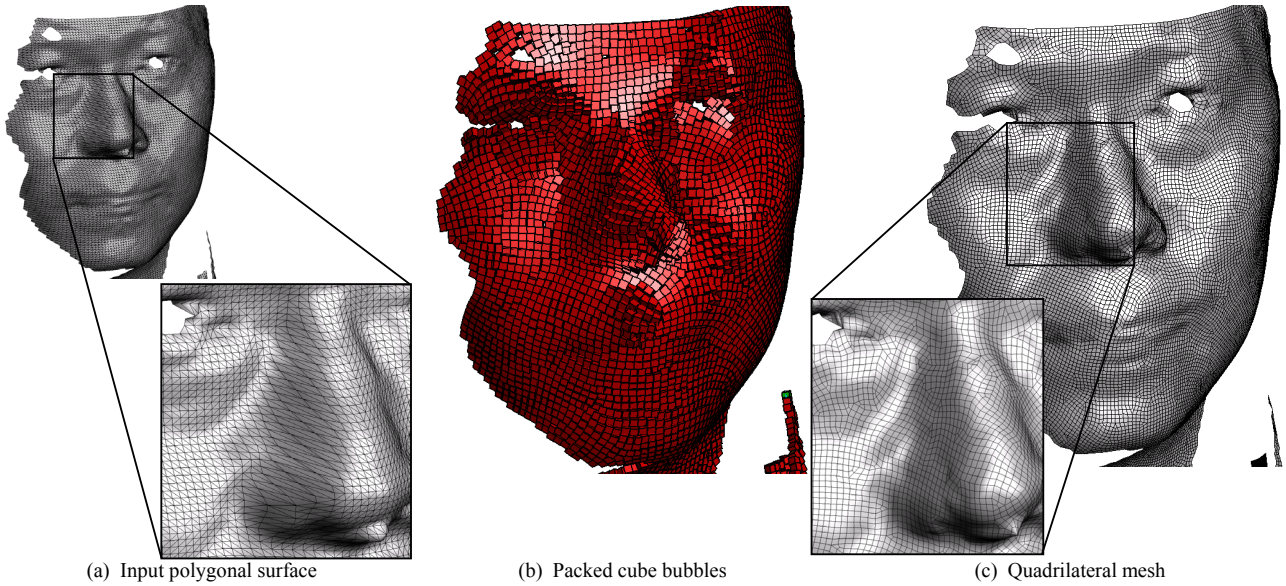
[1] G. Turk, "Re-tiling Polygonal Surfaces," Proceedings of SIGGRAPH'92, pp. 55-64, 1992.
[2] M. Garland and P. Heckbert, "Simplifying Surfaces with Color and Texture using Quadric Error Metrics," Proceedings of IEEE Visualization 98, pp. 263-269, 1998.
[3] M. Garland and P. Heckbert, "Surface Simplification Using Quadric Error Metrics," Proceedings of SIGGRAPH'97, pp. 209-216, 1997.
[4] A. Rassineux, "Surface Remeshing by Local Hermite Diffuse Interpolation", *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 31-49, 2000.
[5] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun, "Anisotropic Polygonal Remeshing", *ACM Transactions on Graphics*, vol. 22, pp. 485-493, 2003.
[6] O. Sifri, A. Sheffer, and C. Gotsman, "Geodesic-Based Surface Remeshing," Proceedings of 12th International Meshing Roundtable, pp. 189-197, 2003.
[7] J. Cabello, "Toward Quality Surface Meshing," Proceedings of 12th International Meshing Roundtable, pp. 201-214, 2003.
[8] P. Alliez, M. Meyer, and M. Desbrun, "Interactive Geometry Remeshing," Proceedings of SIGGRAPH'02, pp. 347-354, 2002.
[9] P. Heckbert, "Fast Surface Particle Repulsion," Carnegie Mellon University, CMU Computer Science Tech. Report CMU-CS-97-130 1997.
[10] V. Surazhsky, P. Alliez, and C. Gotsman, "Isotropic Remeshing of Surfaces: A Local Parameterization Approach," Proceedings of 12th International Meshing Roundtable, pp. 215-224, 2003.
[11] K. Shimada, J.-H. Liao, and T. Itoh, "Quadrilateral Meshing with Directionality Control through the Packing of Square Cells," Proceedings of 7th International Meshing Roundtable, pp. 61-75, 1998.
[12] T. Itoh, K. Shimada, K. Inoue, A. Yamada, and T. Furuhata, "Automated Conversion of 2D Triangular Mesh into Quadrilateral Mesh with Directionality Control," Proceedings of 7th International Meshing Roundtable, pp. 77-86, 1998.
[13] K. Shimada and D. C. Gossard, "Automatic Triangular Mesh Generation of Trimmed Parametric Surfaces for Finite Element Analysis", *Computer Aided Geometric Design*, vol. 15, pp. 199-222, 1998.
[14] K. Shimada, A. Yamada, and T. Itoh, "Anisotropic Triangulation of Parametric Surfaces via Close Packing of Ellipsoids", *International Journal of Computational Geometry and Applications*, vol. 10, pp. 417-440, 2000.
[15] M. Garland, A. Willmott, and P. Heckbert, "Hierarchical Face Clustering on Polygonal Surfaces," Proceedings of Symposium on Interactive 3D Graphics, pp. 49-58, 2001.
[16] S. Yamakawa and K. Shimada, "Anisotropic Tetrahedral Meshing via Bubble Packing and Advancing Front", *International Journal for Numerical Methods in Engineering*, vol. 57, pp. 1923-1942, 2003.
[17] S. Yamakawa and K. Shimada, "Fully-Automated Hex-Dominant Mesh Generation with Directionality Control via Packing Rectangular Solid Cells", *International Journal for Numerical Methods in Engineering*, vol. 57, pp. 2099-2129, 2003.
[18] F. J. Bossen and P. S. Heckbert, "A Pliant Method for Anisotropic Mesh Generation," Proceedings of 5th International Meshing Roundtable, pp. 63-74, 1996.
[19] S. Yamakawa and K. Shimada, "High Quality Triangular/Quadrilateral Mesh Generation of a 3D Polygon Model via Bubble Packing," Proceedings of 7th US National Congress on Computational Mechanics, 2003.
[20] R. Sedgewick, "Range Searching," in *Algorithms in C++*: Addison-Wesley, 1992, pp. 373-388.



(a) Input polygonal surface (b) Constrained edges

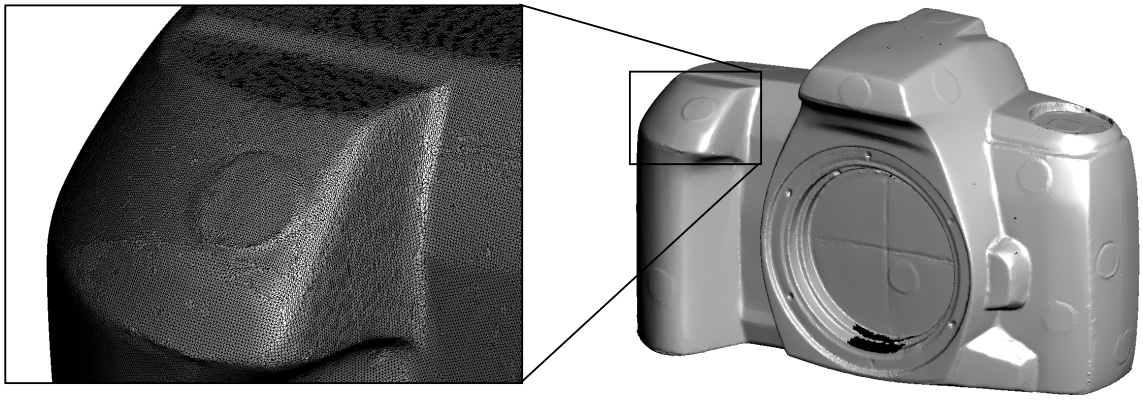
(c) Packed ellipsoidal bubbles (d) Anisotropic mesh

Figure 11 Anisotropic triangular mesh of Concorde airplane

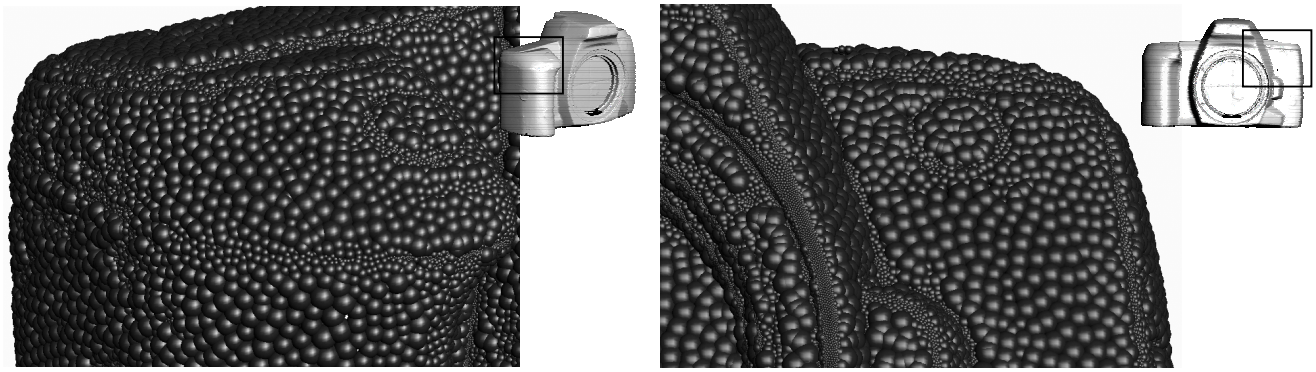


(a) Input polygonal surface (b) Packed cube bubbles (c) Quadrilateral mesh

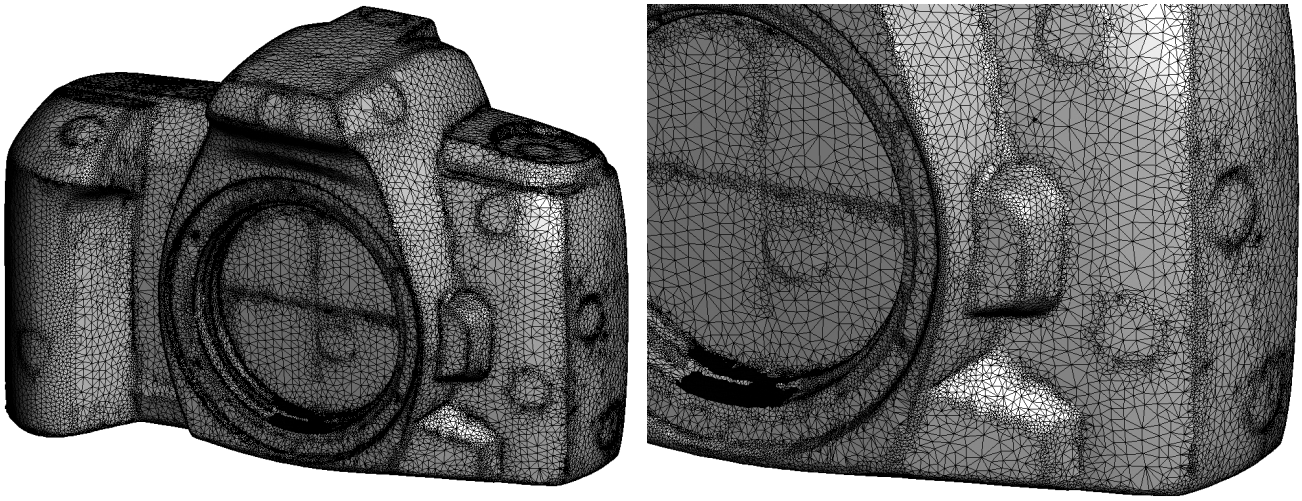
Figure 12 A quadrilateral mesh of a face



(a) Input polygonal surface



(b) Bubble packing



(c) Output mesh

(d) Output mesh (close-up)

Figure 13 A graded triangular mesh of a camera