Submitted to: ASME Journal of Mechanical Design, 2001

QUAD-LAYER: LAYERED QUADRILATERAL MESHING OF NARROW TWO-DIMENSIONAL DOMAINS BY BUBBLE PACKING AND CHORDAL AXIS TRANSFORMATION [◊]

Soji Yamakawa
Graduate Research Assistant
Department of Mechanical Engineering
Carnegie Mellon University

Kenji Shimada[*]
Associate Professor
Department of Mechanical Engineering
Carnegie Mellon University

ABSTRACT

This paper presents a computational method for quadrilateral meshing of a thin, or narrow, two-dimensional domain for finite element analysis. The proposed method creates a well-shaped single-layered, multi-layered, or partially multi-layered quadrilateral mesh. Element sizes can be uniform or graded. A high quality, layered quadrilateral mesh is often required for finite element analysis of a narrow two-dimensional domain with a large deformation such as in the analysis of rubber deformation or sheet metal forming. Fully automated quadrilateral meshing is performed in two stages: (1) extraction of the skeleton of a given domain by discrete chordal axis transformation, and (2) discretization of the chordal axis into a set of line segments and conversion of each of the line segments to a single quadrilateral element or multiple layers of quadrilateral elements. In each step a physically-based computational method called bubble packing is applied to discretize a curve into a set of line segments of specified sizes. Experiments show that the accuracy of a large-deformation FEM analysis can be significantly improved by using a well-shaped quadrilateral mesh created by the proposed method.

1 Introduction

This paper describes a new computational method, called Quad-Layer, that creates a high-quality quadrilateral mesh of a thin, or narrow, two-dimensional domain such as the cross-section of a tire, a thin metal part, or a formed metal joint. The proposed method can control the sizes and anisotropy of quadrilateral elements. A quadrilateral mesh created by Quad-Layer is particularly suitable for the finite element analysis involved with the large deformation of a narrow two-dimensional domain.

One example of such a geometric domain is illustrated in Figure 1 (a), and its desired quadrilateral finite element discretizations can be single-layered as shown in Figure 1 (b), multi-layered as shown in Figure 1 (c), or partially multi-layered as shown in Figure 1 (d). A thin, narrow two-dimensional region is bounded by two lengthwise boundary curves Γ_1 and Γ_2 and two widthwise boundary curves Π_1 and Π_2. In the rest of this paper, we use the term 'length' to refer to the element size in the lengthwise direction shown in Figure 1 (a), and the term 'width' for the element size in the widthwise direction shown in Figure 1 (a).

High quality quadrilateral meshing of this type of domain is often required in the finite element analysis of various artifacts including both narrow objects and the thin cross-sections of flat surfaces. Some examples include:

- Extruded parts - In extrusion a round heated billet is placed in a chamber of a large press and forced through a die by a hydraulic ram, forming a long profile shape with a constant cross-section.
- Sheet metal parts - In continuous roll forming, metal sheet is fed through a series of top and bottom rollers, gradually bending and forming the sheet to a target profile. The attainable shapes are somewhat similar to extrusions except that the profile is limited to the original thickness of the sheet.

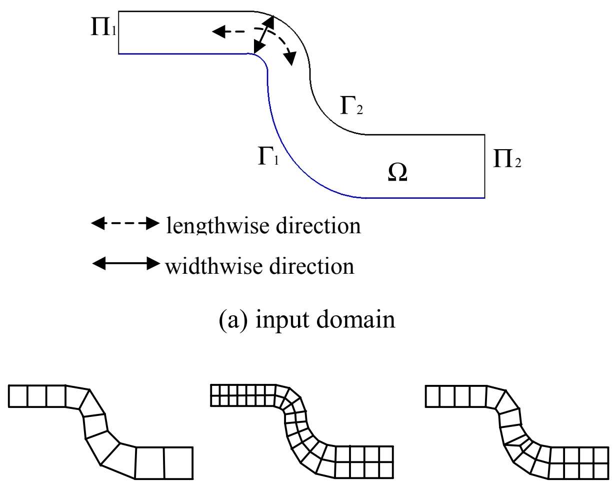
[◊] This paper is an extended and more complete version of the paper accepted for presentation at ASME/DETC/DAC 2001 [19].
[*] Corresponding author:
Kenji Shimada, Department of Mechanical Engineering, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213-3890, U.S.A.
Tel: (412)268-3614, Fax: (412)268-3348, Email: shimada@cmu.edu

- Fastening of sheet metals - The formation of a formed metal joint involves a deformation of a narrow cross-section. In order to fasten two sheet metals, their edges are lined up and rolled together so that they enfold each other.
- Rubber parts - A tire consists of many thin layers of different materials, including layers of rubber, and each of them has a narrow cross-section.
- Paint and coating - In order to tolerate fractures or cracks caused by heat, moisture or other external factors, a car body is covered by layers of coatings. Because these layers are very thin, they often have to be meshed into a layered mesh when a finite element analysis is performed.

For these types of two-dimensional domains, conventional automatic quadrilateral meshing methods, such as advancing front [1-4] and triangle pairing [5, 6], do not perform well. The conventional approaches usually distribute nodes uniformly along the boundary curves, and this yields ill-shaped elements in regions where the curvature of the domain boundary is high or where the curvature changes rapidly. In order to avoid these ill-shaped elements, the user needs to adjust manually the boundary node distribution.

The proposed Quad-Layer method creates well-shaped, layered quadrilateral elements automatically over a narrow two-dimensional domain in two stages: (1) extraction of the skeleton of a given domain by discrete chordal axis transformation, and (2) discretization of the chordal axis into a set of line segments and conversion of each of the line segments to a single quadrilateral element or multiple layers of quadrilateral elements. In both stages a physically-based computational method, called bubble packing, is applied to discretize a curve into a set of line segments of specified sizes.

The rest of this paper is organized as follows. Section 2 describes the problem statement. Section 3 discusses related research, and an overview of the proposed method is shown in Section 4. The key technique of one-dimensional meshing called 'one-dimensional bubble packing' is discussed in Section 5, followed by the extraction of the chordal axis and quadrilateral mesh generation in Sections 6 and 7. After some meshing results are shown in Section 8, Section 9 shows the results of non-linear, large deformation finite element analyses to verify that a quadrilateral mesh created by Quad-Layer improves the solution accuracy of such analyses.



(a) input domain

(b) single-layered quad mesh

(c) multi-layered quad mesh

(d) partially multi-layered quad mesh

Figure 1 An example of an input narrow two-dimensional domain and three types of output layered quadrilateral meshes

2 Problem Statement

The proposed method takes as input a narrow two-dimensional domain Ω and a desired element length distribution $d(\mathbf{x})$, where $\mathbf{x}$ is a two-dimensional coordinate, and generates a quadrilateral mesh automatically. The domain Ω is a two-dimensional region enclosed by four curves $\Gamma_1, \Gamma_2, \Pi_1$ and Π_2 as shown in Figure 1 (a). Γ_1 and Γ_2 are boundary curves in the lengthwise direction, and Π_1 and Π_2 in the widthwise direction. For a narrow domain Γ_1 and Γ_2 are distinctively longer than Π_1 and Π_2. These boundary curves can be given as either parametric curves or polylines.

The output quadrilateral mesh is a single-layered mesh as shown in Figure 1 (b), a multi-layered mesh as shown in Figure 1 (c), or a partially multi-layered mesh as shown in Figure 1 (d). When the width of Ω is smaller than a user-specified width threshold everywhere in Ω, the output will be a single-layered mesh. In contrast, if the width of Ω is larger than the user-specified threshold in any part of Ω, the mesh will be either a multi-layered mesh or a partially multi-layered mesh.

The element length distribution function, $d(\mathbf{x})$, specifies the distribution of desired element lengths, or element sizes in the lengthwise direction. There are several possible ways to define this function:

- Uniform : All elements should have the same length.
- Uniform aspect ratio : All elements should have the same aspect ratio (the ratio of the length and width).
- Curvature-based : Desired element lengths are given based on the curvature of the lengthwise boundary curves, Γ_1 and Γ_2. Element sizes should be smaller in a high curvature region in order to minimize the geometric approximation error.
- Analysis-based : In adaptive remeshing, desired mesh lengths are defined based on a previous finite element analysis result. Smaller element lengths should be defined in a region where the solution changes dramatically.
- User-specified : The user specifies element sizes manually at selected points in the domain. The complete definition of $d(\mathbf{x})$ is then obtained by interpolating the specified element lengths at these points.

3 Related Work

Although there are several approaches proposed for quadrilateral meshing of a general two-dimensional domain, they often generate a poor quality mesh when applied to a narrow domain.

In the advancing front algorithm or its variations, quadrilateral elements are created one by one, or layer by layer, starting from the boundary and moving toward the interior of a domain. Blacker and Stephanson propose an algorithm called ‘Paving’ to create a quadrilateral mesh for a two-dimensional domain [1]. The method creates quadrilateral elements layer by layer. Cass and Benzley propose a generalized paving algorithm for a three-dimensional surface [2]. White presents a modified paving algorithm [7] that creates quadrilateral elements one by one rather than layer by layer. Owen et al. propose an advancing-front algorithm that converts a triangular mesh into a quadrilateral mesh [3]. Thompson and Soni propose an advancing-front algorithm that creates a quad-dominant or hex-dominant mesh [4].

Another type of approach to quadrilateral meshing is node placement and connection. Shimada et al. propose a method that creates a quad-dominant mesh by packing square cells in the domain [5]. Itoh et al. present an algorithm that converts a triangular mesh into a quadrilateral mesh by pairing triangles [6]. Their method first creates ideal node locations for the quadrilateral mesh, then generates a Delaunay triangulation based on the node locations, and finally converts the triangular mesh into a quadrilateral mesh. Viswanath et al. propose an algorithm that creates an anisotropic quadrilateral mesh by packing rectangular cells [8].

There are several two-dimensional meshing methods that make use of medial axis transformation. Tam and Armstrong [9] propose an algorithm that creates a quadrilateral mesh, and Gürsoy and Patrikalakis [10] propose a method for creating a triangular mesh, both based on medial axis transformation. These methods subdivide an original domain into sub-domains, and then mesh each sub-domain by mapping templates. Quadros et al. present another method that uses medial axis transformation [11]. Their algorithm combines medial axis transformation and an advancing front algorithm to create a quadrilateral mesh.

Typically, existing quadrilateral meshing algorithms do not create a layered quadrilateral mesh of good quality for a narrow two-dimensional domain. They encounter difficulty in the node spacing on the lengthwise boundary curves, Γ_1 and Γ_2. To produce a single-layered all-quadrilateral mesh, the number of nodes placed on Γ_1 must be identical to the number of nodes placed on Γ_2. If n nodes are distributed uniformly on Γ_1 and Γ_2, however, distorted elements are generated in regions where the curvature of the boundary changes rapidly. For example, in Figure 2, placing eleven nodes uniformly on Γ_1 and Γ_2 yields several ill-shaped elements in the middle region where the curvature of the boundary changes rapidly.

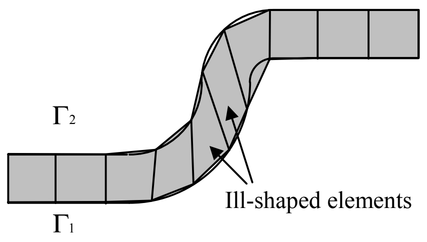


Figure 2 Ill-shaped elements caused by uniformly distributing boundary nodes

4 Quad-Layer Method Overview

The Quad-Layer method creates a well-shaped layered quadrilateral mesh by using a physically-based one-dimensional meshing method, called bubble packing, and discrete chordal axis transformation [12]. The method creates a quadrilateral mesh in six steps, also illustrated in Figure 3:

Step 1. Discretize the two lengthwise boundary curves, Γ_1 and Γ_2, into a set of line segments.

Step 2. Based on the discretized boundary, create a constrained Delaunay triangulation [13, 14].

Step 3. Extract a chordal axis, an approximated medial axis, by discrete chordal axis transformation [12].

Step 4. Re-discretize the chordal axis according to a prescribed element length distribution function.

Step 5. Create a single-layered quadrilateral mesh.

Step 6. Convert the mesh into a multi-layered mesh, if necessary, according to a user-specified width threshold.

These procedures are grouped into two major stages. The first stage, consisting of Steps 1-3, extracts the chordal axis, and the second stage, consisting of Steps 4-6, creates a quadrilateral mesh. In Steps 1 and 4, a one-dimensional discretization algorithm called ‘bubble packing’ is applied. This algorithm packs circular cells, or bubbles, on a given curve, and the bubbles are moved to a stable configuration by dynamic simulation. After the bubbles become stable, their centers give the locations of nodes by which the curve is discretized. The distances between nodes are controlled by a prescribed element length distribution function. Details of the one-dimensional bubble packing are presented in Section 5.

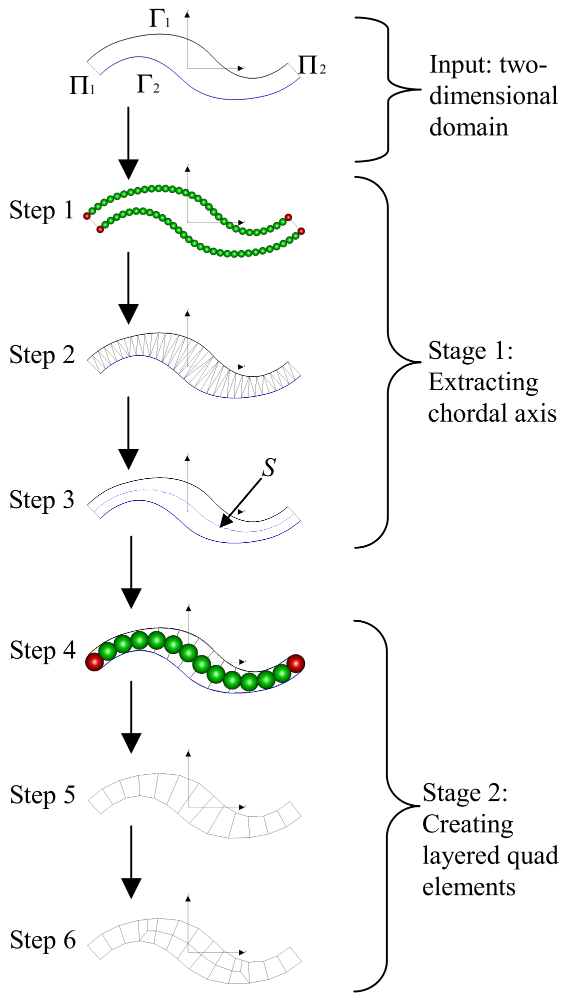

Γ_1
Π_1
Γ_2
Π_2
Input: two-dimensional domain
Step 1
Step 2
Step 3
S
Stage 1: Extracting chordal axis
Step 4
Step 5
Step 6
Stage 2: Creating layered quad elements

Figure 3 Outline of the Quad-Layer method

5 One-Dimensional Bubble Packing

The bubble packing method was originally developed by Shimada et al. for triangular meshing [15, 16] and was later expanded to anisotropic tetrahedral meshing [17]. For quadrilateral meshing of a narrow two-dimensional domain, a one-dimensional version of bubble packing is applied.

The purpose of one-dimensional bubble packing is to find a set of ideally spaced nodes on the boundary curves, Γ_1 and Γ_2, in Stage 1 and on the chordal axis in Stage 2. In this process, spherical bubbles are packed on a given curve $C(s)$, $0 \le s \le 1$, and are moved to a stable configuration by dynamic simulation. $C(s)$ can be a series of piecewise line segments, in which case the line segments must be parameterized before the bubble packing process. The diameters of the bubbles are controlled by the element length distribution function, $d(C(s))$. If an appropriate number of bubbles is closely packed on the curve according to the element length distribution function, the bubbles will move to a stable configuration in which all the proximity-based inter-bubble forces are balanced. After the bubbles become stable their centers give the locations of nodes. Step 1 and Step 4 in Figure 3 show the bubble packing on the boundary curves Γ_1 and Γ_2 and on the chordal axis respectively. The rest of this section discusses how this close packing of bubbles is achieved.

5.1 Computation of the Motion of Bubbles

To run the dynamic simulation we first derive the equation of motion that governs the dynamic behavior of bubbles. A standard numerical integration scheme, such as Euler's method or the 4th order Runge-Kutta method, is then applied to simulate the motion of the bubbles. We derive the second order differential equation that governs the motion of the bubbles by assuming proximity-based inter-bubble forces, a point mass for each bubble, and the effect of viscous damping.

Forces acting on bubbles are approximated by a mass-spring-damper model. In this model, two kinds of forces act on a bubble. One of the forces is a proximity-based inter-bubble force analogous to defining a non-linear spring between a pair of adjacent bubbles. The other force is due to viscous damping. In this mass-spring-damper model, each bubble has two state variables, position and velocity, and two attributes, mass and a damping coefficient. The state variables of the *i*th bubble are denoted as:

$$\mathbf{x}_i \ : \text{Position of bubble } i$$

$$\dot{\mathbf{x}}_i \ : \text{Velocity of bubble } i$$

We assume that all bubbles have the same mass *m* and the same damping coefficient *c*, that is:

$$m_i = m$$

$$c_i = c$$

In order to compute the inter-bubble force between bubble *i* and bubble *j* in Figure 4, we need to find the current length and the neutral length of the spring. The current length l of the spring between bubble *i* and bubble *j* is computed as:

$$l = \|\mathbf{x}_i - \mathbf{x}_j\|,$$

where $\mathbf{x}_i$ is the center of bubble *i*, $\mathbf{x}_j$ is the center of bubble *j*, and $\|\cdot\|$ denotes the L_2 Euclidean norm. On the other hand, the neutral length of the spring l_0 is defined as:

$$l_0 = r_i + r_j,$$

where r_i is the radius of bubble *i* and r_j the radius of bubble *j*. Because the diameters of the bubbles are controlled by the element length distribution function $d(\mathbf{x})$, l_0 can be written as:

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

After l and l_0 are computed, the inter-bubble force is defined as a function of the ratio of l and l_0. Two adjacent bubbles either attract each other, repel each other or do not interact, depending on the value of l/l_0. When $l/l_0 = 1$, the bubbles are at a stable distance and do not either attract or repel each other. When $l/l_0 < 1$, they repel each other and when $1 < l/l_0 < 1.5$, they attract each other. When $1.5 < l/l_0$, again they do not interact because they are located too far away from each other.

As in the previous Bubble Mesh methods, we define the force of the non-linear spring using a cubic function:

$$f(w) = \begin{cases} k(1.25w^3 - 2.375w^2 + 1.125) & \text{for } 0.0 \le w \le 1.5 \\ 0 & \text{otherwise} \end{cases},$$

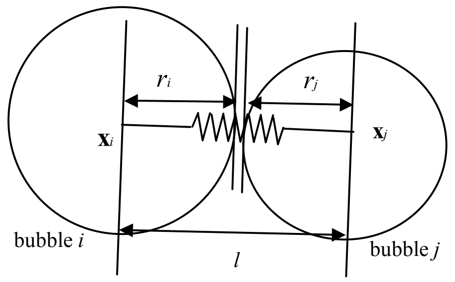


Figure 4 Non-linear spring

where w is the ratio of l and l_0, that is $w = l / l_0$, and k is the spring constant. Since multiple bubbles can be adjacent to a given bubble *i*, the total inter-bubble force acting on the *i*th bubble can be computed by:

$$\mathbf{f}_i = \sum_j f_{ij} \frac{\mathbf{x}_i - \mathbf{x}_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} ,$$

where

$$f_{ij} = \begin{cases} \text{force applied to bubble } i \text{ by bubble } j & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} .$$

In order to make a system of bubbles converge to a stable configuration, damping must be added to the system. We define a damping force acting on bubble i to be proportional to the velocity of the bubble:

$$-c\dot{\mathbf{x}}_i .$$

Finally, we can construct the second order ordinary differential equation that governs the motion of the bubbles as:

$$m\ddot{\mathbf{x}}_i + c\dot{\mathbf{x}}_i = \mathbf{f}_i .$$

A numerical integration scheme such as Euler's method or the 4th order Runge-Kutta method can be used to solve this ordinary differential equation.

5.2 Population Control

Another important procedure during bubble packing is an adaptive population control of the bubbles. The purpose is to adjust the number of bubbles so that there are no significant gaps or overlaps in the final packed configuration. Bubbles should be closely packed in order to obtain well-distributed nodes. If the number of bubbles is too small, large gaps exist between bubbles, and the distance between bubbles will become larger than the ideal distance specified by the element length distribution function $d(C(s))$. In contrast, if the number of bubbles is too large, the bubbles will have large overlaps, making the distance between bubbles shorter than the ideal distance. To obtain well-distributed node points, it is important that the number of bubbles be adjusted adaptively during dynamic simulation.

The optimal number of bubbles, N, is a function of the curve geometry, C, and the specified element length distribution function, d, that is:

$$N = N(C, d) .$$

If the curve is straight and if $d(C(s))$ is constant, i.e., $d(C(s)) = d_0$, an appropriate number of bubbles can be estimated by:

$$N = \frac{L(C)}{d_0},$$

where

$$\begin{array}{ll} L(C) & : \text{Length of the curve} \\ d_0 & : \text{Diameter of a bubble}. \end{array}$$

Although we apply this formula to estimate an appropriate number of initial bubbles, when the geometry is highly curved and the specified element size distribution function is not constant, the formula gives only a rough estimate of an appropriate number of bubbles.

Because the number of initial bubbles placed on a piece of a curve is not always optimal, the number of bubbles needs to be adjusted adaptively during dynamic simulation by adaptive population control. We take advantage of the computation of the inter-bubble force to find over-populated and under-populated regions. If a bubble is receiving repelling forces, this indicates that it is in an over-populated region. In contrast, if a bubble is receiving attracting forces, this signifies that it is in an under-populated region. Bubbles are deleted from over-populated regions and are added in under-populated regions once every few iterations.

5.3 Constraining Bubbles on a Curve

Since all nodes must be created on a target curve, the motion of bubbles must be constrained to the curve. Without a mechanism to enforce this constraint bubbles may move away from a target curve during dynamic simulation because inter-bubble forces usually have a component perpendicular to the tangential direction of the curve.

In order to constrain all bubbles on a curve, the locations of bubbles should be specified in parametric space instead of Cartesian space. The program computes an approximated displacement of bubbles in parametric space from their displacement in Cartesian space by the following method. Suppose the i th bubble is located at $\mathbf{x}_i = C(s_0)$, as shown in Figure 5. The tangential vector $C^s(s_0)$ at this location is:

$$C^s(s_0) = \begin{pmatrix} \dfrac{\partial C_x(s_0)}{\partial s} \\ \dfrac{\partial C_y(s_0)}{\partial s} \end{pmatrix}.$$

We first compute the unconstrained displacement $\Delta\mathbf{x}_i$ in Cartesian space. Then, assuming that the movement is small, and that $C^s(s)$ does not change before and after the iteration, the displacement in parametric space Δs is computed approximately by taking the ratio of the projected length of $\Delta\mathbf{x}_i$ on the tangential vector, $C^s(s)$, and the length of $C^s(s)$:

$$\Delta s = \frac{v_s}{\left\| C^s(s_0) \right\|},$$

where

$$v_s = \frac{\Delta\mathbf{x}_i \cdot C^s(s_0)}{\left\| C^s(s_0) \right\|}.$$

The new, constrained position of the bubble, $\mathbf{x}_i'$ is then obtained as:

$$\mathbf{x}_i' = C(s_0 + \Delta s)$$

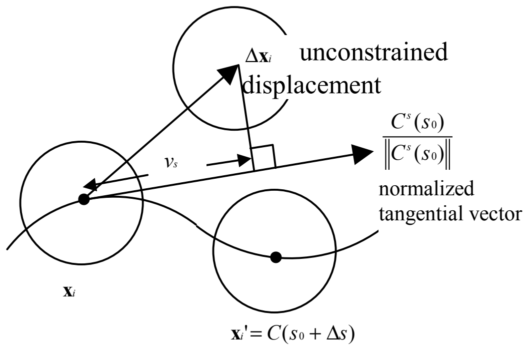

$\Delta\mathbf{x}_i$ unconstrained displacement
$\frac{C^s(s_0)}{\|C^s(s_0)\|}$
normalized tangential vector
v_s
$\mathbf{x}_i$
$\mathbf{x}_i' = C(s_0 + \Delta s)$

Figure 5 Constraining a bubble on a curve

6 Extraction of a Chordal Axis

In this section, we discuss the first stage of the proposed method, extraction of a chordal axis. Our chordal axis extraction method is a slight variation of Prasad's method [12], called discrete chordal axis transformation. Since this algorithm assumes a triangulated domain as input, the boundary of an input domain is first discretized using one-dimensional bubble packing, and the domain is triangulated by constrained Delaunay triangulation. Discrete chordal axis transformation is simple and efficient to compute. A chordal axis is easily extracted from a constrained Delaunay triangulation, for which many efficient algorithms are readily available.

As also illustrated in Figure 3, the proposed method extracts the chordal axis in the following three steps:

(1) Pack bubbles on boundary curves Γ_1 and Γ_2 in order to discretize the curves into polylines.

(2) Construct a constrained Delaunay triangulation.

(3) Extract the chordal axis S based on the constrained Delaunay triangulation.

In the first step, bubbles are packed on Γ_1 and Γ_2. Bubbles are moved to a stable configuration by the method described in Section 5. Since the purpose of this step is to find well-spaced node locations for a constrained Delaunay triangulation, the diameters of the packed bubbles must be small enough so that the triangulation approximates the original domain Ω well. In the current implementation, the diameters of the bubbles are specified to be 1/10 to 1/3 of the minimum width of the domain.

Next, after the bubbles become stable, node points are placed at the centers of the bubbles, and a constrained Delaunay triangulation is constructed using the node points. Algorithms to construct a constrained Delaunay triangulation are already available [13, 14], and they are typically robust and efficient for two-dimensional domains.

The chordal axis S is then extracted by joining the centers of edges that connect two nodes, one on Γ_1 and the other on Γ_2. If bubbles are tightly packed and their diameters are reasonably small, S forms an approximated skeleton of the original domain, Ω. Our chordal axis extraction is a slight variation of Prasad's algorithm presented in [12]. In a discretized polygonal domain, Prasad's algorithm connects only the centers of interior edges, i.e., edges that are not a part of the domain boundary, as illustrated in Figure 6. Because points on the boundary edges are not included in Prasad's chordal axis, line segments **AB** and **CD** in Figure 6 are not a part of Prasad's chordal axis. For our purpose, however, these missing ends of the chordal axis, **AB** and **CD**, must be included so that the chordal axis approximates a medial axis of the two length-wise boundary curves Γ_1 and Γ_2. This modification is necessary for the next stage of the method, described in Section 7.

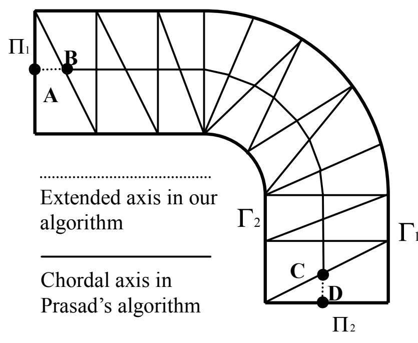

Π_1
B
A
Extended axis in our algorithm
Chordal axis in Prasad's algorithm
Γ_2
Γ_1
C
D
Π_2

Figure 6 Variation of Prasad's algorithm

7 Creating a Layered Quadrilateral Mesh

After extracting the chordal axis, S, a layered quadrilateral mesh is created in the following three steps:

(1) Pack bubbles on the chordal axis. The sizes of bubbles are adjusted according to a given length distribution function, $d(\mathbf{x})$.

(2) Create quadrilateral elements based on the locations of the centers of the packed bubbles.

(3) Apply Laplacian smoothing to further improve the element quality.

After the bubbles are moved to a stable configuration by the method described in Section 5, quadrilateral elements are created based on the locations of the bubbles. Here, we denote the number of bubbles as n_b and the center of the j th bubble as $\mathbf{x}_j$. Bubbles are sorted by their parametric position on the chordal axis in ascending order; if a position on the chordal axis is written as $S(s)$, $0 \le s \le 1$, and if the center of the j th bubble in parametric space is $\mathbf{s}_j$, then $s_j < s_{j+1}$.

For each of the bubble centers, $\mathbf{x}$, straight line $L_S(\mathbf{x})$ is defined such that it intersects the chordal axis, S, at point $\mathbf{x}$ and perpendicular to the tangent vector of the axis. Two nodes $N_{1,j}$ and $N_{2,j}$ are then computed for the j th bubble center by finding the intersections of $L_S(\mathbf{x})$ and the boundary curves, Γ_1 and Γ_2. After all the nodes are placed, the j th quadrilateral element is formed by connecting $N_{1,j}$, $N_{2,j}$, $N_{2,j+1}$ and $N_{1,j+1}$ as shown in Figure 7.

After creating elements we apply a post-process of mesh smoothing in order to improve the quality of the mesh. If a part of the domain Ω has high curvature, and if the user specifies $d(\mathbf{x})$ such that the element length becomes very short, some elements may become inverted as shown in Figure 8. In such a case, we apply several iterations of Laplacian smoothing in order to re-distribute the boundary nodes and correct the inverted elements.

If some elements are significantly wider than other elements, they can be converted into two layers, making the width of mesh elements more uniform. For example, in the mesh shown in Figure 9 (a), elements A, B, C, D and E are wider than other elements. In order to identify elements that need to be divided to form two layers, all the mesh elements are classified into three groups: (1) narrow element, (2) transition element and (3) wide element, as illustrated in Figure 10. The j th element, consisting of nodes $N_{1,j}$, $N_{2,j}$, $N_{2,j+1}$ and $N_{1,j+1}$, is classified by comparing l_{ej} and $l_{e(j+1)}$ with l_{thr}, where l_{ej} is the length of edge $N_{1,j}\,N_{2,j}$, $l_{e(j+1)}$ the length of edge $N_{2,j+1}\,N_{1,j+1}$, and l_{thr} a user-specified width-threshold. If both l_{ej} and $l_{e(j+1)}$ are shorter than l_{thr}, the element is classified as a narrow element. If both l_{ej} and $l_{e(j+1)}$ are longer than l_{thr}, the element is classified as a wide element. If only one of l_{ej} or $l_{e(j+1)}$ is longer than l_{thr}, the element is classified as a transition element. In Figure 9 (a) elements B, C and D are classified as wide elements, and elements A and E are classified as transition elements. After classifying elements, we split wide elements into two layers, and we apply the template shown in Figure 9 (b) to transition elements. This yields a partially two-layered mesh as shown in Figure 9 (c).

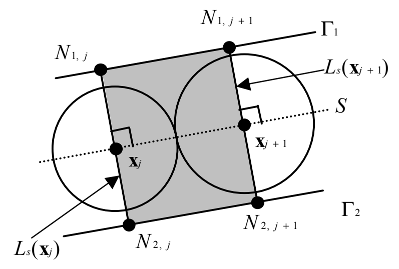


Figure 7 Quadrilateral element generation

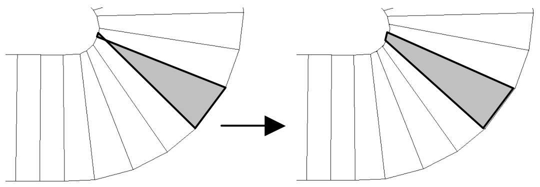


Figure 8 Correcting an inverted element by Laplacian smoothing

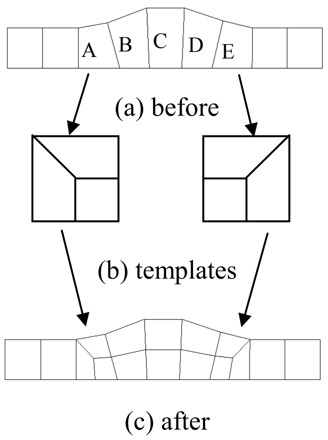


Figure 9 Converting part of a mesh into two layers

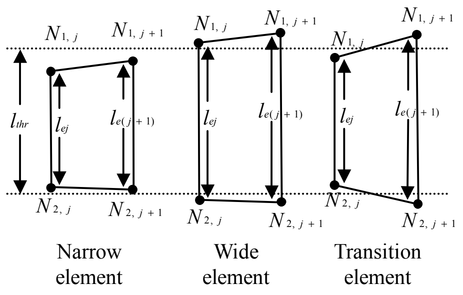


Figure 10 Classification of elements

8 Quadrilateral Meshing Results

In this section, some meshing results of the proposed method are presented for both single and partially double-layered cases, and for input domains both with and without branching geometry.

8.1 Multi-Layered Meshing

Figure 11 shows three different quadrilateral meshes of a tire cross-section: a single-layer mesh of uniform aspect ratio shown in Figure 11 (a), a single-layer mesh of uniform element length shown in Figure 11 (b), and a partially double-layered mesh of uniform element length and width shown in Figure 11 (c).

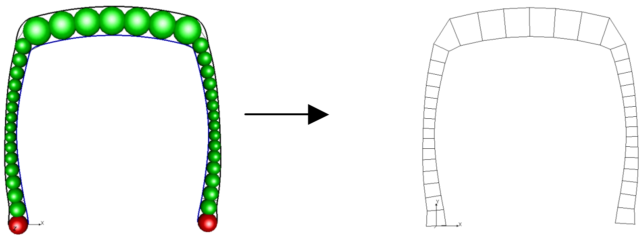
Quadrilateral elements in the mesh shown in Figure 11 (a) have an aspect ratio of almost 1:1. Note that the diameters of the bubbles are identical to the width of the domain.

Quadrilateral elements in the mesh shown in Figure 11 (b) have almost uniform length. The element length distribution function $d(\mathbf{x})$ is a constant equal to the smallest width of the cross-section.

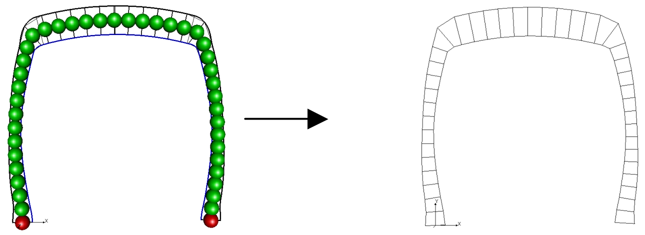
The same length distribution function is used in Figure 11 (c), but the width threshold is chosen so that the mesh becomes partially double-layered.

8.2 Curvature-Based Length Distribution

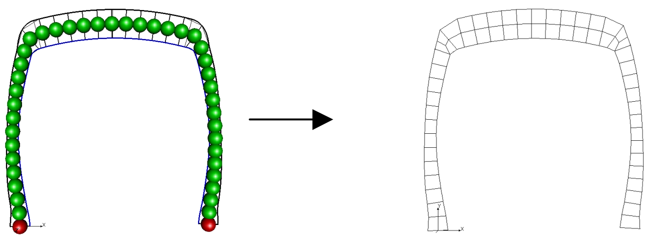
Figure 12 shows a single-layer mesh with a curvature-based element length distribution function. The element length distribution function is specified so that the length of an element becomes smaller in a region where the curvature of a boundary curve is high. Figure 12 also shows the bubble packing on the chordal axis of the letter 'm.'



(a) A single-layer mesh with uniform aspect ratio



(b) A single-layer mesh with uniform element length



(c) A partially double-layer mesh with uniform element length and width

Figure 11 Quadrilateral mesh of a tire cross-section

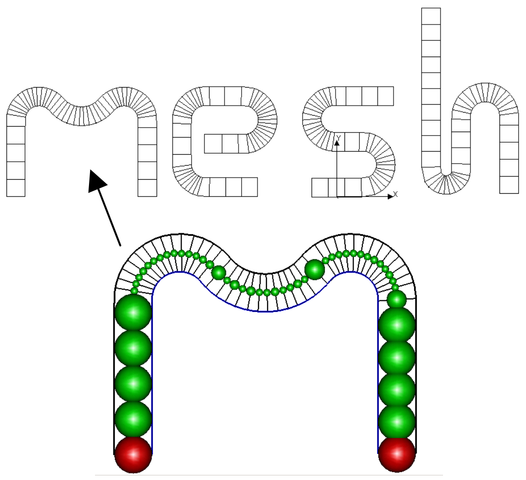


Figure 12 Single-layer mesh with a curvature-based element length distribution function

8.3 Meshing a Geometric Domain with Branches

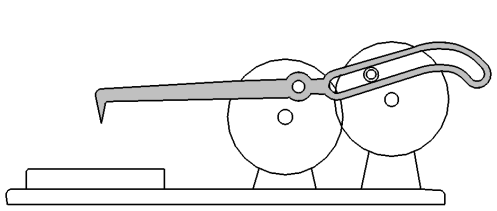
When the chordal axis of an object has branches, the object must be decomposed into two types of sub-domains: sub-domains to be meshed by the proposed Quad-Layer method, and other sub-domains to be meshed by an existing method such as advancing front or triangle pairing. Each sub-domain is meshed independently and then assembled to form a complete quadrilateral mesh.

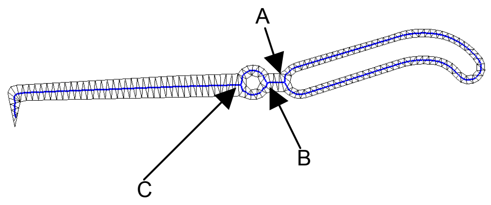
While the general domain decomposition problem is not trivial, some two-dimensional domains can be decomposed easily using chordal axis transformation. For example, the filmstrip hook [18] shown in Figure 13 (a) can be decomposed into the sub-domains shown in Figure 13 (c) based on the branch points of the chordal axis and a width threshold.

The chordal axis of the hook has three branch points, A, B and C, as shown in Figure 13 (b). With the three branch points, the domain is subdivided into eight sub-domains: the righthand side of A, the triangle that includes A , the domain between A and B, the triangle that includes B, two domains between B and C, the triangle that includes C, and the lefthand side of C. The sub-domain between A and B is not a narrow domain because it has an aspect ratio of almost 1:1. An appropriate width threshold is then applied in order to separate a sharp feature such as the sharp corner at the left end of the hook. In this case, a volume threshold of 0.62 is applied, and the sub-domain on the lefthand side of C is split into two sub-domains: the sharp corner on the left and the rest of the arm on the right.

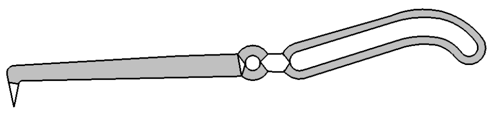
After the domain is decomposed, narrow sub-domains, shaded in Figure 13 (c), are meshed by the Quad-Layer method. Other sub-domains are meshed by the method presented by Itoh et al. [6]. Meshes of sub-domains are then assembled into the mesh shown in Figure 13 (d). Finally, Laplacian smoothing is applied to the mesh in order to further improve the quality of the mesh as shown in Figure 13 (e).

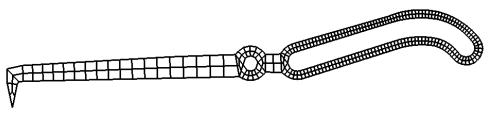
Another example of a geometric domain with branches is the spiral shaped wheel [18] shown in Figure 14. The chordal axis of the wheel has only one branch point as shown in Figure 14 (b). Although three chordal axes are joined at the intersection, there are only two branches because two of the three axes merge together.


(a) A filmstrip hook



(b) The chordal axis of the hook


(c) Decomposition of the hook


(d) A quad-mesh of the hook

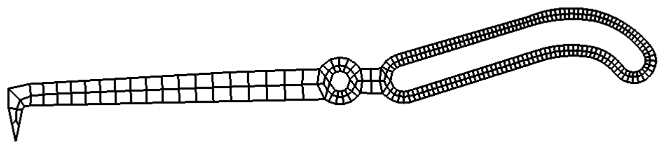
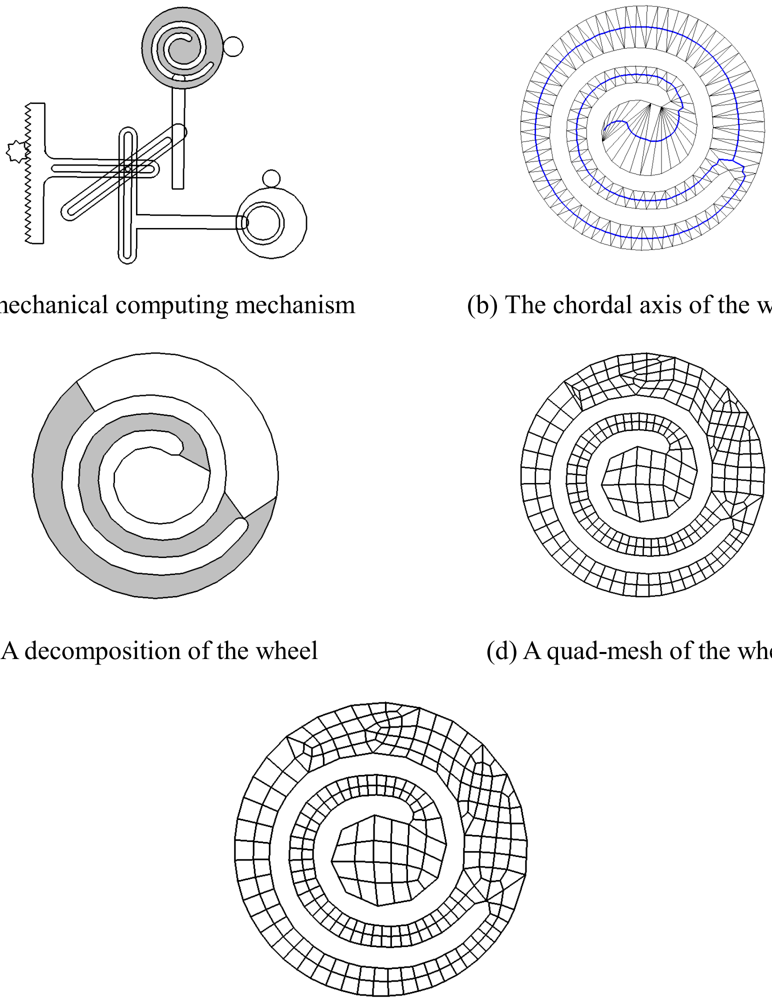

(e) Quad-mesh of the hook afterapplying Laplacian Smoothing

Figure 13 A film-strip hook

This domain is decomposed into three sub-domains: the triangle, which includes the intersection, and two sub-domains corresponding to the two branches of the chordal axis. This decomposition, however, is not appropriate because it connects a circular feature at the center with a narrow domain. In order to separate the circular feature from the narrow domain, a width threshold of 1.25 is applied. This threshold also subdivides the sub-domain that corresponds to the branch departing from the branch point and returning to the bottom. As a result, the sub-domain is further subdivided into two pieces, one of which is above the triangle, and the other is below the triangle that includes the branch point. The former sub-domain is not considered a narrow domain because it is wider than the threshold. This sub-domain is thus merged with the triangle and meshed with a conventional method. Final decomposition is shown in Figure 14 (c).

The shaded sub-domains in Figure 14 (c) are meshed by Quad-Layer, and the other sub-domains are meshed by Itoh et al.'s method. Assembling all the sub-domain meshes yields the mesh shown in Figure 14 (d). The mesh is smoothed by Laplacian smoothing as shown in Figure 14 (e).


(a) A mechanical computing mechanism

(b) The chordal axis of the wheel

(c) A decomposition of the wheel

(d) A quad-mesh of the wheel

(e) The quad-mesh of the wheel after applying Laplacian Smoothing

Figure 14 A mechanical computing mechanism

9 Application to Large Deformation FEA

Several finite element analyses of a large deformation of a rubber part were performed. The goal of the experiments is to show that the quality of a mesh has a significant effect on the accuracy of such a non-linear finite element analysis.

Figure 15 shows the results of finite element analyses in which a rubber part is squeezed and highly deformed between two rigid plates. The analyses are performed by using a commercial finite element analysis package, MARC, with four-node quadrilateral elements. Three meshes used in the experiments are:

- a four-layer mesh **A** created by Quad-Layer as shown in Figure 15 (a)
- a single-layer mesh **B** created by Quad-Layer as shown in Figure 15 (b)
- a single-layer mesh **C** created by a conventional method with uniformly distributed nodes as shown in Figure 15 (c)

The four-layer mesh **A** has 256 elements, many more elements than meshes **B** and **C**, and it is reasonable to assume that the solution obtained with this mesh is closer to the exact solution than the solutions obtained with meshes **B** and **C**. Single-layer mesh **B** has 15 high quality elements; none of the elements is inverted anywhere in the domain during or after the deformation. In comparison, single-layer mesh **C,** created by a conventional method, has 15 distorted elements. Note that the elements are further distorted during the deformation, and eventually two elements, the 6th and 7th elements from the bottom, become inverted during the simulation.

Mesh **B** created by Quad-Layer gives more accurate results in estimating the maximum Cauchy stress. The maximum Cauchy stress computed with the four-layer mesh is 3.737 N/mm^2. Assuming that this value is reasonably close to the exact solution, we use this value as a reference Cauchy stress. The maximum Cauchy stress computed using single-layer mesh **B** is 3.828 N/mm^2 with only 2.44% error. On the other hand, the maximum Cauchy stress computed with single-layer mesh **C** is 4.31 N/mm^2 with 15.3% error. These results indicate that the quality of a mesh has a significant effect on the accuracy of a non-linear, large deformation finite element analysis.

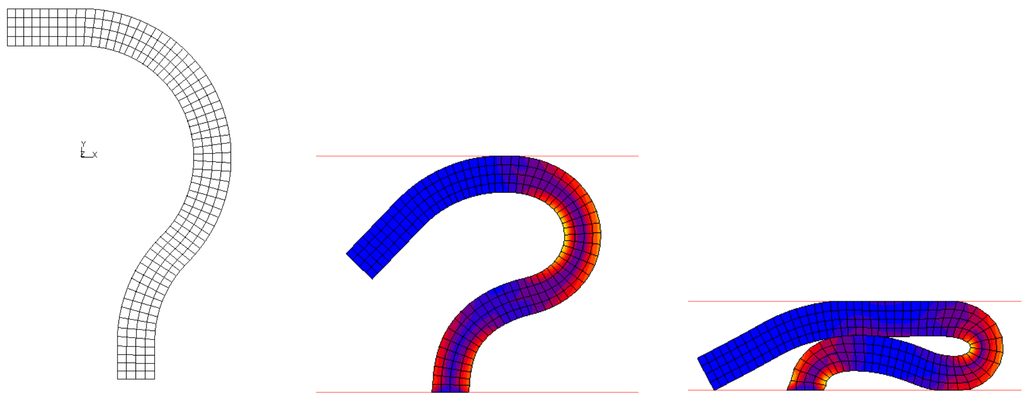
10 Conclusion

A new method for the automatic generation of a layered quadrilateral mesh of a thin, or narrow, two-dimensional domain is presented. A mesh created by the proposed Quad-Layer method is suitable for large deformation finite element analysis due to the high quality of its element shapes.

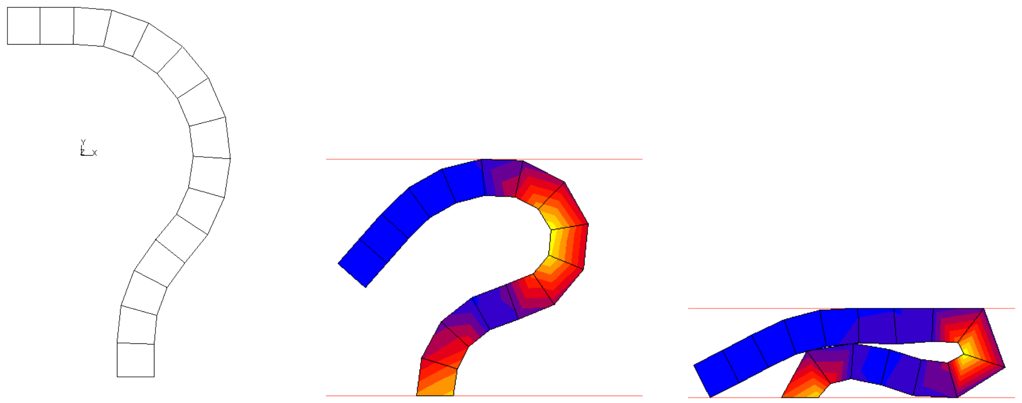
Our experiments indicate that the method creates a high quality mesh for various geometric domains. There is, however, no provable bound of the mesh quality—the quality of the mesh depends on the curvature distribution of the domain boundaries as well as the element size distribution function. It is a future research goal to identify a set of rules for defining an appropriate element size distribution function for a given curvature distribution.

Acknowledgments

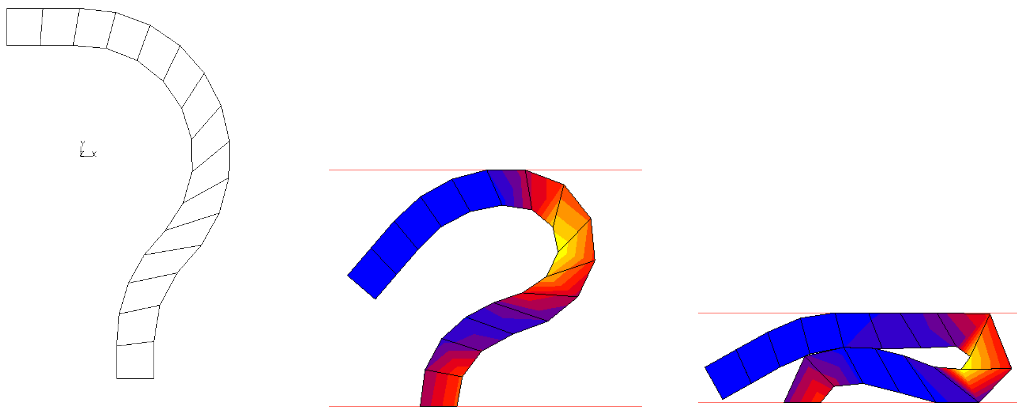
This material is based in part on work supported under a NSF CAREER Award (No. 9985288).



(a) Four-layer mesh **A** created by our algorithm. The max Cauchy stress is 3.737 N/mm^2.



(b) Single-layer mesh **B** created by our algorithm . The max Cauchy stress is 3.828 N/mm^2 (2.44% error compared with mesh **A**))



(c) Single-layer mesh **C** created by a conventional meshing algorithm. The max Cauchy stress is 4.310 N/mm^2 (15.3% error compared with mesh **A**)

Figure 15 Large deformation finite element analysis of a rubber part

References

[1] T. D. Blacker and M. B. Stephenson, "Paving : A New Approach to Automated Quadrilateral Mesh Generation," *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 811-847, 1991.

[2] R. J. Cass and S. E. Benzley, "Generalized 3-D Paving : An Automated Quadrilateral Surface Mesh Generation Algorithm," *International Journal for Numerical Methods in Engineering*, vol. 39, pp. 1475-1489, 1996.

[3] S. J. Owen, M. L. Staten, S. A. Canann, and S. Saigal, "Advancing Front Quadrilateral Meshing Using Triangle Transformations," presented at 7th International Meshing Roundtable, pp. 409-428, 1998.

[4] D. S. Thompson and B. K. Soni, "Generation of Quad- and Hex-dominant, Semistructured Meshes Using an Advancing Layer Scheme," presented at 8th International Meshing Roundtable, pp. 171-178, 1999.

[5] K. Shimada, J.-H. Liao, and T. Itoh, "Quadrilateral Meshing with Directionality Control through the Packing of Square Cells," presented at 7th International Meshing Roundtable, pp. 61-75, 1998.

[6] T. Itoh, K. Shimada, K. Inoue, A. Yamada, and T. Furuhata, "Automated Conversion of 2D Triangular Mesh into Quadrilateral Mesh with Directionality Control," presented at 7th International Meshing Roundtable, pp. 77-86, 1998.

[7] D. R. White, "Redesign of the Paving Algorithm : Robustness Enhancements through Element by Element Meshing," presented at 6th International Meshing Roundtable, pp. 323-335, 1997.

[8] N. Viswanath, K. Shimada, and T. Itoh, "Quadrilateral Meshing with Anisotropy and Directionality Control via Close Packing of Rectangular cells," presented at 9th International Meshing Roundtable, pp. , 2000.

[9] T. K. H. Tam and C. G. Armstrong, "2D Finite element mesh generation by medial axis subdivision," *Advances in Engineering Software*, vol. 13, pp. 313-324, 1991.

[10] H. N. Gürsoy and N. M. Patrikalakis, "An Automatic Coarse and Fine Surface Mesh Generation Scheme Based on Medial Axis Transform: Part I Algorithm, Part II Implementation," *Engineering with Computers*, vol. 8, pp. 121-137,179-196, 1992.

[11] W. R. Quadros, K. Ramsawami, F. B. Prinz, and B.Gurumoorthy, "LayTracks: A New Approach To Automated Quadrilateral Mesh Generation using MAT," presented at 9th International Meshing Roundtable, pp. 239-250, 2000.

[12] L. Prasad, "Morphological Analysis of Shapes," in *http://cnls.lanl.gov/Highlights/1997-07/*: Los Alamos National Laboratory, 1997.

[13] J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator," presented at First Workshop on Applied Computational Geometry, pp. 124-133, 1996.

[14] P. L. George, "TET MESHING: Construction, Optimization and Adaptation," presented at 8th International Meshing Roundtable, pp. 133-141, 1999.

[15] K. Shimada, A. Yamada, and T. Itoh, "Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles," presented at 6th International Meshing Roundtable, pp. 375-390, 1997.

[16] 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.

[17] S. Yamakawa and K. Shimada, "High Quality Anisotropic Tetrahedral Mesh Generation via Ellipsoidal Bubble Packing," presented at 9th International Meshing Roundtable, pp. 263-273, 2000.

[18] N. P. Chironis and N. Sclater, *Mechanisms and Mechanical Devices Sourcebook*, Second ed: McGraw Hill, 1996.

[19] S. Yamakawa and K. Shimada, "*Quad-Layer: Layered Quadrilateral Meshing of Narrow Two-Dimensional Domain by Bubble Packing and Chordal Axis Transformation*," accepted for presentation at ASME/DETC/DAC, 2001.