# Freeform surface flattening based on fitting a woven mesh model

**Charlie C. L. Wang**

Department of Automation and Computer-Aided Engineering, Chinese University of Hong Kong,
Shatin, N.T., Hong Kong, P. R. China
E-mail: cwang@acae.cuhk.edu.hk

**Kai Tang*** and **Benjamin M. L. Yeung**

Department of Mechanical Engineering, Hong Kong University of Science and Technology,
Clear Water Bay, N.T., Hong Kong, P. R. China
E-mail: mektang@ust.hk

## Abstract

This paper presents a robust and efficient surface flattening approach based on fitting a woven-like mesh model on a 3D freeform surface. The fitting algorithm is based on *tendon node mapping* (TNM) and *diagonal node mapping* (DNM), where TNM determines the position of a new node on the surface along the warp or weft direction and DNM locates a node along the diagonal direction. During the 3D fitting process, strain energy of the woven model is released by a diffusion process that minimizes the deformation between the resultant 2D pattern and the given surface. Nodes mapping and movement in the proposed approach are based on the discrete geodesic curve generation algorithm, so no parametric surface or pre-parameterization is required. After fitting the woven model onto the given surface, a continuous planar coordinate mapping is established between the 3D surface and its counterpart in the plane, based on the idea of geodesic interpolation of the mappings of the nodes in the woven model. The proposed approach accommodates surfaces with darts, which are commonly utilized in clothing industry to reduce the stretch of surface forming and flattening. Both isotropic and anisotropic materials are supported.

**Keywords:** surface flattening, woven material, strain energy, 3D fitting, and freeform surface.

* Corresponding Author: Dr. Kai Tang, Dept. of Mechanical Engineering, Hong Kong University of Science and Technology, Hong Kong;
E-mail: **mektang@ust.hk**

## 1. Introduction

Surface flattening is an important process in many applications (e.g., aircraft industry, ship industry, shoe industry, apparel industry, etc.). In the traditional process of footwear industry, the profile of the shoe upper layer is first estimated and then cut out; after sewing together the pieces of the layer, a foot shape mould is inserted to deform the leather to a desired shape [1]. In the aircraft industry, structures reinforced by woven fabrics are commonly used [2]. Similar to the footwear case, profiles of the woven fabrics are estimated and cut out, and then they are laid onto a certain 3D shape. In both cases, the profile of the material is still conjectured in practice by human based on trial-and-error and this estimation is quite time consuming and inaccurate. In the Computer-Aided Design (CAD) of products, people expect to obtain an accurate profile. Actually, they want to obtain the profile in a reverse way: firstly designing the 3D surface of the product on a CAD system, and then determine the corresponding 2D profile of the surface. This is exactly the following surface flattening problem:

**Problem Definition** Given a 3D freeform surface and the material properties, find its counterpart pattern in the plane and a mapping relationship between the two so that, when the 2D pattern is folded into the 3D surface, the amount of distortion – wrinkles and stretches – is minimized.

In this paper, we present a surface flattening technique based on fitting a woven-like mesh (*woven mesh*) model onto a 3D surface $M$. Two mapping methods: *tendon node mapping* (TNM) and *diagonal node mapping* (DNM) are proposed to initially locate the nodes of a woven mesh on the given surface. In the tendon node mapping, two mutually perpendicular geodesic curves are generated on $M$ which are called *tendon*s since they will not be moved in the ensuing energy releasing process and they are mapped into two perpendicular straight lines on the planar woven before the fitting. The tendon nodes are located on the tendon curves with equal distance. The diagonal node mapping method is then incorporated to position new nodes based on the other three located nodes belonging to the same quad in the woven mesh. Thus, by a propagation procedure, the nodes can be fitted on $M$ one by one. During the fitting of nodes, strain energies at the fitted nodes are released by a diffusion process. The strain energy is defined based on the geodesic distance of adjacent nodes and their Euclidean distance on the surface. The difference between the original 2D woven mesh and the given surface is minimized, so the deformation between the 2D profile and the 3D freeform surface is minimized. Both the node mapping and movement in our approach are based on the discrete geodesic curve generation algorithm [4]; therefore, different from other existing methods [5-9], no parametric surface or pre-parameterization is required by us. After fitting a woven mesh model, a planar coordinate mapping is developed to compute the 2D coordinate of every point on $M$. The proposed fitting technique accommodates surfaces with darts which are commonly adopted in practice to reduce the distortion of surface forming and flattening. Also, for the strain energy minimization, not only isotropic but also anisotropic materials can be simulated.

The freeform 3D surface considered in this paper is represented as a two-manifold polygonal mesh with a boundary, which is topologically equivalent to a disk. The mesh is a complex of vertices and the connectivity between the vertices – here we adopt the data structure in [3] to store the mesh. Using this data structure, we can easily obtain the adjacent relationship of vertex-vertex, vertex-edge, vertex-face, and edge-face.

The paper is organized as follows. We will first review some related work in surface flattening. The woven mesh model is then introduced. The detail fitting methodology is presented in section 4, in the sequence of tendon node mapping, diagonal node mapping, boundary propagation, and strain energy minimization. Section 5

describes the planar coordinate mapping scheme which establishes the continuous mapping relationship between every point on a given surface and its flattened 2D counterpart. A number of experimental examples are then presented to illustrate the proposed flattening algorithm, and comparisons are made with two other known surface flattening algorithms (one is pure geometry-oriented and another is energy-based). Finally in Section 7 we summarize the paper.

## 2. Related Work

Due to its importance, in both theory and practice, research in surface flattening has been active for a number of years, and not limited to only design and manufacturing. In the following we give a short summary on the various related developments over the past few decades.

**Parameterization**

The flattening of a triangular 3D mesh, which provides a bijective mapping between the mesh and a triangulation of a planar polygon, plays an important role in parameterization and texture mapping. An excellent survey of recent advances in mesh parameterization is given in [10], see also the references therein. Floater [11] investigated a graph-theory based parameterization for tessellated surfaces for the purpose of smooth surface fitting; his parameterization (actually a planar triangulation) is the solution of linear systems based on convex combination. In [12], Hormann and Greiner used Floater's algorithm as a starting point for a highly non-linear local optimization algorithm which computes the positions for both interior and boundary nodes based on local shape preservation criteria. The method is promising, but it is not clear if the procedure is guaranteed to converge to a valid solution. A quasi-conformal parameterization method based on a least-squares approximation of the Cauchy-Riemann equations is introduced in [13], where the defined objective function minimizes angle deformation. Desbrun et al. [14] developed an efficient parameterization algorithm minimizing the distortion of different intrinsic measures of the original mesh. However, in both [13] and [14], the linear stretch is not considered. Sheffer and de Sturler [15, 16] presented a texture mapping algorithm that causes small mapping distortion. Their algorithm consists of two steps: 1) using the Angle Based Flattening (ABF) parameterization method to provide a continuous (no foldovers) mapping, which concentrates on minimizing the angular distortion of the mapping and hence unavoidablely often leads to relatively large linear distortion; 2) to reduce the linear distortion, an inverse mapping from the plane to the result of ABF is computed to improve the parameterization – the improved result has low length distortion. In [17], a texture stretch metric is introduced to minimize the linear distortion via non-linear optimization. Since non-linear numerical optimization is conducted in [15-17], these approaches are time consuming. Most recently, in [18], a fast and simple method for generating a low-stretch mesh parameterization is developed. It starts from any other parameterization (e.g., the Floater shape preserving parameterization [11], or the intrinsic parameterization [14]) and then improves the parameterization gradually by a diffusion process using the stretch metric of [17]. It can significantly improve the stretch in a mesh parameterization. However, since the boundary vertices are not moved, the 2D boundary profile depends on the initial parameterization. When the natural boundaries are required (as mentioned earlier in our problem definition), they use the intrinsic parameterization [14]. Since in [14] the stretch is not minimized, the resultant 2D profiles are seldom satisfied in either its length or the enclosed area.

**Strain-energy minimization**

McCartney et al. [19] flatten a triangulated surface by minimizing the strain energy in the 2D pattern. The 3D surface is first triangulated using Delaunay triangulation. Then the triangles are transformed onto a plane. However, there are some flattened triangles that cannot preserve their length relationship with respect to their counter-parts on the surface. The length differences are measured as strain energy, where the zero strain energy indicates that the flattened triangles preserve their length relationships with the original triangles on the surface, i.e. no deformation occurs. Thus, iterative method is applied to minimize this strain energy in the 2D pattern. The endpoints of the triangles are moved in orthogonal directions by trial to obtain smaller energy in each iteration. Wang et al. [20] improve McCartney's algorithm by using a spring-mass system. The system guides the endpoints to approach better positions by the force of springs and the computational speed of the minimization is improved. The accuracy of the flattening can also be controlled by using the spring constant.

There are also some other energy-minimization based flattening algorithms (cf. [21-24]). All these algorithms though share a common strategy: the energy minimization scheme is applied on the 2D pattern. In other words, they assume the original 3D surface has zero energy, i.e., without wrinkles or stretches, while the 2D pattern is sought that minimizes the deformation energy. On the contrary, many physical processes are just opposite. For example, when a motorcycle helmet is made, a certain piece is first cut out of a 2D sheet, which is totally relaxed and hence has zero energy. This piece then is folded onto the hard model of the helmet shape to form one layer of the helmet; the energy, in the form of wrinkles and stretches, thus is generated in this folded 3D layer. Naturally, it should be asked if a "forward" energy minimization can generate better flattening result, as it corresponds closer to the physical process. Thus, we choose this manner to flatten a given freeform surface. Also, by the limitation of the irregular mesh utilized in the above algorithms [19-24], the anisotropic material is hardly simulated. Our approach utilizes a woven-like regular quadrilateral mesh model, which greatly facilitates the simulation of anisotropic material based surface flattening and folding.

**Woven fabrics related models**

Some surface flattening solutions specifically applied to woven fabrics models [5-7]. Woven fabrics consist of a series of vertical threads (warp) that cross with a series of horizontal threads (weft). The strength of the threads is usually strong and they resist deforming under force, but the shear deformation at the crossing between a warp thread and a weft thread can occur. In [5], three assumptions are made to model a ply of woven fabrics: 1) the warp and weft threads are inextensible; 2) a thread segment between adjacent crossings is straight on the surface; 3) no slippage occurs at a crossing when the ply is deformed. The algorithm of Aono et al. [5] is a geometrical approach for flattening a woven ply. A base line is chosen on the 3D surface and equidistance points (crossings) are mapped on it. Equidistance nodes are then mapped throughout the whole surface under predefined sweeping direction. Shear deformation is expected and cuts (called darts) can be inserted to reduce the shear [6, 7]. The problem of this method is that the final 2D pattern is not unique. The shape of the profile crucially depends on the position and the orientation of the base line and the sweeping direction. There are cases that, when the base line and/or the sweeping direction are not properly chosen, the algorithm diverges, thus failing to generate the 2D profile. This problem is solved in our approach via a strain-energy releasing process. Another problem occurs in the mapping calculation of an auxiliary mesh point and a regular mesh point using spheres. When the surface is much concaved, multiple intersection points incur between spheres and the surface,

so it is difficult to determine a correct one as the mapping point. Since the mapping method in our woven fitting is based on the geodesic path, this problem is solved naturally. Also, our new technique overcomes the limitation of straight darts and parametric surface in their approaches.

**Dart insertion**

Some approaches found in literature considered the issue of where to insert cuts in the 2D pattern. Parida and Mudur (1993) [25] presented an algorithm to obtain planar development (within acceptable tolerances) of complex surfaces with cuts and overlaps only in specified orientations. Their algorithm first obtains an approximate planar shape by flattening the triangles on the surface, cracks are generated while triangles are flattened one by one; and then, they reorient cracks and overlap parts in the developed plane to satisfy orientation constraints. Their algorithm might generate many cracks and calculation errors. The approach of Wang et al. [20] generates the cutting line from the stretch energy distribution map; however, the length of cutting paths is not considered in their paper. In [26], Sheffer tried to find the shortest cutting path that passes through the nodes with high Gaussian curvature to reduce the parameterization distortion of the triangulated surface. Unfortunately, this method is not able to find protrusions with widely distributed curvatures (e.g., looped cylindrical surfaces). To enhance Sheffer's approach, Wang et al. [27] developed a technique that computes the shortest path from a node to the surface boundary; and the cutting paths on the surface with widely distributed curvatures are generated while preventing flipped triangles in the developed 2D pattern. Recently, Katz and Tal [28] proposed a hierarchical mesh decomposition algorithm that decomposes a given mesh into meaningful components referring to segmentation at regions of deep concavities. However, this cut insertion technique cannot be directly applied to reduce stretches in flattening. In our woven fitting model, a natural solution of darts process is given.

**Geodesic curves**

As already mentioned at the beginning of this paper, the discrete geodesic curve generation algorithm is crucially utilized in our woven fitting approach. Theoretically, a curve on a surface is called a geodesic if, at every point on the curve, its osculating plane contains the normal of the surface at that point. In [4], an efficient algorithm is given for the computation of a discrete geodesic on a tessellated surface, where the tessellation normal is taken into account to determine the geodesic path $G$ by a given point $p_0$ and a direction vector $t_0$. A local incremental algorithm is adopted. At every point $p_i$ on $G$, the geodesic path is locally coincident to the intersection curve of the plane determined by $p_i$, $t_i$ and $n_i$, where $n_i$ is the tessellation normal at $p_i$ and $t_i$ is the tangent of the geodesic path at this point. The pseudo-codes of a detailed discrete geodesic algorithm following [4] are given in Appendix A. Since the geodesic path is computed locally, this algorithm is efficient.

## 3. Woven Mesh Model

Our surface flattening algorithm is based on fitting a woven mesh model onto a given freeform surface $M$ represented in the form of a polygonal mesh. After the fitting, all nodes of the woven mesh lie on $M$ exactly, and the strain-energy in the woven mesh is minimized. In this section, the necessary definitions of the woven mesh model are given.

**Definition of the woven mesh model**

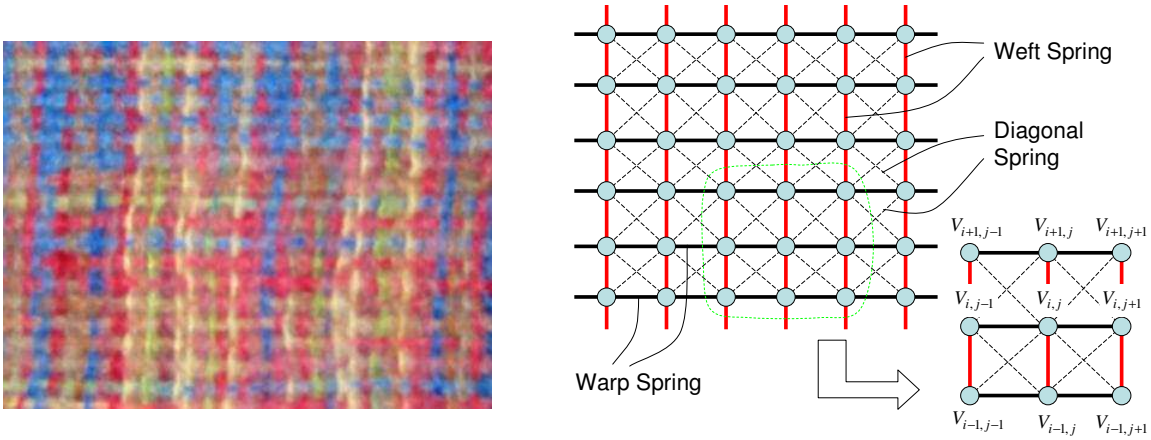
The surface to be flattened can be regarded as a ply of woven fabric consisting of horizontal and vertical threads interwoven in a specific fashion as seen in Fig. 1a. The ply generally has strong tensile-strain resistance in the thread directions and a weaker shear-strain resistance [29]. As already alluded earlier, three assumptions are made with this model.

**Assumption 1** The weft threads and the warp threads are not extendable.

**Assumption 2** No slippage occurs at the crossing of a weft and a warp thread.

**Assumption 3** A thread between two adjacent crossing is mapped to a Geodesic curve segment on the 3D surface.

*Assumption* 1 and 2 come from the mechanical behavior of materials [29], and based on them and the properties of a developable surface [30], *Assumption* 3 is proposed. In our approach, the woven fabric is modeled by a spring mesh $\Gamma$. An example spring mesh model is shown in Fig. 1b.




**(a)** **(b)**

**Fig. 1 Woven mesh model: (a) a real woven fabric; (b) our spring mesh simulation of a woven fabric.**

There are three components in this model, weft (vertical) spring, warp (horizontal) spring, and diagonal spring. For real woven fabrics, there are no diagonal threads. The reason that they are added is to simulate the shear deformation resistance. Since the woven fabrics present strong tensile-strain resistance in the thread directions but weaker shear-strain resistance, the coefficients of diagonal springs usually are much smaller than that of weft and warp springs. Each of the three types of the spring has its own initial length at which the spring attains the zero energy. A *node* is an intersection between springs whose position determines the deformation of the springs connected to that node. Each node is indexed by $V_{i,j}$, where $i, j$ are integers representing the indexes of row and column respectively. In our model, a node can be categorized into two groups, *internal nodes* and *boundary nodes*.

**Definition 1** For a mesh node $V_{i,j}$, the valence function $\lambda(V_{i,j})$ is the number of springs connecting to it. If $\lambda(V_{i,j}) = 8$, $V_{i,j}$ is an *internal node*; otherwise, $V_{i,j}$ is called a *boundary node*. Also, a principle valence function $\psi(V_{i,j})$ is defined to represent the number of non-diagonal springs linking to $V_{i,j}$.

Usually, a mesh node's valence depends on both the number of its neighboring nodes and whether the springs linking $V_{i,j}$ and its neighbors passing cross the boundary curve. This will be classified by the following definitions.

**Definition 2** For a mesh node $V_{i,j}$, if there is another mesh node $V_{i+a,j+b}$ satisfying: 1) $a,b \in \{-1,0,1\}$ and 2) $|a|+|b|>0$, $V_{i+a,j+b}$ is a *neighboring node* of $V_{i,j}$.

**Definition 3** For a mesh node $V_{i,j}$ and its neighbor $V_{i+a,j+b}$, if the geodesic curve linking them crosses the boundary (including darts) of the given surface $\mathrm{M}$, the spring linking them is said to be *invalid*; otherwise, the spring linking them is a *valid spring*.

For a woven mesh in 2D, all the weft springs are aligned in one direction and all the warp springs are aligned in another direction. In our model, their directions are orthogonal to each other, although they don't have to be. If the initial length of the weft spring and the warp spring are $r_{weft}$, and $r_{warp}$ respectively, the initial length of the diagonal spring $r_{diag}$ is given by

$$r_{diag} = \sqrt{r_{weft}^2 + r_{warp}^2}\ . \tag{1}$$

Based on a user specified center node $V_{i_C,j_C}$ in the woven mesh $\Gamma$, the nodes on $\Gamma$ are classified into two types: tendon nodes and region nodes.

**Definition 4** $\forall V_{i,j} \in \Gamma$, if $i = i_c$ or $j = j_c$, $V_{i,j}$ is defined as a *tendon node*; otherwise, $V_{i,j}$ is called a *region node*.

**Definition 5** For a region node $V_{i,j} \in \Gamma$, if $i > i_c$ and $j > j_c$, $V_{i,j}$ is called a type-I node; likewise, it is a type-II node if $i > i_c$ and $j < j_c$, a type-III node if $i < i_c$ and $j < j_c$, and a type-IV node if $i < i_c$ and $j > j_c$.

When the woven mesh $\Gamma$ is fitted onto the 3D freeform surface, every node, no matter tendon or region ones, is on the freeform surface precisely. However, the directions and lengths of different kinds of springs may not be preserved as compared to their 2D counterparts. This leads to the strain energy. One major objective of our flattening algorithm is to release this strain energy in the woven mesh model during the fitting process. The detailed description of strain energy is given as follows.

**Strain Energy**

The strain energy defined on a women mesh gives a clear measurement to the deformation of a planar woven mesh before and after surface fitting. Actually, it measures the length change of springs in the woven mesh model. The strain energy on the spring $V_{i,j}V_{i+a,j+b}$ ($a,b \in \{-1, 0, 1\}$ and $|a| + |b| > 0$) can be expressed as

$$J_{(i,j,a,b)} = \frac{1}{2}k\left(\left\|V_{i,j}V_{i+a,j+b}\right\| - l^0\right)^2, \tag{2}$$

where $k$ is the spring constant, and $l^0$ is the initial length of the spring at its zero-energy state. Different spring constants give different levels of resistance to the length change on the spring. How to set a proper spring constant ratio is similar in principle to the case of making a laminate, in which, layers of materials are stacked together. For an isotropic cloth model, the counterpart in the laminate system is a quasi-isotropic laminate [25], which can be produced by stacking the same kind of material with orientation at -45˚, 0˚, +45˚ and 90˚. Thus, the diagonal spring in an isotropic cloth model reflects the tensile deformation in the cloth model. In order to simulate the inextensibility of the weft and the warp threads in real woven fabrics, the spring constants for the weft springs and the warp springs are set to be $K$ times larger than that of the diagonal springs, where $K$ is an empirically determined integer, chosen 500 to 550 in our system. The relatively large spring constant serves as a penalty to inhibit the deformation of the weft and the warp springs, and the relatively small spring constant for the diagonal springs allows them to deform freely. Therefore the deformation of the spring mesh is dominated by the deformation of the diagonal springs, i.e. shear deformation, and this is close to the deformation pattern of real woven fabrics.

The entire energy of the woven mesh $\Gamma$ is define as the summation of energy of all the springs,

$$E = \sum_{V_{i,j}V_{i+a,j+b} \in \Gamma} J_{(i,j,a,b)} \ . \tag{3}$$

The energy measured is that required to deform the woven fabric when fitting it onto a surface. For a surface with zero Gaussian curvature everywhere, i.e. a developable surface, the energy of the model would be zero. For a non-developable surface, since it cannot be flattened onto a planar figure without deformation, the energy of the model would be large. Thus, the energy $E$ represents inversely the developability of a woven mesh model.

## 4. Fitting Methodology

A configuration of a woven mesh fitted on a freeform surface is called an *optimum* if its deformation is the minimum. Correspondingly, an optimal configuration for the woven mesh is one with its strain energy $E$ minimized. In order to solve this minimization problem, we conduct a diffusion process to move every node of $\Gamma$ on the given surface $\mathrm{M}$. The issue of darts insertion will also be investigated.

Our fitting process starts with a user specified center point and two perpendicular geodesic paths on the surface $\mathrm{M}$; the rest of the nodes in the woven mesh are then determined based on the already located nodes in a propagation manner. The thus generated fitting mesh on surface $\mathrm{M}$ incurs strain energy, which is then released by a diffusion process. Three criteria are defined for a good woven fitting: 1) the overall deformation of the weft, the warp and the diagonal springs should be minimized; 2) no overlapping occurs; and 3) the spring mesh covers the entire freeform surface $\mathrm{M}$. Every step in our fitting process is designed to meet these three criteria.

We start with the *tendon node mapping* (TNM).

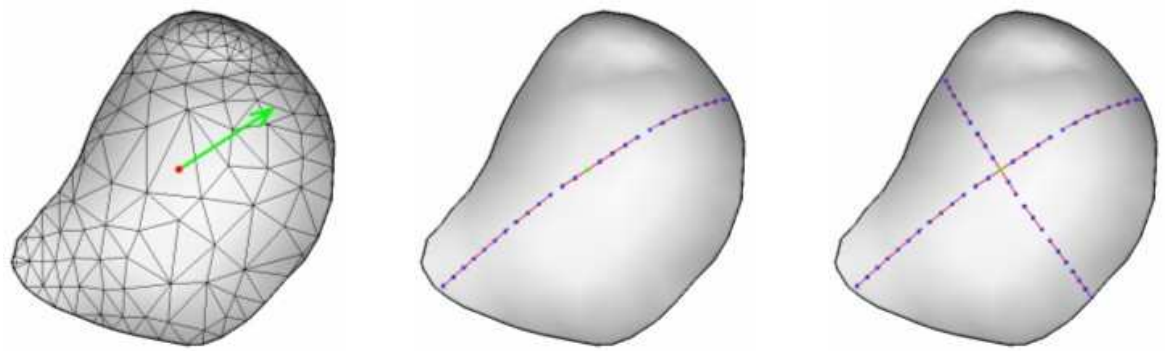
**Tendon node mapping (TNM)**

Two constrained paths – the tendons – are mapped by equidistance nodes in two different directions, representing a serials of warp springs and a serials of weft springs. This mapping causes no deformation in the

springs. Thus, the strain energy on the tendon springs is zero. All nodes on the tendons are fixed in the later energy releasing process, so the orientation of the fitted woven mesh is preserved. The nodes and springs on the two constrained paths act as support for the entire woven mesh model – that's why they are called tendons. In our implementation, after manually specifying a center point $p_c$ and a warp direction vector $t_{warp}$ on $\mathrm{M}$, we obtain the weft direction at $p_c$ by

$$t_{weft} = n_c \times t_{warp}, \tag{4}$$

where $n_c$ is the normal vector to surface $\mathrm{M}$ at $p_c$. With $t_{warp}$, $t_{weft}$, and $p_c$ available, by repeatedly calling ***Algorithm*** ComputeDiscreteGeodesicPath (see Appendix A), we can easily position all the nodes on the tendons. For example, when determining nodes on the $t_{warp}$ direction, first let $p_{act} = p_c$ be the current active node and set $V_{i_C,j_C} = p_c$ with $V_{i_C,j_C}$ being the center node in $\Gamma$; calling ***Algorithm*** ComputeDiscreteGeodesicPath(…) with ($V_{i_C,j_C}$, $t_{warp}$, $\mathrm{M}$, $r_{warp}$) as input will return a sequence of points $\varpi$, so the last point in $\varpi$ is assigned to $V_{i_C,j_C+1}$. Then, calling ***Algorithm*** ComputeDiscreteGeodesicPath(…) again with ($V_{i_C,j_C+1}$, $t_{warp}$, $\mathrm{M}$, $r_{warp}$) produces the mapping node on $\mathrm{M}$ for $V_{i_C,j_C+2}$, and so forth, until the boundary of $\mathrm{M}$ is reached. All tendon nodes in the $+t_{warp}$ direction are thus determined. The nodes in $-t_{warp}$, $+t_{weft}$ and $-t_{weft}$ directions can be obtained in the similar way. Fig. 2 shows an example of tendon node mapping on a freeform surface, where the red point in Fig. 2a is the point $p_c$, the green point in Fig. 2b, 2c is $V_{i_C,j_C}$, and the blue points represent the tendon nodes.



**(a)** **(b)** **(c)**

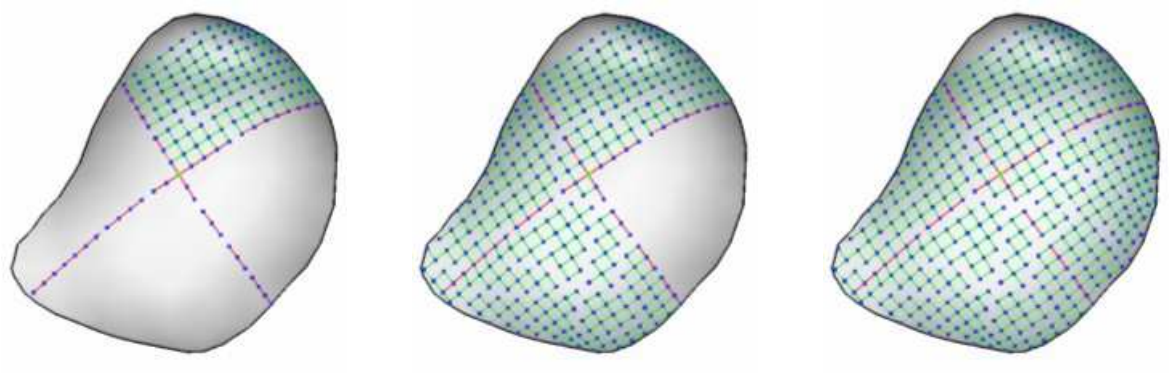
**Fig. 2 Tendon node mapping: (a) center point and $t_{warp}$ specified; (b) after mapping nodes on the warp tendon; (c) after mapping nodes on the weft tendon.**

**Diagonal node mapping (DNM)**

After all the tendon nodes have been positioned, the surface is divided into four quadrants. Nodes are mapped on the surface quadrant by quadrant independently, row by row and column by column. Fig. 3 shows an example of DNM in four quadrants. The mapping method is called *diagonal node mapping* since the initial

position of each node is searched in the diagonal direction between the two tendon directions. Details of this mapping are described as follows.

For a type-I node $V_{i,j}$ (see Definition 5), if $V_{i-1,j-1}$, $V_{i-1,j}$ and $V_{i,j-1}$ all have been positioned, we determine $V_{i,j}$ on the surface M by the following four steps: 1) determine the two unit vectors $t_1 = V_{i-1,j}V_{i-1,j-1} \big/ \left\| V_{i-1,j}V_{i-1,j-1} \right\|$ and $t_2 = V_{i,j-1}V_{i-1,j-1} \big/ \left\| V_{i,j-1}V_{i-1,j-1} \right\|$; 2) set the diagonal direction as $t_{diag} = \frac{1}{2}(t_1 + t_2)$; 3) starting at $V_{i-1,j-1}$, search the point on the geodesic path along the $t_{diag}$ direction with distance $r_{diag}$, by calling ***Algorithm*** ComputeDiscreteGeodesicPath($V_{i-1,j-1}$, $t_{diag}$, M, $r_{diag}$), and assign that point to $V_{i,j}$; and 4) locally adjust the position of $V_{i,j}$, if necessary. For a type-II node $V_{i,j}$, we can obtain its position by searching along the diagonal geodesic path formed by $V_{i-1,j+1}$, $V_{i-1,j}$ and $V_{i,j+1}$. A type-III or a type-IV node can be positioned similarly. The searching distance $r_{diag}$ is determined by Eq. (1). After every new node $V_{i,j}$ has been determined, the validity of springs linking $V_{i,j}$ is detected following Definition 3 in section 3.



**(a)** **(b)** **(c)**

**Fig. 3 Diagonal node mapping: (a) type-I nodes have been mapped; (b) after all type-I, -II, and -III nodes are mapped; (c) the result of the final DNM.**

After the node $V_{i,j}$ is initially positioned by the DNM, if the warp and the weft springs at it are not perpendicular to each other, the strain energy would be generated. Since the position of $V_{i,j}$ will have great influence on the other nodes afterward, it is desirable that the strain energy at $V_{i,j}$ be reduced as early as possible. This is helped by a local energy releasing process. When $V_{i,j}$ has $n$ springs linked to it, we have

$$v_{move} = \sum_{k=1}^{n} \Delta l_k \frac{vs_k}{\left\| vs_k \right\|} \tag{5}$$

where $vs_k$ is the vector of the $k$th spring pointing from $V_{i,j}$ to the other end, and $\Delta l_k$ is the length change of the $k$th spring from its initial value. The strain-energy is reduced by iteratively changing the position of $V_{i,j}$. In each

iteration step, we move $V_{i,j}$ along the $v_{move}$ direction to a point with geodesic distance $\frac{1}{4}\|v_{move}\|$. In our tests, usually after 10 or so iterations, $V_{i,j}$ moves to a balanced position – with $v_{move} \approx 0$. The vector computed in Eq.(5) acts as a force in the energy releasing.

**Boundary propagation**

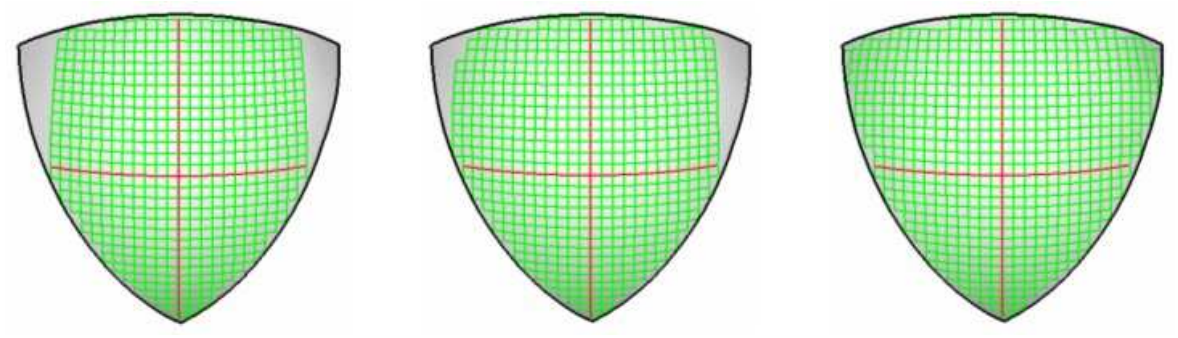
Since the DNM is based on the tendons and performed in a sequential manner, generally, the resulting woven model of TNM and DNM cannot fully fill the surface $\mathrm{M}$. Referring to Fig. 4a, the unmapped region is due to the absence of nodes on the previous row or column when performing the DNM. Here, we adopt a propagation method to fill the unmapped regions after TNM and DNM. The idea of boundary propagation is, if there is no node that can be diagonally mapped, first find a node with principle valence of three to locate a new point in its weft or warp direction. After that, go back to check whether DNM can position new nodes. The procedure is repeated until neither DNM nor the weft or warp extension can generate new nodes. To speed up the boundary propagation, a queue of boundary nodes is used to prioritize the candidates of the next new node. The pseudo-codes of the boundary propagation are given below, and an example of the woven model fitting before and after boundary propagation is shown in Fig. 4.

---

***Procedure*** BoundaryPropagation( $\mathrm{M}$ , $\Gamma$ )

1. Build the queue $Q$ of all the boundary nodes in $\Gamma$;
2. **loop** {
3. **while** (any node $V_b$ in $Q$ can have its diagonal neighbor mapped by DNM) {
4. Map the diagonal neighbor of $V_b$ onto $\mathrm{M}$ by DNM;
5. Adjust the nodes around $V_b$ and the newly mapped node in $Q$ – some added and some removed;
6. }
7. **if** ( (any node $V_b$ in $Q$ with principle valance three)
   **AND** (a new node can be added in weft or warp direction next to $V_b$) ) {
8. Add the new node in weft or warp direction next to $V_b$;
9. Adjust the population of Q by glancing the nodes around $V_b$ and the newly mapped node;
10. }
11. **else**
12. **return;**
13. }

---



**(a)** **(b)** **(c)**

**Fig. 4 Region filling by boundary propagation on an octant of a sphere: (a) mapped nodes after TNM and DNM; (b) an intermediate result of boundary propagation; (c) the final result of boundary propagation.**

**Energy minimization by diffusion**

Due to the choice of the direction and the mapping sequence of the two constrained base paths, the strain energy of the woven mesh $\Gamma$ after TNM, DNM and boundary propagation is usually not at the minimum state, even for a developable surface. The corresponding mapping represented by $\Gamma$ at this stage is not correct. Therefore, the strain energy must be minimized to obtain a better mapping. To minimize $E$ (Eq. (3)), one should let every node $V_i$ in $\Gamma$ satisfy

$$\frac{\partial E}{\partial V_i} = 0 . \tag{6}$$

The left part of the above equation can be rewritten as (referring to Eq. (2))

$$\frac{\partial E}{\partial V_i} = \sum\nolimits_j k_j \frac{V_j V_i}{\left\| V_i V_j \right\|} (\left\| V_i V_j \right\| - l_j^0) \tag{7}$$

where $V_j$ and $V_i$ form a spring, $k_j$ is the constant of this spring, and $l_j^0$ is the initial length (zero strain-energy) of the spring.

A diffusion-like process is applied to solve Eq. (7). The idea is borrowed from polygonal mesh fairing [31], where one common strategy for attenuating the noise in a mesh is to go through a diffusion process. Considering the current $\frac{\partial E}{\partial V_i}$ at every node as a force in terms of a spring-mass system, the new position of $V_i$ is obtained by

$$V_i^{new} = V_i - \lambda \frac{\partial E}{\partial V_i}, \tag{8}$$

where $\lambda$ is a damping factor and the movement of $V_i$ is along the geodesic path determined by $\frac{\partial E}{\partial V_i}$. Our idea of diffusing the strain energy by (7) and (8) is similar to the quasi-Newton type minimization algorithm: iteratively computing $\frac{\partial E}{\partial V_i}$ at every node $V_i \in \Gamma$ and moving $V_i$ along the geodesic path directed by $\frac{\partial E}{\partial V_i}$ with

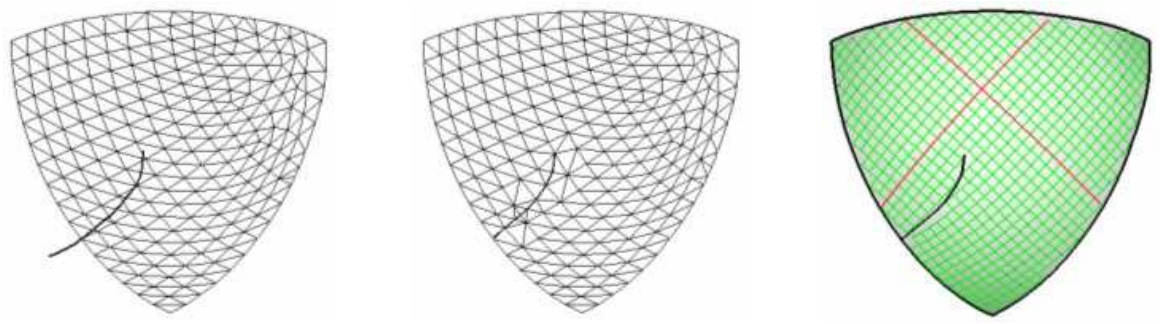
distance $\lambda\left\|\frac{\partial E}{\partial V_i}\right\|$. Choosing a smaller $\lambda$ will lead to more accurate result, but at the cost of slower convergence – we usually chose $\lambda = 0.1$ in our implementation. The iteration stops when the change of $E$ is less than a certain tolerance (1% in our implementation).

During the diffusion process some nodes may move outside the boundary of the freeform surface $\mathbf{M}$. Since the movement of a node is based on ***Algorithm*** ComputeDiscreteGeodesicPath(…), if the boundary is detected, the currently moved node should be removed from $\Gamma$, this will also cause the removal of all the linking springs to the node being removed. Conversely, some springs can shrink during the energy minimization, which will leave some portions of $\mathbf{M}$ uncovered by the mesh. Therefore, new nodes are inserted to fill those uncovered regions – this is carried out by ***Procedure*** BoundaryPropagation. After a new node is inserted, we follow the Definition 3 to check whether the springs linking to it are valid. This check is required when darts are incorporated.

**Insertion of darts**

The configuration, i.e. the $V_{i,j}$ neighboring node relationship, of the woven mesh model in the plane is the same as that on the freeform surface. However, the pattern, i.e. the area and the boundary, of the mesh in the plane may be quite different than that on the freeform surface, especially if the surface is non-developable. This is a direct result of our "forward" energy-minimization – the deformation is only allowed to occur in the woven mesh on the surface, but not in the plane which is considered to be totally relaxed.

For a non-developable surface, the deformation of the woven mesh model is seen to concentrate in highly double curved regions (elliptic or hyperbolic). In regions with severe spring deformation, i.e., with higher strain energy, darts can be inserted to release the energy, i.e. reduce the deformation. Different from [6, 8, 9], the darts are processed in a very natural manner in our approach. Users can define a dart by first specifying a freeform curve, straight or curved. After that, a polygonal curve is created on $\mathbf{M}$ by projecting the user specified curve onto $\mathbf{M}$ (recall that $\mathbf{M}$ is given as a polygonal mesh). Then, the constrained Delaunay triangulation algorithm [32] is applied to the projected 3D segments so they become edges of some triangle elements in the mesh representing $\mathbf{M}$. "Doubling" these newly created triangular edges into coincident edges will give a dart on $\mathbf{M}$. Comparing to the darts insertion methods in [6, 8, 9], our dart insertion method automatically associates the topological change of the surface with the insertion of a dart. Therefore, no modification to the just prescribed fitting algorithm is necessary on an $\mathbf{M}$ with darts. Fig. 5 illustrates how a dart is defined on an octant of a sphere.



**(a)** **(b)** **(c)**

**Fig. 5 An example of dart insertion on an octant of a sphere: (a) a user specified space curve; (b) the corresponding triangular edges after projection and Delaunay triangulation; (c) the fitted woven mesh.**

## 5. Surface to Plane Mapping

We have so far presented how to fit a woven mesh onto a freeform surface $\mathrm{M}$. The result of this fitting is a mapping between the nodes in $\Gamma$ and their counterparts in the plane. This mapping however is discrete – only a finite number of points (nodes) on $\mathrm{M}$ have their counterparts in the plane established. In this section, we describe how to interpolate this discrete mapping so that it becomes continuous – every point on $\mathrm{M}$ has its counterpart in the plane. In particular, since our surface $\mathrm{M}$ is given as a (triangular) polygonal mesh, every vertex $X_i$ on the mesh should have precise mapping in the plane – this will ensure a smooth boundary curve in the plane and the preservation of any important singular vertices on the boundary. It is important to point out that although using very small $r_{weft}$ and $r_{warp}$ (thus increasing the number of mapped nodes) might offer a discrete mapping fine enough for approximation, the computing time, especially that for the energy minimization process, will nevertheless increase tremendously, so will the numerical instability. A more plausible strategy is to use a relatively coarse mesh of nodes and still obtain a smooth boundary and preserve sharp features, through sound interpolation.

**Planar mapping of a vertex**

For any vertex $X_i \in \mathrm{M}$, we determine its closest woven mesh node $V_{I,J} \in \Gamma$, called the *anchor node*, by a local searching. The anchor node should be the one which falls in some 1-ring triangular face around $X_i$. Since the connectivity of $\mathrm{M}$ is fully stored in our data structure, this determination can be done efficiently. After obtaining $V_{I,J}$, a row vector $t_{row}$ is formed by either $V_{I+1,J}V_{I,J}$ or $V_{I-1,J}V_{I,J}$ – choosing $V_{I+1,J}V_{I,J}$ or $V_{I-1,J}V_{I,J}$ depends on the sign of the projection of $X_iV_{I,J}$ on $V_{I+1,J}V_{I,J}$, i.e., $V_{I+1,J}V_{I,J}$ if it is positive, and $V_{I-1,J}V_{I,J}$ otherwise. In the similar manner, a column vector $t_{col}$ is formed by either $V_{I,J+1}V_{I,J}$ or $V_{I,J-1}V_{I,J}$. In case $V_{I,J}$ is a boundary woven node, some of $V_{I\pm1,J}V_{I,J}$ and $V_{I,J\pm1}V_{I,J}$ might not exist, then, the existing one will be taken as the row or column vector. In the worst case, $V_{I,J}$ has neither $V_{I+1,J}$ nor $V_{I-1,J}$ neighbor, or neither $V_{I,J+1}$ nor $V_{I,J-1}$ neighbor, then we have to use other woven node to perform the vertex mapping.

The projections of $X_iV_{I,J}$ on $t_{row}$ and $t_{col}$ are $< X_iV_{I,J}, t_{row} >$ and $< X_iV_{I,J}, t_{col} >$ respectively, and the planar coordinate of $X_i$ is

$$x_i = r_{warp}\left[(J - j_c) \pm \frac{< X_iV_{I,J}, t_{col} >}{\left\| < X_iV_{I,J}, t_{col} > \right\|}\right] \tag{9}$$

$$y_i = r_{weft}\left[(I - i_c) \pm \frac{< X_iV_{I,J}, t_{row} >}{\left\| < X_iV_{I,J}, t_{row} > \right\|}\right] \tag{10}$$

where $i_c$ and $j_c$ is the index of the intersection node of tendons, and the sign of $\pm$ is determined by the direction of $X_i$ relative to $V_{I,J}$. An illustration of the planar coordinate mapping is shown in Fig. 6.

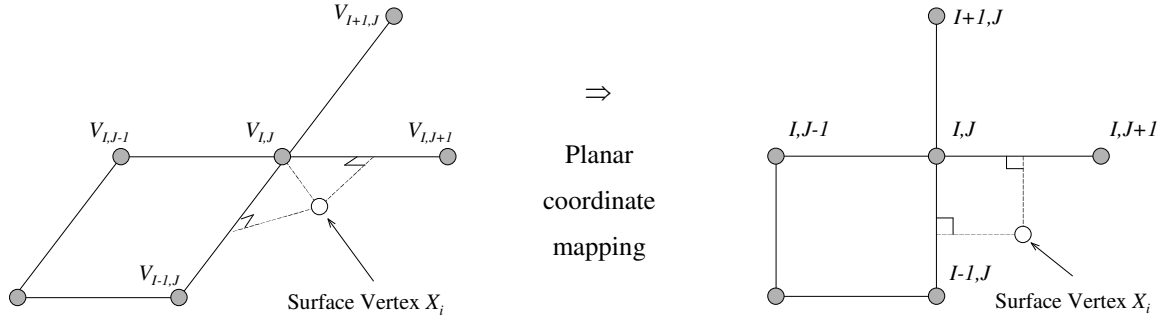



**Fig. 6 Vertex planar coordinate mapping**

**Planar mapping of a dart**

The above mapping mechanism requires special amendment if the vertex to be mapped lies on a dart. Using the same mechanism, a dart would be mapped to a single curve in the plane (e.g., Fig. 7a), the correct solution though should be two disjoint curves (see Fig. 7b). Topology (i.e., connectivity), in addition to geometry, therefore must be considered when a dart is mapped. First of all, the formal definition of a dart vertex is needed.

**Definition 6** $\forall X_i \in \mathrm{M}$, and $X_i$ is on the boundary of M, if any other vertex $X_j \in \mathrm{M}$ can be found coincident at $X_i$, $X_i$ is a *dart vertex*.

For a dart vertex $X_i$, based on its two incident boundary edges $L_1$ and $L_2$, two planes $P_1$ and $P_2$ are decided by $L_1$ and $L_2$ and the normal vectors $n_1$ and $n_2$ on them. $P_1$ and $P_2$ separate the space into two sub-spaces: the interior and exterior space at $X_i$ (see Fig. 8). All the mesh elements surrounding $X_i$ are in the interior space. For mapping a dart vertex, only the woven nodes in its interior space can be the candidates for selection of the anchor node. With this constraint satisfied, the two coincident vertices on different sides of the dart will be correctly mapped to different interior sides.

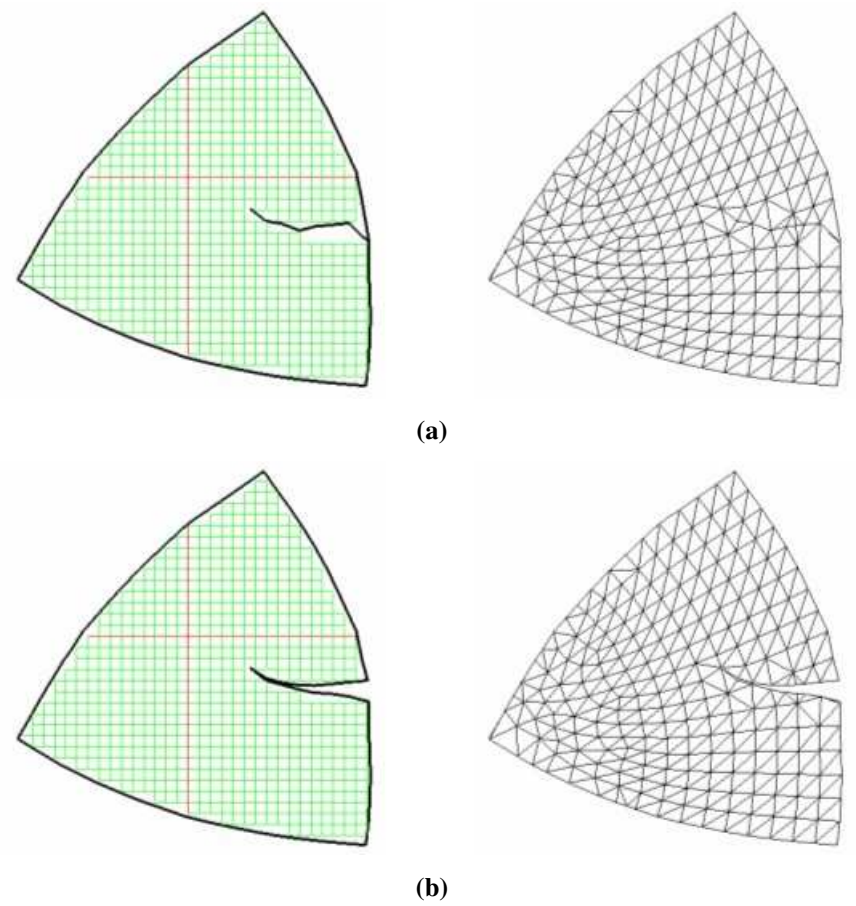



**Fig. 7 Planar mapping of a dart on an octant of a sphere: (a) incorrect mapping result ; (b) correct mapping result after the connectivity topology is considered.**

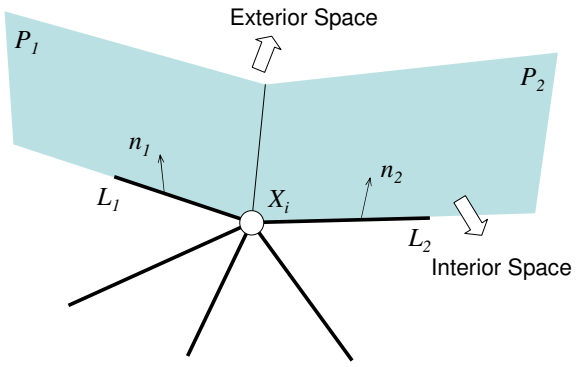



**Fig. 8 The interior and exterior space at a dart vertex**

## 6. Experiments and Discussions

We have fully implemented the proposed woven fitting algorithm, as a standalone software module written with Visual C++. The testing of the following examples is performed on a PC with AMD 2400+ CPU and 512MB RAM, operated by Windows XP. Six examples are presented to illustrate the correctness and practicality of the proposed algorithm. Comparisons between our algorithm and some existing representative

surface flattening algorithms are also provided, in terms of both the accuracy and the computing cost – we have actually implemented the conformal mapping based parameterization algorithm ([14]) and the spring mass based 2D deformation algorithm ([20]). In all our tests, for isotropic material the spring constant ratio is set to be:

$$\text{Weft} : \text{Warp} : \text{Diagonal} = 1 : 1 : 1,$$

and the ratio for anisotropic material is:

$$\text{Weft} : \text{Warp} : \text{Diagonal} = 500 : 500 : 1.$$

For the values of $r_{warp}$ and $r_{weft}$, we choose them both to be half the average edge length on $M$. A good flattening algorithm should try to preserve the boundary length and the surface area on the 2D pattern while achieving reasonably fast computing time. These three factors will be the yardsticks for the comparison, which are listed in Table 1.

The first example is a conical surface, which is often used to test the basic ability of a flattening algorithm, since a cone is a developable surface which can be flattened into a plane without any deformation. Fig. 9 depicts our flattening result. As shown in the figure, the inner and outer circumferences of the 2D pattern correctly agree with the two base circles of the cone (drawn in bolded dash lines, which are added for comparison). The 2D mappings of the vertices of the mesh presentation the cone are also displayed.

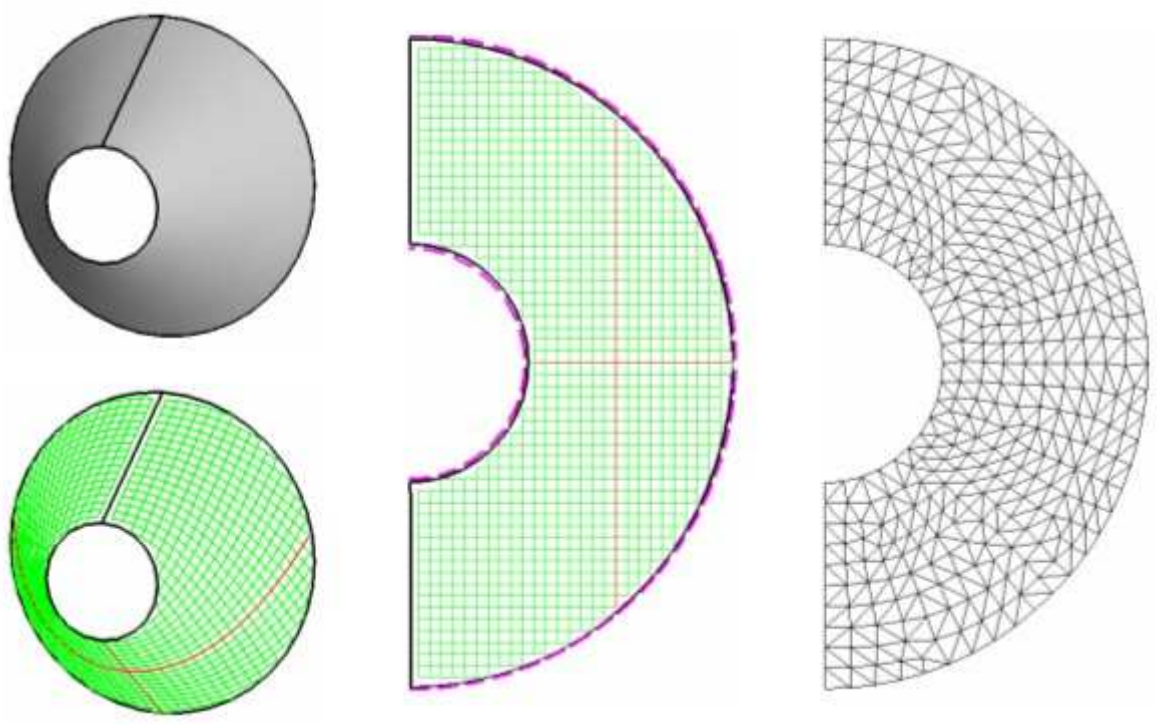


**Fig. 9 Example I: a conical frustum of isotropic material**

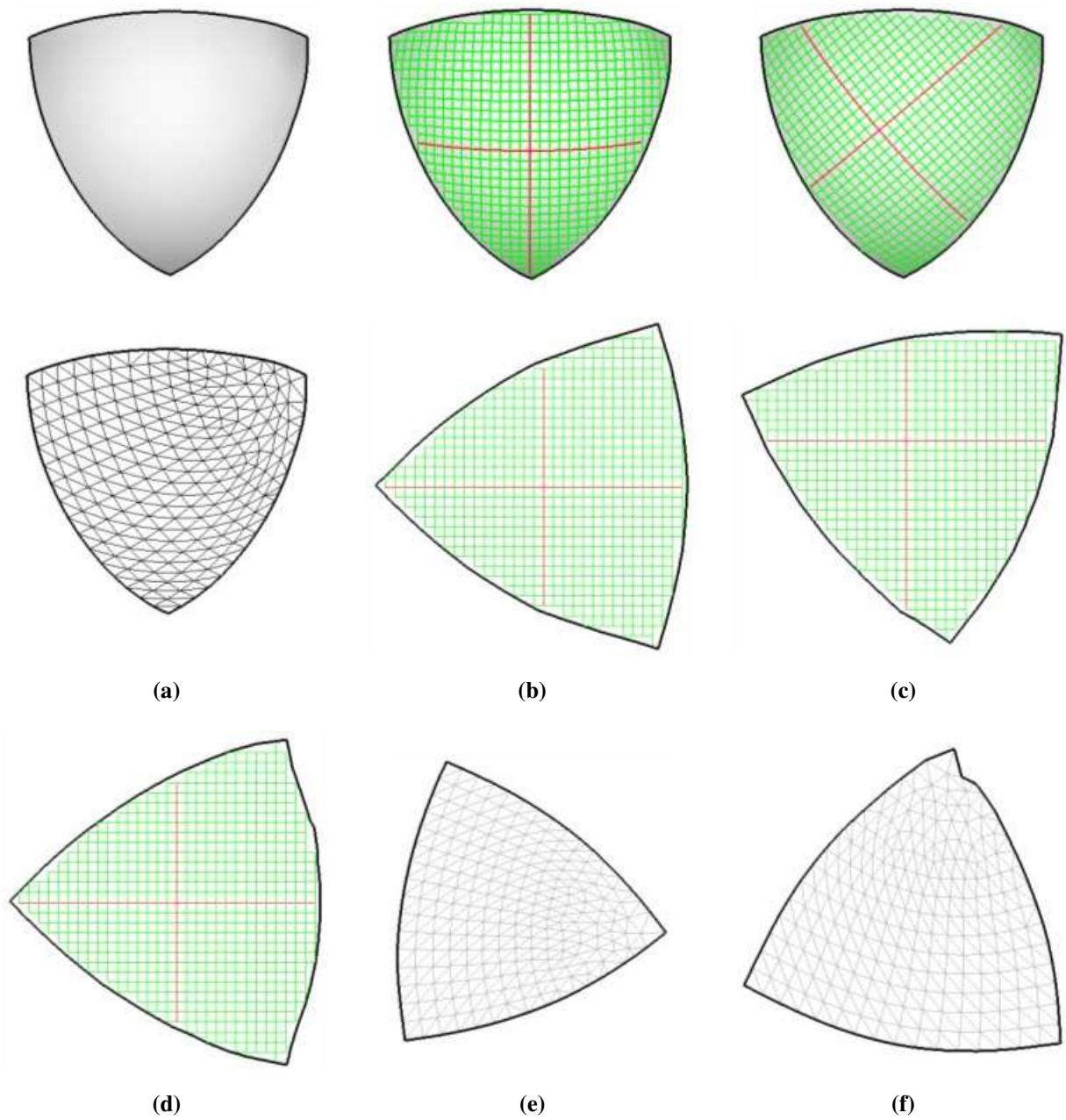



**Fig. 10 Example II: an octant-sphere: (a) the given octant-sphere (shaded and mesh view); (b) the isotropic woven fitting result with orientation I; (c) the isotropic woven fitting result with orientation II; (d) the anisotropic woven fitting result with orientation I; (e) the conformal map flattening result; (f) the spring-mass system flattening result.**

The octant-sphere, which is non-developable, is used to test the algorithm on uniform non-developable surfaces, as well as the effect of dart insertions. Conceivably, for isotropic materials, a good flattening algorithm should generate an identical or similar 2D pattern for a surface regardless of the tendon orientation chosen on the surface. Fig. 10b and 10c show the fitting results of isotropic material on an octant-sphere with two different orientations. The results are as expected – the flattened 2D shapes are identical, only different in orientation (see also the data in Table 1). Fig. 10d displays the fitting result of anisotropic material. This time, one tendon is chosen to be a symmetric great arc through a corner on the octant. Due to the anisotropic properties of our

woven model, the flattened 2D pattern should be symmetric to the 2D counterpart of that tendon, and this is correctly captured in Fig. 10d. To compare ours with others, the flattening results of the conformal mapping based parameterization method ([14]) and the spring-mass based 2D deformation method ([20]) are shown in Fig. 10e and 10f respectively. Since the given surface is non-developable, the flattening result of [14] shrinks severely (with more than 23% area reduction, see Table 1). In the implementation of [14], we fix an arbitrary boundary edge by this spatial length when solving the sparse matrix to obtain a natural boundary. When the surface is non-developable, the whole boundary and the surface area will shrink due to the influence of the fixed boundary edge. About the result shown in Fig. 10f, although the area and boundary length errors are comparable to that of Figs. 10b and 10c, the resultant 2D profile is unsatisfied – with some concave points. Moreover, the two methods from [14] and [20] can only be used on isotropic models; they become inapplicable if the underlying material carries anisotropic woven characteristics, e.g., Fig. 1a.

Our third example, Example III, still on an octant-sphere, demonstrates the effect of the insertion of a dart. The statistics given in Table 1 clearly shows that both the boundary length and area errors are reduced after inserting a dart. This testifies to what is expected from dart insertion on non-developable surfaces. The comparison with the methods of [14] and [20] is also performed. The data in Table 1 shows that the result from [14] did not ameliorate much compared to that without dart insertion. This is because the method in [14] only minimizes the angle distortion, and does not take the linear stretch into consideration. The flattening result from [20] (Fig. 11f) still shows a jaded 2D boundary after the dart insertion. This is due to the fact that the energy-minimization iterations from [20] are performed on the 2D pattern only, ignoring the geometry and topology of the 3D surface after the initial flattening. Our approach is different – we consider the shape information of the 3D surface from the beginning through the end. One way to measure the distortion is by texture mapping. Fig. 11g, 11h and 11i show the surface texture mapping results of our method, that from [14], and [20] respectively. Among the three, the texture mapping of our method demonstrates an overall better quality of uniform distribution and less distortion. The texture mapping in Fig. 11h is uniform, but the textures are enlarged due to the area shrinkage in the 2D pattern. The textures in Fig. 11i are not enlarged; however, some distortion occurs near the right upper corner, a direct result of the jaded boundary curve in the 2D pattern (Fig. 11f).

The 3D surface used in Example IV has been originally given in Fig. 2; it is an arbitrary freeform surface in which the Gaussian curvature varies from zero to some high value. To this surface, the spring-mass flattening method [20] gives the best boundary length accuracy, while our woven fitting approach bestows the best area accuracy (see Table 1). The result of the conformal mapping method [14] is unsatisfactory – with both large area and boundary length errors. In terms of computing time, our algorithm runs as fast as that of [14], but with more accurate result.

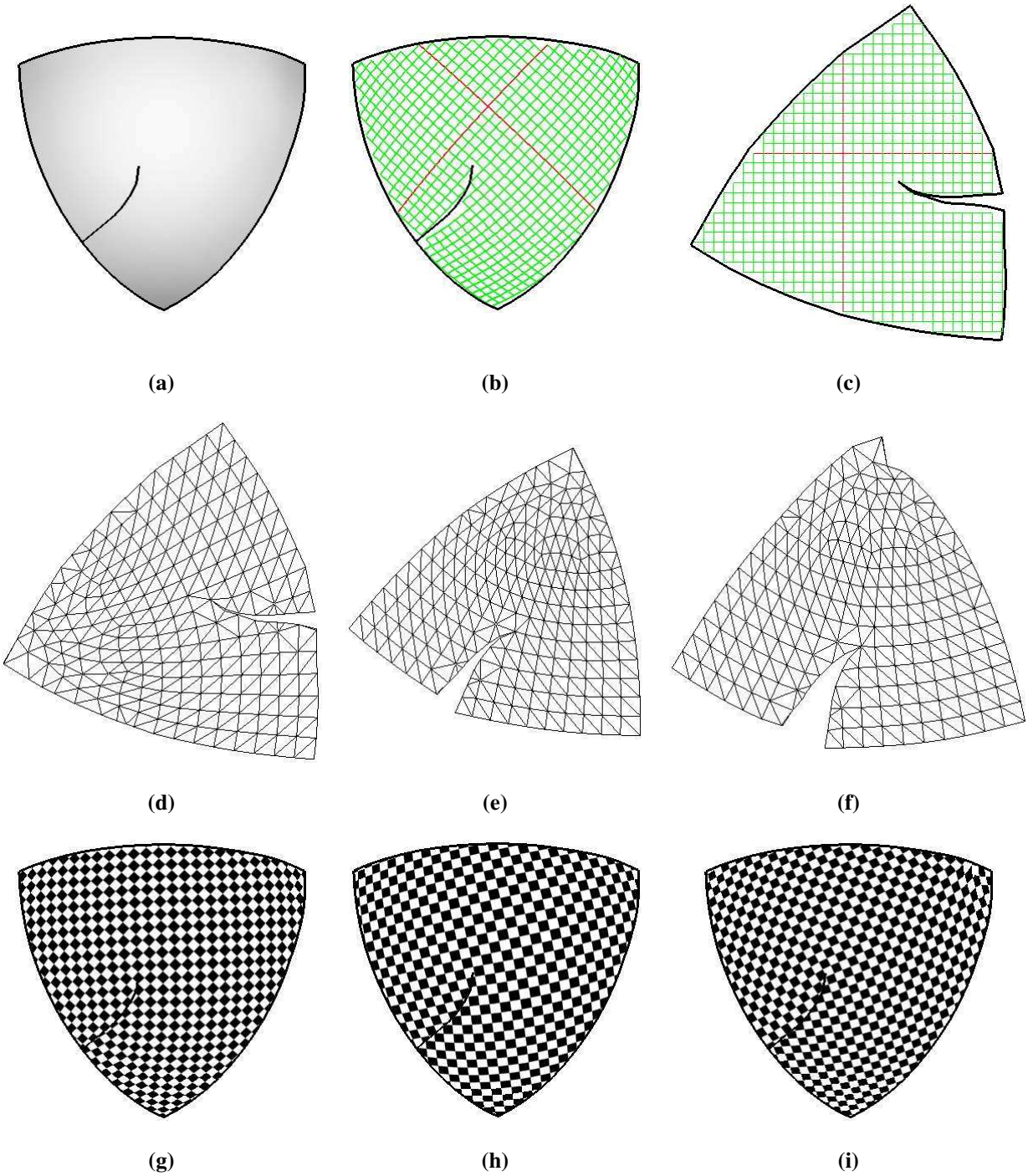
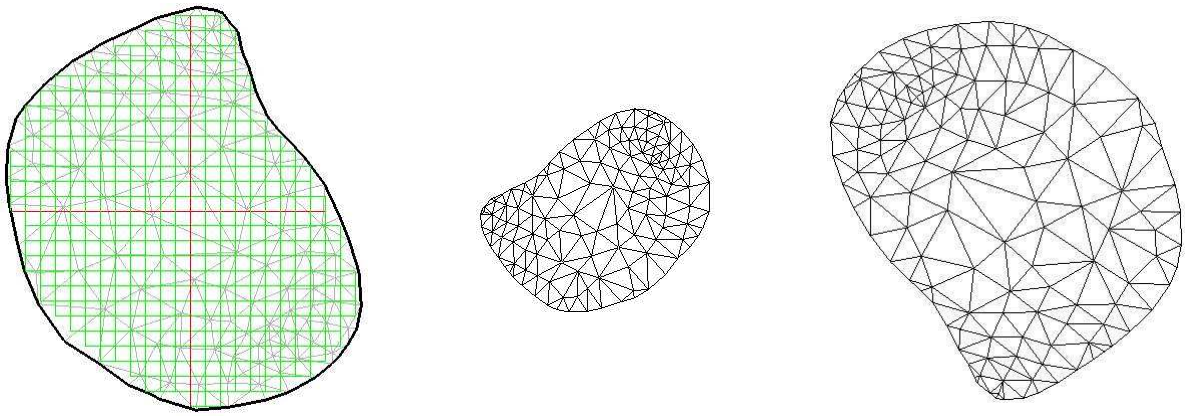


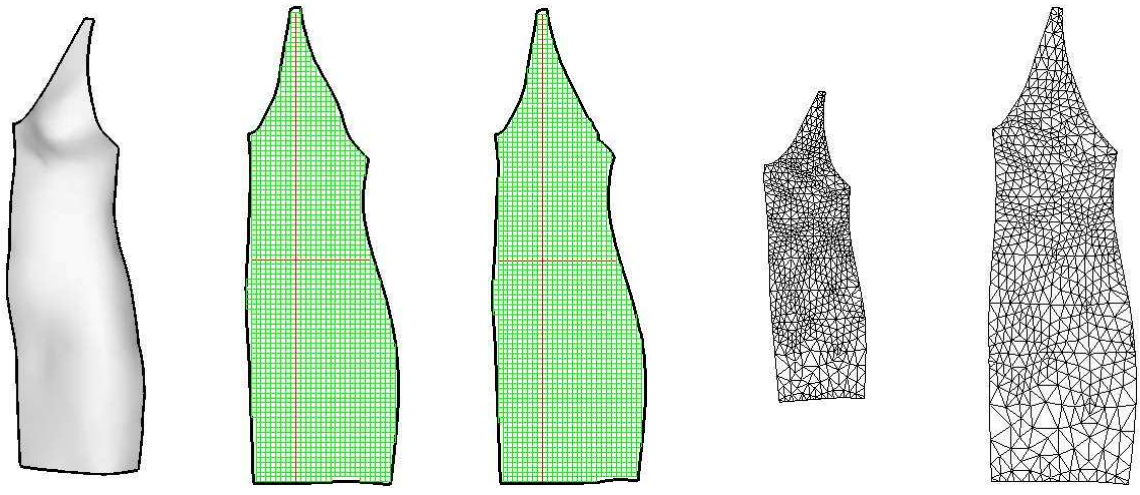

**Fig. 11 Example III: an octant-sphere with dart: (a) the surface with a dart; (b) the fitting isotropic woven model; (c) the flattened 2D pattern; (d) the mapped 2D mesh; (e) the flattened 2D mesh by [14]; (f) the flattened 2D mesh by [20]; (g), (h) and (i): the texture mappings of (d), (e) and (f) respectively.**



**(a)** **(b)** **(c)**

**Fig. 12 Example IV: a freeform surface originally given in Fig. 2: (a) the flattening result by the isotropic woven model fitting; (b) the 2D shape by [14]; (c) the 2D shape by [20].**

Examples V and VI are applications of the proposed flattening algorithm in clothing design. Both isotropic and anisotropic woven models are tested, and they are found to yield similar boundary length and area errors. When compared to the conformal mapping method [14] (for isotropic material only), our algorithm generates more accurate results. As for the spring-mass method [20], it is found to run much slower than the proposed algorithm, even though both achieve a similar accuracy in terms of boundary length and area. Besides, the numerical stability of a spring-mass system crucially depends on the shape quality of the triangular mesh representing $M$: when there are triangles with sharp angles, the diffusion process tends easily to diverge. However, our flattening algorithm is independent of the surface representation, and thus is more robust.



**(a)** **(b)** **(c)** **(d)** **(e)**

**Fig. 13 Example V: a piece of garment: (a) the 3D surface of the garment; (b) the flattening result by the isotropic woven model fitting; (c) the flattening result by the anisotropic woven model fitting; (d) the 2D shape by [14]; (e) the 2D shape by [20].**

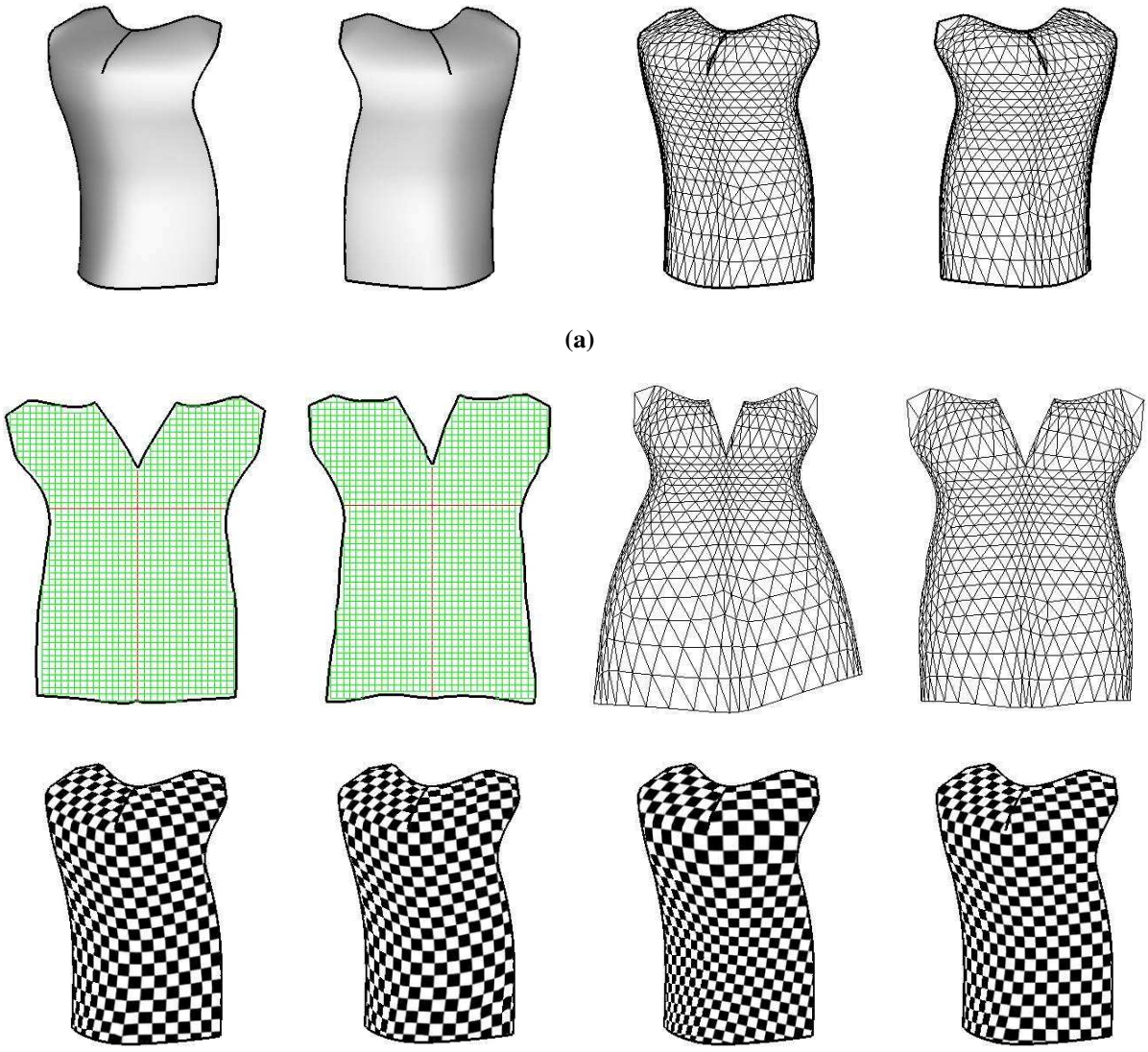



**Fig. 14 Example VI: a dress: (a) the 3D surface of the dress with a dart; (b) the result of isotropic woven model fitting; (c) the result of anisotropic woven model fitting; (d) the 2D pattern and the texture mapping by [14]; (e) the 2D pattern and texture mapping by [20].**

**Table 1 Computational Statistics of test examples**

| Example | Algorithm | Figure | Computing time | Boundary Error (%) | Area Error (%) |
|---|---|---|---|---|---|
| I | ***this*** (isotropic material) | 9 | 3s | 0.05 | 0.25 |
| II | ***this*** (isotropic material) – with orientation I | 10b | 1s | 6.37 | 2.38 |
| | ***this*** (isotropic material) – with orientation II | 10c | 1s | 6.22 | 2.55 |
| | ***this*** (anisotropic material) – with orientation I | 10d | <1s | 5.70 | 2.39 |
| | conformal map (ref. [14]) | 10e | <1s | 7.55 | 23.57 |
| | spring mass (ref. [20]) | 10f | 3s | 4.84 | 0.91 |
| III | ***this*** (isotropic material) | 11c,d | 2s | 3.72 | 1.90 |
| | conformal map (ref. [14]) | 11e | <1s | 7.43 | 20.28 |
| | spring mass (ref. [20]) | 11f | 6s | 1.56 | 0.66 |
| IV | ***this*** (isotropic material) | 12a | <1s | 14.94 | 4.61 |
| | conformal map (ref. [14]) | 12b | <1s | 35.61 | 67.42 |
| | spring mass (ref. [20]) | 12c | 5s | 9.02 | 4.85 |
| V | ***this*** (isotropic material) | 13b | 2s | 1.79 | 0.30 |
| | ***this*** (anisotropic material) | 13c | 2s | 1.75 | 0.38 |
| | conformal map (ref. [14]) | 13d | <1s | 33.7 | 57.66 |
| | spring mass (ref. [20]) | 13e | 22s | 1.84 | 0.41 |
| VI | ***this*** (isotropic material) | 14b | 3s | 3.74 | 1.91 |
| | ***this*** (anisotropic material) | 14c | 2s | 1.85 | 1.82 |
| | conformal map (ref. [14]) | 14d | <1s | 2.56 | 7.90 |
| | spring mass (ref. [20]) | 14e | 15s | 3.84 | 1.12 |

## 7. Summary

A surface flattening method based on fitting a woven mesh model onto a 3D freeform surface is proposed in this paper. The fitting algorithm comprises two mappings followed by an energy minimization. The two mappings are *tendon node mapping* (TNM) and *diagonal node mapping* (DNM), where TNM determines the position of a new node on the surface along the warp or weft direction and DNM locates a node along the diagonal direction. A diffusion process is adopted to release the strain energy in the initially fitted model, where the strain energy measures the length and area deformation between the mapped nodes on the given surface and their counterparts in the plane. After the energy minimization, a planar coordinate mapping is introduced that establishes a continuous mapping between every point on the given freeform surface and its counterpart in the plane. A number of experimental examples are presented to illustrate the proposed flattening algorithm, and the results of our method and other approaches are compared. Our approach has the following advantages:

- We assume zero energy on the 2D pattern and directly minimize the energy on the 3D surface itself – this conforms closer to the physical folding process;
- The woven mesh model used in the proposed approach can support both isotropic and anisotropic materials;
- Nodes mapping and movement in the proposed approach are based on the discrete geodesic curve generation algorithm, so no parametric surface or pre-parameterization is required;
- Insertion of darts is incorporated in the woven mesh model in a natural manner;
- Compared to other surface flattening approaches, the proposed algorithm is found to perform efficiently while preserving the accuracy;
- The topology of the freeform surface will not effect the stability of the proposed algorithm.

Regarding further improvement to the proposed algorithm, one salient subject is how to better model the woven properties. In real woven fabrics, the cloth would shear until it reaches a locking angle at which shearing no longer occurs and instead is replaced by buckling deformation; this parameter can be embedded into the existing algorithm to provide a better approximation. Another issue is the geometric modeling of the woven mesh itself. Currently the woven mesh is represented in a row and column index. Tri-axial woven fabric or complex knitted fabric may not be able to be represented by this index. The indexing system can be modified to model different kinds of woven materials.

## References


[1] Middleton D.H., *Composite Materials in Aircraft Structures*, Longman Group UK Limited, pp.156-182, 1990.

[2] Jing F., Joneja A., Tang K., Modeling wrinkles on smooth surfaces for footwear design, *CAD'04 Conference*, Pattaya Beach, Thailand, May 24-28, 2004.

[3] Wang C.C.L., Wang Y., and Yuen M.M.F., Feature-based 3D non-manifold freeform object construction, *Engineering with Computers*, vol.19, no.2-3, pp.174-190, 2003.

[4] Ravi Kumar G.V.V., Srinivasan P., Devaraja Holla V., Shastry K.G., Prakash B.G., Geodesic curve computations on surfaces, *Computer Aided Geometric Design*, vol.20, no.2, pp.119-133, 2003.

[5] Aono M., Breen D.E., Wozny M.J., Fitting a woven-cloth model to a curved surface: mapping algorithms, *Computer-Aided Design*, vol.26, no.4, pp.278-292, 1994.

[6] Aono M., Denti P., Breen D.E., Wozny M.J., Fitting a woven cloth model to a curved surface: dart insertion, *IEEE Computer Graphics & Applications*, vol.16, no.5, pp.60-70, 1996.

[7] Aono M., Breen D.E., Wozny M.J., Modeling methods for the design of 3D broadcloth composite parts, *Computer-Aided Design*, vol.33, no.13, pp.989-1007, 2001.

[8] Wang C.C.L., Chen S.F., Fan J., and Yuen M.M.F., Two-dimensional trimmed surface development using a physics-based model, 1999 ASME DETC, *25th Design Automation Conference*, Las Vegas, Nevada, September, 1999.

[9] Yeung B.M.L., Tang K., and Wang C.C.L., Fitting a fabric woven model onto a surface based on energy minimization, *CAD'04 Conference*, Pattaya Beach, Thailand, May 24-28, 2004.

[10] Floater M.S., and Hormann K., Recent advances in surface parameterization, In *Multiresolution in Geometric Modeling*, pp.259-284, 2003.

[11] Floater M.S., Parametrization and smooth approximation of surface triangulations, *Computer Aided Geometric Design*, vol.14, no.3, April 1997, pp.231-71.

[12] Hormann K., and Greiner G., Mips: an efficient global parameterization method, In *Curve and Surface Design: St. Malo 1999*, Vanderbilt University Press, pp.153–162, 1999.

[13] Levy B., Petitjean S., Ray N., and Maillot J., Least squares conformal maps for automatic texture atlas generation, *SIGGRAPH 2002*, *ACM Transactions on Graphics*, vol.21, no.3, pp.362-71, 2002.

[14] Desbrun M., Meyer M., and Alliez P., Intrinsic parameterizations of surface meshes, *Eurographics 2002*, *Computer Graphics Forum*, vol.21, no.3, pp.209-218, 2002.

[15] Sheffer A., and de Sturler E., Parameterization of Faceted Surfaces for Meshing Using Angle Based Flattening, *Engineering with Computers*, vol.17, no.3, pp.326-337, 2001.

[16] Sheffer A., and de Sturler E., Smoothing an Overlay Grid to Minimize Linear Distortion in Texture Mapping, *ACM Transactions on Graphics*, vol.21, no.4, pp.874-890, 2002.

[17] Sander P.V., Snyder J., Gortler S.J., and Hoppe H., Texture mapping progressive meshes, *SIGGRAPH 2001 Conference Proceeding*, pp.409-416, 2001.

[18] Yoshizawa S., Belyaev A., and Seidel H.P., A fast and simple stretch-minimizing mesh parameterization, *International Conference on Shape Modeling and Applications 2004*.

[19] McCartney J., Hinds B.K., and Seow B.L., The flattening of triangulated surfaces incorporating darts and gussets, *Computer-Aided Design*, vol.31, pp.249-260, 1999.

[20] Wang C.C.L., Chen S.S.F., and Yuen M.M.F., Surface flattening based on energy model, *Computer-Aided Design*, vol.34, no.11, pp.823-833, 2002.

[21] Azariadis P.N., and Aspragathos N.A., Design of plane development of doubly curved surface, *Computer-Aided Design*, vol.29, no.10, pp.675-685, 1997.

[22] Azariadis P.N., and Aspragathos N.A., Geodesic curvature preservation in surface flattening through constrained global optimization, *Computer-Aided Design*, vol.33, no.8, pp.581-591, 2001.

[23] Bennis C., Vezien J.M., and Iglesias G., Piecewise surface flattening for non-distorted texture mapping, *ACM SIGGRAPH Computer Graphics*, v.25 n.4, pp.237-246, 1991.

[24] McCartney J., Hinds B.K., Seow B.L., and Gong D., An energy based model for the flattening of woven fabrics, *Journal of Materials Processing Technology*, v.107, pp.312-318, 2000.

[25] Parida L., Mudur S.P., Constraint-satisfying planar development of complex surfaces, *Computer-Aided Design*, vol.25, no.4, pp225-232, 1993.

[26] Sheffer A., Spanning tree seams for reducing parameterization distortion of triangulated surface, *SMI 2002: International Conference on Shape Modelling and Applications.*

[27] Wang C.C.L., Wang Y., Tang K., Yuen M.M.F., Reduce the stretch in surface flattening by finding cutting paths to the surface boundary, *Computer-Aided Design*, vol.36, no.8, pp.665-677, 2004.

[28] Katz S., Tal A., Hierarchical mesh decomposition using fuzzy clustering and cuts, *SIGGRAPH 2003*, *ACM Transactions on Graphics*, v.22, no.3, July 2003, 954-961.

[29] Dowling N.E., *Mechanical Behavior of Materials*, Prentice-Hall, Inc. NY, 1993.

[30] do Carmo M.P., *Differential Geometry of Curves and Surfaces*, Englewood Cliffs, N.J.: Prentice-Hall, 1976.

[31] Kobbelt L., Discrete fairing and variational subdivision for freeform surface design, *The Visual Computer*, vol.16, no.3-4, pp.142-158, 2000.

## Appendix A Discrete Geodesic Path Computing

The discrete geodesic is computed from a given starting $p_0$ while moving along a direction specified by a vector $t_0$. The pseudo-codes of the discrete geodesic path generation algorithm (ref. [4]) are outlined below.

***Algorithm*** ComputeDiscreteGeodesicPath( $p_0$, $t_0$, M, $L$ )

*Input*: a polygonal mesh surface M, a starting point $p_0$ on M, a user specified direction vector $t_0$, and a specified length of the geodesic path.

*Output*: a sequence of points $\varpi$ describing the discrete geodesic path.

1. $\varpi \leftarrow \phi$, $p^* \leftarrow p_0$, $t^* \leftarrow t_0$ and $l \leftarrow 0$;

2. **while** ( $l < L$ ) {

3. Add $p^*$ at the tail of $\varpi$ ;

4. $n_i \leftarrow$ Normal( $p^*$, M ); // To compute the tessellation normal $n_i$ at the last point $p_i$ in $\varpi$ ;

5. Construct a half-plane $\Gamma$ containing the vectors $n_i$, $t^*$ and passing through point $p^*$;

6. Obtain an intersection line of $\Gamma$ and a face of M which starts at $p^*$ and ends at $p_{new}$ ;

7. $t^* \leftarrow p_{new}p^* / \|p_{new}p^*\|$ and $l \leftarrow l + \|p_{new}p^*\|$ ;

8. **if** ( $l \geq L$ ) **then** $p_{new} \leftarrow p^* + (L - (l - \|p_{new}p^*\|)) \frac{p_{new}p^*}{\|p_{new}p^*\|}$ and $l \leftarrow L$ ;

9. $p^* \leftarrow p_{new}$ ;

10. **if** ( ( $p_{new}$ is at the boundary of M ) **AND** ( $l < L$ ) ) {

11. Send the boundary arrived warning;

12. Add $p^*$ at the tail of $\varpi$ ;

13. **return** $\varpi$ ;

14. }

15. }

16. **return** $\varpi$ ;

***Function*** Normal( $p$, M )

*Input*: a polygonal mesh surface M , a point $p$ on M .

*Output*: the tessellation normal $n$ at $p$.

1. **if** ($p$ lies inside a face $F$) **then** $n$ is the normal of $F$;

2. **if** ($p$ lies on a edge $E$) {

3. **if** ($E$ is a boundary edge)

4. $n$ is the normal of the face containing $E$;

5. **else**

6. $n$ is the average normal of $E$'s left and right faces;

7. }

8. **if** ($p$ lies on a vertex $V$) **then** $n$ is the average normal of all faces incident on $V$;

9. **return** $n$;