Fitting a Woven Fabric Model onto a Surface Based on Energy Minimization

Benjamin M. L. Yeung[1], Kai Tang[1] and Charlie C. L. Wang[2]

[1]Hong Kong University of Science and Technology, mektang@ust.hk
[2]Chinese University of Hong Kong, cwang@acae.cuhk.edu.hk

ABSTRACT

Surface flattening is applied in many applications, such as in the aircraft, shoe, and garment industries. In this paper, a surface flattening algorithm based on the energy-minimization on the 3D surface is presented, which calculates the optimal 2D pattern that folds to the 3D surface with minimum deformation. Both isotropic and woven-like anisotropic materials are supported by the algorithm. Darts insertion is also incorporated in the algorithm to further enhance the flattening result.

Keywords: surface flattening, woven fabric model, 3D fitting and energy minimization.

1. INTRODUCTION

In many applications, such as shoe and aircraft industries, surface flattening is one important process. In the shoe industry, the profile of the shoe upper leather is first estimated and then cut out, then the pieces of leather are sewed together and a foot shape mould is inserted in to let the leather to deform into a desired shape. In the aircraft industry, there are many structures reinforced by fabrics and woven fabric is commonly used [1]. Profiles of the woven fabric are estimated and cut out, and then they are laid onto a certain 3D shape. In both processes, the profile of the material is still estimated by human in some factories based on trial-and-error and this estimation is quite time consuming and not accurate. In order to obtain an accurate profile, surface flattening algorithms are developed in the past few decades. The task of a surface flattening algorithm is this: given a 3D free-form surface and the material properties, find its counter-part 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.

The flattening of triangular mesh is also a key problem in parameterization and texture mapping. Floater [2] investigated a graph-theory based parameterization for tessellated surfaces for the purpose of smooth surface fitting; his parameterization – actually planar triangulation is the solution of linear systems based on convex combination. In [3], 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 very 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 [4], where the defined objective function minimizes angle deformation. Sheffer and De Sturler [5] also 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 so 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.

McCartney et al. [6] flatten a triangulated surface by minimizing the strain energy in the 2D pattern. The surface is first triangulated using Delaunay triangulation. Then the triangles are transformed onto a 2D plane. However, there are some flattened triangles that cannot preserve their length relationship with respect to the triangles on the surface. This length differences are measured as strain energy. If the strain energy is zero, that means the flattened triangle 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. [7] improve McCartney's algorithm by using a spring-mass system. This 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. [8-11]). 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 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.

Woven fabrics consist of a series of vertical threads (warp) with crossing with a series of horizontal threads (weft). The strength of the threads is usually strong and the threads resist deforming under force, but the shear deformation at the crossing between a warp thread and a weft thread can occur. Three assumptions are made to model a ply of woven fabrics (Aono et al. [12-14]):

- The warp and weft threads are inextensible;
- A thread segment between adjacent crossings is straight on the surface;
- No slippage occurs at a crossing when the ply is deformed.

The algorithm of Aono et al. is a geometrical approach for flattening a woven-cloth 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 [13, 14]. 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. Also the shape of the darts on the surface cannot be controlled.

Some approaches in literature considered the issue of where to insert cuts in flattening triangulations. Parida and Mudur (1993) [15] 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 surface by flattening triangles, 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. [7] generates the cutting line from the stretch energy distribution map; however, the length of cutting paths is not considered in their paper. In [16], 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. [17] developed an technique that computes the shortest path from a node to the surface boundary in linear time; and the cutting paths on the surface with widely distributed curvatures are generated while preventing flipped triangles in surface flattening. Recently, Katz and Tal [18] proposed a hierarchical mesh decomposition algorithm that decomposes a given mesh into meaningful components referring to segmentation at regions of deep concavities. Obviously, this cut insertion technique cannot be directly applied to reduce stretches in flattening.

Motivated by the issues with the existing flattening algorithms as discussed above, in this paper, we present an energy-minimization based surface flattening algorithm that distinguishes from the existing flattening algorithms in (1) the energy minimization is conducted "forward", that is, it is applied on the 3D surface, not in the plane; (2) it supports both isotropic and woven materials; (3) the insertion of cuts is also incorporated into the algorithm as an option to further reduce the energy.

The structure of this paper is organized as follows. In section 2, the mesh model and the definitions used in our algorithm are preliminarily described. The methodology of finding the optimal pattern is presented in section 3, which includes the initialization of the mesh on surface, the energy minimization by using the conjugate gradient method and the steepest descent method, the control of the mesh size adaptively and the handling of the insertion of cut. Some graphical examples are illustrated in section 4. Finally, section 5 concludes the paper and indicates the direction for future research.

2. PRELIMINARIES

The problem to be solved in this paper is to fit a planar woven fabric model (mesh inextensible in warp and weft direction) onto a specified region on a 3D parametric surface so that the 2D shape of the fitted fabric piece can be determined. During the entire fitting procedure

(including the energy minimization), all nodes of the fabric model lie on the given 3D surface exactly. Dart insertion is given as an option to gain more accurate fitting results; it is defined manually in our approach.

In the following of this section, we lay out necessary preliminaries and definitions. They include the assumptions and the components in the model, the identification of an internal node and a boundary node, the energy definition and the discussion in choosing a right combination of spring constants to model isotropic or woven material.

2.1 Definition of the Model

The model used in our system is for plain weave woven fabrics. As already alluded earlier, three assumptions are made with this model: 1) the weft threads and the warp threads are not extendable, 2) no slippage occurs at the crossing of a weft and a warp thread, and 3) a thread between two adjacent crossings is mapped to a straight line segment on the 3D surface. The woven fabric is modeled by a spring mesh. An example unit cell of this model is shown in Fig.1.

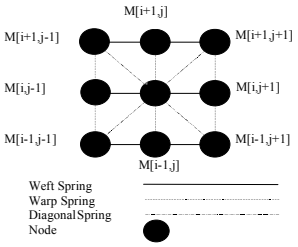


Fig. 1. Unit cell of the spring mesh.

There are three components in this model, weft (horizontal) spring, warp (vertical) spring and diagonal spring. For real woven fabrics, there are no diagonal threads. The reason of introducing diagonal springs in the model is explained in section 2.2. Each of three types of the spring has its own initial length at which the spring stores zero energy. The intersection between springs is called a node whose position determines the deformation of the springs connected to that node. Each node is indexed by M[i, j], where i, j are integers representing the row and column respectively. As shown in Fig.1, the internal nodes have either four or eight springs connected. A boundary node is one with neither eight nor four springs connected. For a cloth model in 2D, all the weft springs are aligned in one direction and all the warp springs are aligned in another direction. In this 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 defined as

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

When the spring mesh is mapped onto the 3D freeform surface, the directions of different kinds of springs and their lengths may not be preserved. Every node is on the freeform surface precisely. A node near the boundary of the freeform surface has less than eight or four springs connected and becomes a boundary node. Those boundary nodes are used to approximate the corresponding 2D boundary of the cloth model and the algorithm for achieving this mapping and approximation will be discussed in the later chapters.

2.2 Definition of Energy and Deformation

The spring energy can be expressed as follows:

$$en_i = \frac{1}{2} k_i (\Delta l_i)^2 \tag{2}$$

where en_i is the ith spring energy (i = 0, 1, …, n); n is the total number of springs in the spring mesh; k_i is the spring constant of the ith spring; and Δl_i is the deformation of the ith spring.

The principle of setting the spring constant ratio between them is similar to the case in making a laminate, in which, layers of materials stacked together. For isotropic cloth model, the counterpart in the laminate system is quasi-isotropic laminate [15], which can be produced by stacking the same material with orientation at -45°, 0°, +45° and 90°. Thus, the diagonal spring in isotropic cloth model reflects the tensile deformation in the cloth model. In order to model the inextensibility of the weft and the warp threads in real woven fabric, 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.

2.3 Definition of Directional Energy

The energy minimization algorithm finds the local minimum with respect to the direction of the initial cloth model on surface as a flattened pattern. When handling anisotropic material, if the direction of the material is not specified on the surface, the flattened pattern may depend on the position of the initial cloth model.

In order to obtain a unique flattened pattern for anisotropic material, a directional energy term is proposed. The concept of directional energy is the measure of the deviation of a specified weft yarn or warp yarn from a direction. Yarn means a series of nodes in the same row for the weft yarn or in the same column for the warp yarn, and the direction is defined as a plane function. The directional energy (*Dir*) is defined as follows.

If M[i,j] is in the specified weft yarn or warp yarn, then,

$$Dir_{M[i,j]} = k\left|\alpha x_{M[i,j]} + \beta y_{M[i,j]} + \gamma z_{M[i,j]} + d\right|, \quad (3)$$

else

$$Dir_{M[i,j]} = 0,$$

where the plane of direction is $f = \alpha x + \beta y + \gamma z + d$; i or j is fixed if the weft yarn or the warp yarn wanted to be directional is specified. k is a constant to adjust the degree of the specified yarn following a specified direction. If k is set to be very large compared with the spring constant in the cloth model, then the specified yarn is forced to form a plane curve.

The total energy of the spring mesh (*EN*) is the summation of energy of all the springs, i.e.,

$$EN = \sum_{i=1}^{n} en_i + \sum_{i=1}^{row\_max} \sum_{j=1}^{col\_max} Dir_{M[i,j]}. \quad (4)$$

The energy measured is the energy 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.

3. METHODOLOGY

A configuration of a piece of woven fabric on a freeform surface is said to be an optimum if its deformation is the minimum. Correspondingly, an optimal configuration for the spring mesh is one with its mesh energy minimized. In order to solve this minimization problem, the conjugate gradient method is applied. Before the energy minimization starts, an initial spring mesh needs to be fitted on the freeform surface. After the energy of the mesh is minimized, dart(s) may be inserted in certain higher deformation regions to further minimize the spring mesh energy. Lastly, the boundary of the cloth model is smoothly approximated.

3.1 Initial Mesh Fitting

A good initial mesh on a freeform surface can enhance the speed of the energy minimization process and prevent the occurrence of unfavorable results. There are three criteria for a good initial mesh: 1) the lengths of the weft and the warp springs should be at their initial length; 2) no overlapping occurs; and 3) the mesh should cover the entire freeform surface. The procedure we take for finding an initial fitting mesh is similar to that of Aono et al. [12], but with certain variations. It consists of three steps: first, nodes are mapped onto two constrained paths; the rest of the nodes are then mapped based on the nodes on the constrained paths; finally, nodes are filled into the unmapped region on the given surface.

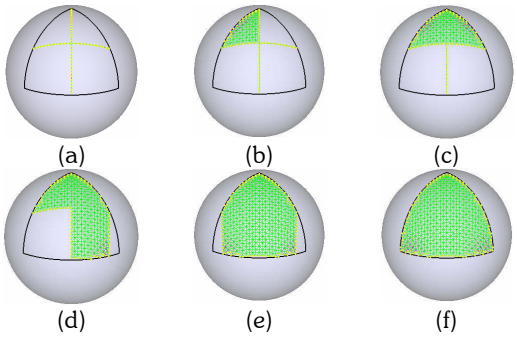


Fig. 2. Initial mesh fitting

A. Mapping of the constrained paths

Two constrained paths are mapped by equidistance nodes in two different directions, representing a series of weft springs and a series of warp springs. This mapping causes no deformation in the springs. In the current implementation, the two constrained paths are defined by the intersection curves of two orthogonal planes with the surface, with their positions selected in a way such that their intersection point is close to the center of the surface (cf. Fig. 2). For example, as an approximation, for a surface which is parameterized into u and v, where $0<u<1$, and $0<v<1$, the intersection point can be chosen to be at $u=0.5$ and $v=0.5$ and the normal of the plane can be defined as a vector parallel to the tangent plane of the surface at the intersection point with x component or y component equaling to zero. An equidistance node is calculated by solving the solution of the intersection point between a sphere (with radius, r_{weft} or r_{warp}), a plane, and a surface. Newton-Raphson method is applied to obtain the solution [12]. This mapping is called *One-Neighbour Mapping* in the rest of this paper.

One-Neighbor Mapping Formulation

Refer to Fig. 3. Suppose we want to map a point (x, y, z) with distance r from a point (x_0, y_0, z_0), which fits into a plane (normal = [Nx, Ny, Nz]) and on a surface parameterized in u, v (x(u,v), y(u,v), z(u,v)). This becomes a problem of two equations solving for two variables. Specifically, we have

Sphere equation:

$$F_{sphere} = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0 \quad (5)$$

Plane equation:

$$F_{plane} = N_x \times (x - x_0) + N_y \times (y - y_0) + N_z \times (z - z_0) = 0 . \quad (6)$$

Point (x, y, z) can be expressed as (x(u,v), y(u,v), z(u,v)). Then, u and v are the two variables to be solved.

B. Mapping nodes based on the constrained paths

As illustrated in Fig. 2, the surface is divided into four quadrants. Nodes are mapped on the surface quadrant by quadrant, row by row and column by column. A new node is mapped such that the weft spring and the warp spring connected to that node have no deformation. The position of a new node on the surface can be calculated by finding the intersection point of two spheres (with radii, r_{weft} and r_{warp}) and the surface. This is called *Two-Neighbour Mapping* in the rest of this paper.

Two-Neighbour Mapping Formulation

Refer to Fig. 4. Suppose we want to map a point (x, y, z) with distance r_{weft} from a point (x_a, y_a, z_a), with distance r_{warp} from a point (x_b, y_b, z_b) and on a surface parameterized in *u*, *v* *(x(u,v), y(u,v), z(u,v))*. This becomes a problem of two equations with two variables:

Sphere equation:

$$F_{sphereA} = (x - x_a)^2 + (y - y_a)^2 + (z - z_a)^2 - r_{weft}^2 = 0 \quad (7)$$

Plane equation:

$$F_{sphereB} = (x - x_b)^2 + (y - y_b)^2 + (z - z_b)^2 - r_{warp}^2 = 0 \quad (8)$$

Point (x, y, z) can be expressed as (*x*(*u*,*v*), *y*(*u*,*v*), *z*(*u*,*v*)). Then, u and v are the two variables to be solved. Newton method is applied to solve this problem.

C. Node filling

Since the mapping is performed based on the constrained paths and in a sequential manner, for some special surfaces, the prescribed mapping procedures cannot fully fill the surface. This usually happens when the specified region to be filled on the given surface is quite different from a rectangle.

Referring to Fig. 5, the unmapped region is due to the absence of node on the previous row or column when performing the Two-Neighbor Mapping. The Node Filling algorithm helps solve this problem. The idea of the Node Filling algorithm is to use *One-Neighbor Mapping* to approximate a new node by a boundary node which is not close to the boundary of the surface and then *Two-Neighbor Mapping* is utilized to map other new nodes. Fig. 5 is an example of the Node Filling process. Node2 and node4 are the boundary nodes, named as M[i,j] and M[i+1,j] respectively. First, node3 is mapped by One-Neighbor Mapping, which is applied on node2 with radius r_{weft} and a plane parallel to the direction of node2 and node1. The normal of the plane is defined as follows.

Assume there exists a node M[i,j], and the node M[i+a,j+b], where a,b∈{–1,0,1} and |a|+|b|=1, is going to be mapped based on the direction between the nodes M[i-a,j-b] and M[i,j]. Then the center of the sphere is at M[i,j] and the radius is r_{weft} or r_{warp}. The normal of the plane, *N*, can be defined by the cross product of the unit normal $n_{i,j}$ at M[*i,j*] on the freeform surface and the unit vector parallel to M[*i,j*]- M[*i-a,j-b*], i.e.,

$$N = n_{i,j} \times \frac{(M[i,j] - M[i-a, j-b])}{\left\| M[i,j] - M[i-a, j-b] \right\|} . \quad (9)$$

Then, node5 is mapped by *Two-Neighbor Mapping*, which is based on node4 and the new node node3. The Node Filling algorithm is only applied on the boundary nodes of the mapped region and it is iterated until no new node is created in that iteration.

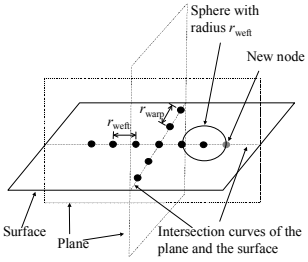


Fig. 3. One-neighbor mapping

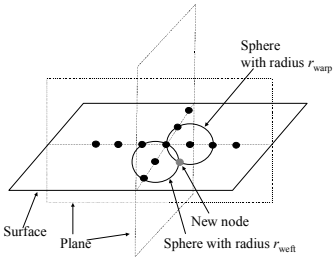


Fig. 4. Two-neighbor mapping

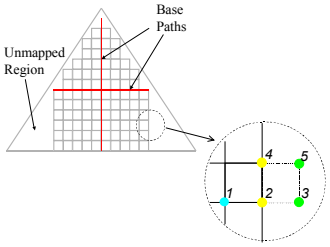


Fig. 5. Node fitting on a given surface

3.2 Energy minimization

Due to the choice of the direction and the mapping sequence of the constrained base paths, the energy of the initial spring mesh is usually not at the minimum state, even for a developable surface. The boundary in 2D at this stage would not be the correct boundary. Thus, energy minimization is applied to obtain a better boundary.

The freeform surface used in our system is parameterized in u and v as a NURBS. Thus, for each node on the surface, it can be expressed by two variables, u and v. For the spring mesh with population N, there are 2*N* variables. Let X_i be the solution vector at the i^{th} iteration

$$X_i = [node1, node2 \cdots, nodeN]^T . \quad (10)$$

Since node *i* can be expressed by (u_i,v_i), thus equation (10) can be rewritten as

$$X_i = [u_1, v_1, u_2, v_2 \cdots u_N, v_N]^T . \quad (11)$$

The steepest descent method and conjugate gradient method are used to solve the minimization problem for these 2*N* variables in equation (11). Since some nodes may leave the surface during the iterations of energy minimization, we also define a mechanism to adaptively control the population of the nodes on the surface. The minimization process is terminated when the change of the total energy *EN* is within a small tolerance □ and the population of the nodes no longer changes.

The steepest descent method and the conjugate gradient method are iterative methods and their general formulation can be written as

$$X_{i+1} = X_i + \alpha D_i \quad (12)$$

where *D* is the direction vector with the same dimension as the solution vector *X*, and α is a positive scalar to be chosen, such that the spring mesh energy with variables in X_{i+1} is minimized. This direction vector provides a downhill direction, but does not guarantee the function to reach the minimum at this direction. Thus, we have to find an α which minimizes the function value in direction *D*. The α can be found by using golden section search. The difference between the two methods lies in the formulation of the direction vector, as described below.

Direction vector for Steepest Gradient Method:

$$D_i = -\nabla EN(X_i), \quad (13)$$

which is the gradient direction at X_i. The convergence rate of the steepest descent method is very slow, but it only needs the knowledge of the derivatives of the variable in the recent iterations.

Direction vector for Conjugate Gradient Method

$$D_{i+1} = -\nabla f_{i+1} - \beta_i D_i \quad (14)$$

$$\beta_i = \frac{\nabla f_{i+1}^T \cdot \nabla f_{i+1}}{\nabla f_i^T \cdot \nabla f_i} \quad (15)$$

There are generally two versions for β and equation (14) is the Fletcher-Reeves version. As shown by the equations, the direction vector needs the direction vector and the gradient vector from the previous iteration, thus usually the steepest descent method is run once before it to provide the direction and the gradient vectors. For further details of the steepest gradient and conjugate gradient methods, it is suggested to refer to the references written by Belegundu et al. [19] and Miller [20].

3.3 Adaptive population control

During the iteration, the positions of nodes will move and some of them may move outside the boundary of the freeform surface. Our freeform surface is defined by NURBS; if the parameter moves outside the defined ranged or forming springs crossing dart (to be discussed in the next session), then the position will be undefined. In this case, the node is deleted. Then the node population will change and the □ in the conjugate gradient method cannot be determined. To deal with this, the steepest descent method is run once after each population change.

In some cases, the initial spring mesh may shrink during the iteration, which leads some portions of the surface not be covered by the mesh. In order to cover the entire specified surface region, new nodes are inserted to fill those uncovered regions; the insertion is performed by the node-filling algorithm.

The configuration, i.e. the M[i,j] neighboring node relationship, of the spring mesh in 2D is the same as that on the freeform surface. However, the pattern, i.e. the area and the boundary, of the spring mesh in 2D 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 in the spring mesh on the surface but not in the mesh in 2D which is considered to be totally relaxed

3.4 Dart insertion

For a non-developable surface, the deformation of the spring mesh is seen to concentrate in highly elliptic and hyperbolic regions. In the regions with severe spring deformation, i.e. with higher spring energy, dart(s) can be inserted to release the energy, i.e. reduce the deformation. A dart on a cloth model means the springs, which are across the dart, are deleted and those nodes near the dart are defined as positive dart nodes, negative dart nodes or boundary nodes. There is no spring

formed between two different dart nodes. After the dart(s) is inserted, the model has more freedom to move and the energy minimization is run to further minimize the energy to obtain a better pattern. Aono et al. [14] suggested a dart or polygon cut could be applied on the flattened pattern. However, this approach cannot control the shape of the dart on surface. In order to control the shape of the dart on surface, a plane dart and a curve dart are proposed. Although a plane dart can be modeled as a curve dart, it is developed due to its simpler formulation

A. Plane dart

The definition of a plane dart consists of a plane equation and its boundary box. A plane equation can be expressed as

$$F = \alpha x + \beta y + \gamma z + d \,, \tag{16}$$

where α, β, γ are the components of the normal of the plane and d is the distance between the plane and the origin. Six values are stored as a boundary box for the plane dart; they are x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}. With the above information, the plane dart is defined as an intersection curve of the plane and the surface within the boundary box. A spring is sait to *cross* a plane dart if one of its two end points, node1 and node2, is inside the boundary box and F[node1]*F[node2]<0. If a spring crosses the dart, its end nodes are then respectively labeled as a *positive* dart node and a *negative* dart node.

B. Curve dart

A curve dart is conducted to handle a dart whose shape is not a plane curve. The shape of a curve dart is first designed by the user on a plane. Then the plane curve is projected onto the surface along the viewing angle of the user, so the projected curve has the shape of the plane curve viewed by the user. Since the projected curve is on the surface, so the surface curve is represented as a function of u and v. Finally, the surface curve is approximated by a number of biarcs [21] in u-v domain. A biarc approximation is used because it requires less number of biarcs to approximate a curve within a prescribed tolerance compared with the number of data points. Biarcs can also be converted into NURBS representation easily, its control polygon consists for two isosceles triangles and each triangle is a control polygon of an arc. The checking of whether a spring crosses a dart is performed in the u-v domain with the assumption that the spring in the u-v domain is also a line segment. If the surface is parameterized in a highly irregularly way, this assumption may only affect the definition of the springs (if it is crossing the dart) at the end point of the dart curve.

3.5 Boundary approximation

After the energy of the spring mesh is minimized, the 2D boundary of the mesh should be approximated for smoothness. The simplest way to obtain the boundary is to connect all the boundary nodes together to form a closed polygon. However, a very fine mesh is needed to obtain a smooth boundary. If the boundary of the surface has a corner but there is no node mapped to this corner, then this corner feature would be lost on the 2D mesh.

In order to use a coarse mesh to obtain a smooth boundary and preserve every feature on the boundary of the surface, the points on the boundary are sampled and approximated evenly. For every boundary point, the closest node M[i,j] to it is searched. A row vector is formed by M[i,j] - M[i±1,j] and the normal of a row plane is the cross product of the row vector and the surface normal at M[i,j]. A column vector is formed by M[i,j] - M[i,j±1], and the normal of a column plane is the cross product of the column vector and the surface normal at M[i,j]. Using the row plane and column plane to cut the boundary and two intersection points representing the extension of weft thread and the warp thread are calculated. Using the two intersection points and M[i,j], a plane can be defined. The boundary point is then projected onto the plane and the position on the plane is transformed to the rectangular coordinates in the corresponding 2D pattern (Fig. 6).

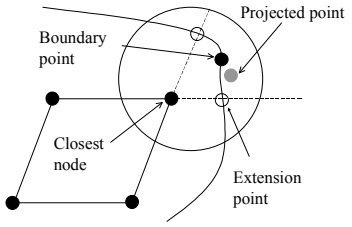


Fig. 6. Mapping of the boundary points

4. RESULTS AND DISCUSSIONS

The prescribed flattening algorithm has been fully implemented on a PC with a modest configuration. Five examples, a partial cone of anisotropic material, an octant-sphere of isotropic material, the same octant-sphere but of anisotropic material, and finally a free-form surface of both materials, are presented next to illustrate the correctness and practicality of the algorithm. A cone is a developable surface, which can be flattened into a plane without any deformation. Thus, the cone example is used to examine the basic ability of our algorithm to flatten a developable surface. The octant-sphere, which is non-developable, is used to test the algorithm on non-developable surfaces, as well as the effect of dart insertions. To compare the effect of the material

properties on the final flattened 2D pattern, both isotropic and anisotropic materials are used for this surface. Finally, the third surface is a piece of clothing in the form of a NURBS surface. In the figures given next, the springs are colored in red, green and blue. The springs with the highest tensile strain energy are colored by pure red and those with the highest compressive strain energy are colored by pure blue. Pure green spring indicates zero energy. The other springs are colored proportionally from the highest value of the tensile down to the highest compressive strain energy. For a clear illustration of the boundary in the 3D model, the boundary nodes are represented by small yellow spheres and the dart nodes are represented by small blue and red spheres. The isotropic material used has a spring constant ratio:

weft : warp : diagonal = 100 : 100 : 100.

The anisotropic material used has a spring constant ratio:

weft : warp : diagonal = 500 : 500 : 1.

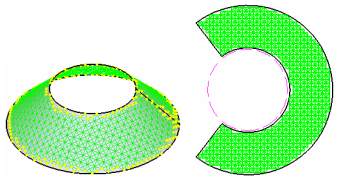
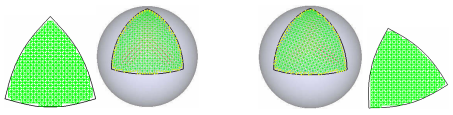


Fig. 7. Mapping of the boundary points

Example I: A conical frustum of anisotropic material

A conical frustum with an anisotropic material is tested. Since this is a developable surface, it should be fitted without any deformation by a 2D pattern. Fig. 7 shows the flattening result. As seen in the figure, the total energy of the spring cone is achieved at a very low level and the energy of the spring is evenly distributed. It is expected that the inner and the outer circumferences of a flattened cone frustum are circles. The boundary of the flattened pattern agrees with the circles (pink), which are added for comparison. There is a small region which has relatively higher strain in a series of diagonal springs. They are expected to be released if the tolerance term in the termination criterion of the energy minimization is lowered.



(a) in orientation 1 (b) in orientation 2

Fig. 8. Optimized isotropic mesh on an octant-sphere in different orientations

Example II: An octant of isotropic material

Conceivably, for isotropic materials, a good flattening algorithm should generate an identical or similar 2D pattern for a surface regardless of whatever the orientation of the spring mesh on the surface. To test our algorithm in this aspect, Fig. 8 shows an optimized isotropic mesh on an octant-sphere in different orientations. The result is as expected; the flattened patterns are identical, only different in orientation.

Example III: An octant of anisotropic material

Fig. 9 shows an optimized anisotropic mesh on an octant-sphere. The strain concentration regions on the surface are shifted compared with that in isotropic mesh (Fig. 8a). Also, unlike the 3D surface which is symmetric with respect to its centroid, the boundary of the corresponding 2D pattern is symmetric to its center line, because of the anisotropic properties of the spring mesh and the symmetric pattern in the initialization stage. If the initial pattern is not symmetric, then the minimization result is the nearest local minimum of the initial mesh and the flattened pattern becomes different. Direction energy is added to control the orientation of the minimized result. Fig. 10 has two examples of constraininh orientation of anisotropic mesh. In each example, there is a column of nodes labeled as small yellow spheres constrained to the direction of the grey double arrow. The flattened patterns are not the same in the case of an isotropic mesh – they are different in shape.

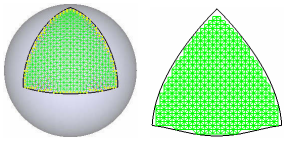
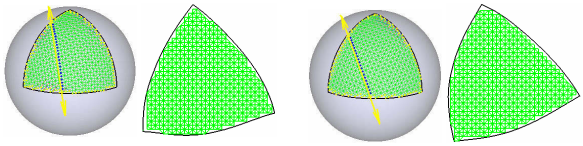


Fig. 9. Optimized anisotropic mesh on a octant-sphere



(a) with prescribed orientation 1 (b) with prescribed orientation 2

Fig. 10. Optimized anisotropic mesh on a octant-sphere with different prescribed orientations

Example IV: Insertion of plan dart and curve dart

Since the octant-sphere is not a developable surface, so deformation is developed in the minimum energy pattern. The location and the direction of buckling can be determined by the color of the springs. To further reduce the strain, a dart can be inserted using the

proposed plane dart and curve dart. A plane dart is inserted onto an octant-sphere (Fig. 11). A sphere is an elliptic surface, so a V-shape result at the dart is expected. Fig. 12 shows the result of inserting a curve dart onto an octant-sphere with isotropic and anisotropic materials. Their flattened patterns are slightly different at the tip of the dart and at the corners. For the anisotropic example, the energy of the springs is evenly spread compared with the example without dart (Fig. 8b). Due to the energy release by the dart insertion, the shape of the anisotropic mesh becomes similar to the shape of the isotropic mesh.

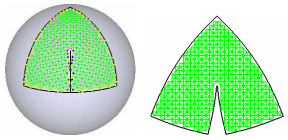
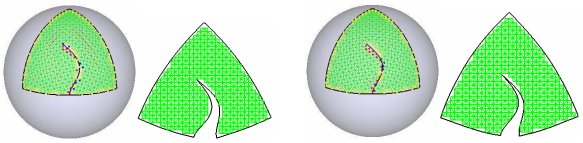


Fig. 11. Optimized isotropic mesh on a octant-sphere with a plane dart

A good flattening algorithm should try to preserve the boundary length and the surface area on the 2D mesh while at the same time minimizes the total energy (deformation). To give a quantitative measure of the optimization results, the boundary error and the area error are used. The data given in Table 1 verify the proposed algorithm in this regard. It is shown that the energy and the errors decrease when a cut is inserted.



(a) isometric mesh (b) anisometric mesh

Fig. 12. Optimized isotropic vs. anisotropic mesh on a octant-sphere with a curve dart

Table 1. Measuring data of fitting results

Example	*EN*	Boundary Error (%)	Area Error (%)
Octant, Isotropic (Fig. 8a)	8.53	5.35	0.37
Octant, Isotropic, with Curve Dart (Fig. 12a)	1.05	2.35	0.10
Octant, Isotropic, with Plane Dart (Fig. 11)	1.65	2.65	0.10
Octant, Anisotropic (Fig. 9)	0.26	5.46	0.42
Octant, Anisotropic, with Curve Dart (Fig. 12b)	0.05	2.59	0.11

Example V: A freeform surface of anisotropic material

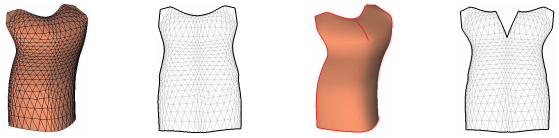
Finally, we apply our surface flattening algorithm to a piece of garment of an anisotropic material modeled as a freeform NURBS surface (Fig. 13a). In practice, a cut is usually made on the pattern near the center line at the collar to enhance the flattening quality. A dart is thus inserted accordingly, and Fig. 13b and 13c show the corresponding optimization result.

Table 2. Computing statistic of the garment patch

Example	Figure	Boundary Error (%)	Area Error (%)
Garment, anisotropic	13a	0.16	2.21
Garment, anisotropic with plane dart	13b	0.24	0.94
Garment, isotropic with plane dart	13c	2.08	0.09

(a) anisotropic, without dart (b) anisotropic, with dart (c) isotropic, with dart

Fig. 13. Garment patch



(a) (b) (c) (d)

Fig. 14. Flattening the garment patch with the planar energy minimization approach [7]

Table 3. The boundary and area error of the planar energy minimization approach [7]

Example	Figure	Boundary Error (%)	Area Error (%)
Garment	14a, 14b	13.52	4.10
Garment with dart	14c, 14d	5.32	3.84

To compare the method presented in this paper and the approach in [7], we tessellate the NURBS surface of the garment patch (as shown in Fig. 14a), and apply the 2D energy minimization approach to determine its related 2D pattern. The result is given in Fig. 14b. After inserting same vertical dart, the flattening result is as shown in Fig. 14d. The boundary errors and area errors are listed in Table 3. Obviously, the approach introduced in this paper gives much more accurate results.

5. CONCLUSION

A robust and practical energy-minimization based surface flattening algorithm is proposed. We model the 3D surface by a spring mesh and take the difference in length between the corresponding spring in 2D and 3D as the energy of that spring. Our algorithm has the following characteristics.

- We assume zero energy on the 2D pattern and directly minimize the energy on the 3D surface itself – this conforms closer to a physical folding process;
- It supports both isotropic and anisotropic materials;
- The minimization is conducted in the parametric domain of the 3D surface;
- Darts insertion is also implemented in our system, directly on the 3D surface, with a plane curve or non-plane curve shape.

For the future development, there is room for the improvements in the energy minimization, so that its speed can fulfill the requirement of real time application.

6. REFERENCES

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

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

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

[4] 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, July 2002, pp.362-71.

[5] 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, 2002, pp.874-90.

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

[7] Wang, C.C.L., Chen, S.S.F., and Yuen, M.M.F., Surface flattening based on energy model, Computer-Aided Design, v34, n11, pp.823-833, 2002.

[8] Azariadis, P.N., and Aspragathos, N.A., Design of plane development of doubly curved surface, Computer Aided Design, v.29 n.10, pp.675-685, 1997.

[9] Azariadis, P.N., and Aspragathos, N.A., Geodesic curvature preservation in surface flattening through constrained global optimization, Computer-Aided Design, V33, n.8, pp.581-591, 2001.

[10] 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.

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

[12] Aono, M., Breen, D.E., and Wozny, M.J., Fitting a woven-cloth model to a curved surface: mapping algorithms, Computer-Aided Design, v.26 n.4, pp.278-292, 1994.

[13] Aono, M., Denti, P., Breen, D.E., and Wozny, M.J., Fitting a woven cloth model to a curved surface: dart insertion, IEEE Computer Graphics & Applications, v.16 n.5, pp.60-70, 1996.

[14] Aono, M., Breen, D.E., and Wozny, M.J., Modeling methods for the design of 3D broadcloth composite parts, Computer-Aided Design, v.33 n.13, pp.989-1007, 2001.

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

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

[17] Wang, C.C.L., Wang, Y., Tang, K., and Yuen, M.M.F., Reduce the stretch in surface flattening by finding cutting paths to the surface boundary, Computer-Aided Design, in press.

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

[19] Belegundu, A.D., and Chandrupatla, T.R., Optimization Concepts and Applications in Engineering, Prentice-Hall, 1999.

[20] Miller, R.E., Optimization Foundations and Application, John Wiley & Sons, 2000.