Available online at www.sciencedirect.com

ScienceDirect

Computer-Aided Design 40 (2008) 381–395

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

WireWarping: A fast surface flattening approach with length-preserved feature curves

Charlie C.L. Wang*

Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

Received 28 April 2007; accepted 25 November 2007

Abstract

This paper presents a novel approach — *WireWarping* for computing a flattened planar piece with length-preserved feature curves from a 3D piecewise linear surface patch. The property of length-preservation on feature curves is very important to industrial applications for controlling the shape and dimension of products fabricated from planar pieces. *WireWarping* simulates warping a given 3D surface patch onto plane with the feature curves as tendon wires to preserve the length of their edges. During warping, the surface-angle variations between edges on wires are minimized so that the shape of a planar piece is similar to its corresponding 3D patch. Two schemes — the progressive warping and the global warping schemes are developed, where the progressive scheme is flexible for local shape control and the global scheme gives good performance on highly distorted patches. Experimental results show that *WireWarping* can successfully flatten surface patches into planar pieces while preserving the length of edges on feature curves.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Surface flattening; Freeform mesh surfaces; Feature curves; Length preserved; Sheet manufacturing industry

1. Introduction

The work presented in this paper is motivated by the development of 3D design automation systems for freeform products in those industries where the products are fabricated from planar pieces of sheet material (e.g. metal in ship industry, fabric in apparel industry and toy industry, leather in shoe industry and furniture industry). How to determine the shape of 2D pieces from designed 3D surface patches now becomes the bottleneck in the design and manufacturing cycle. The cycle is completed only after obtaining the 2D pieces since the final products are fabricated by warping and stitching the 2D pieces. Ideally, a flattened 2D piece is expected to have an isometric mapping to its corresponding 3D patch in the representation of a piecewise linear surface. However, from differential geometry [14], we know that only those developable surfaces satisfy this property. Therefore, the existing approaches in the computer-aided design area for surface flattening (e.g. [1–4,25,43,39,41,45]) and the computer graphics literature for mesh parameterization (e.g. [13,16,18,20,35,36]) adopt various criteria (including the length variation criterion) to minimize the difference between the 3D surface patch and its corresponding 2D region. Moveover, a fast flattening approach is needed in interactive design systems where any 3D shape editing should result in its corresponding 2D pieces instantaneously.

The problem of surface flattening (or parameterization) is usually formulated under a constrained optimization framework, the resultant 3D patch generally is not a developable surface, length variation is always found in the flattening results. For an engineering application such as the 3D garment design and manufacturing shown in Fig. 1, the length variations will lead to many problems. If the length variation occurs on the boundaries of two pieces that are going to be sew together, unexpected wrinkles will form on the fabricated product. If the length variation happens in the interior region of a patch, it will destroy the designed fit (e.g. if the length of chest girth varies on the flattened patterns of the shirt shown in Fig. 1, the products made by these patterns may be too tight or too loose). A good garment shape and fit (i.e. without the unexpected wrinkles) are two necessary criteria to evaluate whether a suit is a high-end garment product. This is also true for other industrial applications (e.g. the shoe industry

* Tel.: +852 26098052; fax: +852 26036002.
E-mail address: cwang@mae.cuhk.edu.hk.

0010-4485/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2007.11.011

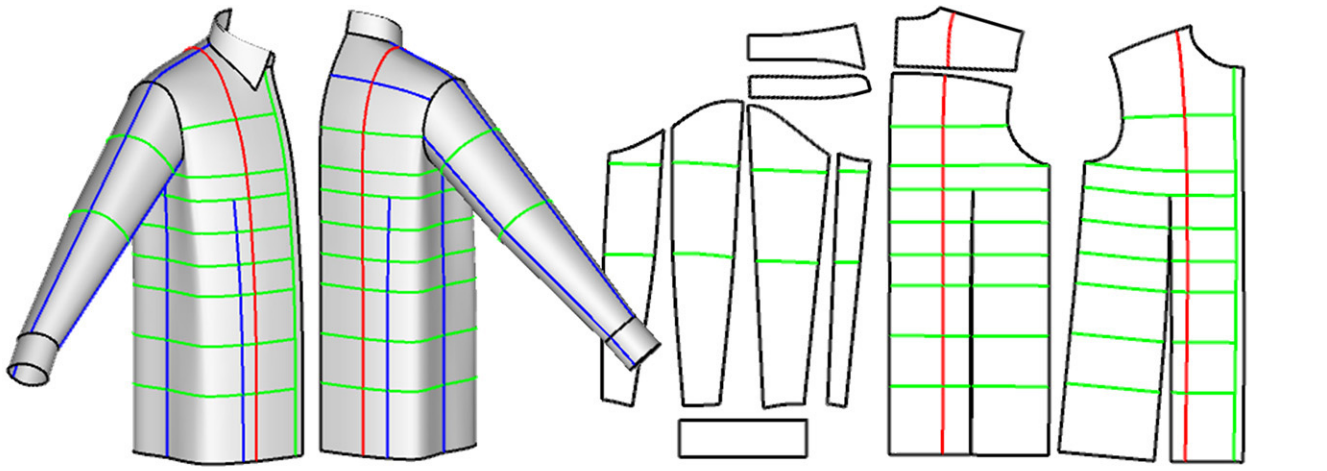

Fig. 1. Example of surface flattening on 3D shirt (front and back views), where the black curves are the boundaries of 3D surface patches, the blue curves are the darts that will be cut out (so that become black curves), the red ones are the key feature curves, and the green ones are the accessory feature curves – the definitions about key feature curves and accessory feature curves will be given in Section 2. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

and the furniture industry). Therefore, the designers in these industries desire a surface flattening tool, which can preserve the length of boundaries and feature curves (e.g. the black and green curves in Fig. 1) on a 2D piece according to its 3D surface patch. In this paper, a method (named as *WireWarping*) is exploited to simulate warping a given 3D surface into 2D with the boundaries and feature curves as tendon wires so that the length of their edges are preserved. The boundaries and feature curves are referred to as *wires* for the rest of this paper. The surface-angle variation between edges on a feature curve during the warping needs to be minimized to make the shape of a pattern in 2D similar to its corresponding 3D patch.

Problem definition. Given a piecewise linear surface patch in $\Re^3$, its counterpart pattern in $\Re^2$ will be computed to make the length of edges on the boundaries and feature curves (named as *wires*) optimally invariant; meanwhile, the angles between neighbouring edges on the wires in $\Re^2$ are optimized to preserve their values on the given 3D surface patch.

1.1. Literature review

Before reviewing the techniques of surface flattening, we cannot avoid reviewing the theoretical background of developable surface from differential geometry. In differential geometry [14], the definition of a developable surface is derived from ruled surfaces: for a ruled surface $X(t, v) = \alpha(t) + v\beta(t)$, it is developable if β, $\dot{\beta}$ and $\dot{\alpha}$ are coplanar for all points on X. In general, a differential surface is developable if and only if it belongs to one of the following surface types: planes, generalized cylinder, conical surfaces (away from the apex), or tangent developable surfaces. Based on this, some researches in literature focused on modelling [19,33,10] or approximating [9,30,32] a model with developable ruled surfaces (or ruled surfaces in other representations — e.g. B-spline or Bézier patches). However, it is difficult to use these approaches to model freeform surfaces. Another limitation of these approaches is that they can only model surface patches with four-sided boundaries as the surfaces are usually defined on a square parametric domain. Although trimmed surfaces were considered in [42], the modelling ability for freeform objects by this category of approaches is still very limited.

In the area of computer-aided design, the surface flattening for pattern design has been studied in various industries (cf. [1–4,25,43,39,41,45]). An ideal surface flattening of a given 3D surface patch P to its corresponding 2D piece D preserves the distances between any two points — mathematically an *isometric mapping* is needed. However, this property is only held on those developable surfaces. Based on this reason, the surface flattening approaches always evaluate the error of distance variations between surface points on P and D, and tries to minimize this error under a non-linear optimization framework. The computation of non-linear optimization in terms of vertex position is very time-consuming and can hardly preserve the invariant length of feature curves. Similarly, the mesh parameterization approaches in literature (cf. [13,16,18, 20,35,36]) give strength on how to minimize the distortions between P and D in angles, areas or lengths. The detail review is given in [15] by Floater and Hormann. Another interesting category of surface flattening approaches solves the problem by computing mappings for dimensionality reduction [6,34,22] or through a multidimensional scaling (MDS) technique [44]. These approaches are all based on computing an optimal mapping that projects the geodesic distances on surfaces into Euclidean distances in $\Re^2$ (i.e. a lower dimension space). Nevertheless, it is difficult to embed the hard constraints on the length of feature curves in the mapping computation.

The idea of preserving the length of feature-curves on a network relates to the isometric tree described in [23] by Manning, where he introduced an isometric tree consisting of a network of curves that are mapped onto the plane isometrically. However, the network considered in [23] is with the tree topology and the isometric curves are the branches of the tree, which are flattened one by one without considering the relationship between these curves. The relationship between feature curves will be fully addressed under a constrained optimization framework in this paper. Another flattening algorithm driven by curves is [8], where Bennis et al. mapped isoparametric curves onto plane followed by a relaxation process to position the surface between them. They also employed a progressive algorithm to process complex surfaces; however, the relationship between these isoparametric curves was not well addressed. Azariadis and Aspragathos [2] proposed a method for optimal geodesic curvature preservation in surface flattening with feature curves. Nevertheless, the drawback of their method is that: their method was based on an optimization in terms of vertex positions, which is highly non-linear and cannot be efficiently solved as the proposed one here in quadratic form. Besides, the length preserved curve mapping does also relate to the intrinsic form of curves discussed in [28].

In literature, some approaches directly model developable (or flattenable) surfaces in $\Re^3$ instead of computing a surface flattening mapping. The authors in [12] processed a given mesh surface by fitting a conical surface locally at every vertex so that the expected normal vectors can be determined. After that, a deformation process is applied to adjust the position of surface vertices to follow the given normal vectors. The resulting local surface is of a conical form. More generally, Wang and Tang in [40] adopted the discrete definition of Gaussian curvature to define the measurement for the developability on given polygonal mesh surfaces. A constrained optimization approach was conducted to deform mesh surfaces to increase their discrete developability. Liu et al. in [21] presented a novel PQ mesh, which can be used to model developable surface in strips. Recently, Wang presented a FL mesh modelling scheme in [37], which models developable mesh surfaces with a more complicated shape. Of course, if a given mesh surface P is developable, the length of feature curves will not be changed during the flattening. However, it is never easy to modify any of these approaches so that they can process a surface from non-developable to developable while preserving the length of feature curves. Besides, the computation in [40,12,21,37] is much slower than the *WireWarping* approach introduced in this paper.

1.2. Contributions and overview

The work presented in this paper has the following technical contributions:

- A novel method *WireWarping* is introduced as a fast surface flattening algorithm which preserves the length of feature curves.
- Two schemes — the *Progressive Warping* and the *Global Warping* are developed, where the former scheme is more flexible for local shape control on feature curves and the latter one gives better performance on those highly curved surface patches.
- The flattening process is benefited from reformulating the constrained optimization problem in the angle space. *WireWarping* is a fast surface flattening approach — in all our tests, the computation can be finished in an interactive speed.

The paper is organized as below. After giving necessary preliminaries in Section 2, the progressive warping scheme of *WireWarping* is presented in Section 3. Section 4 formulates the warping of wires globally under a framework of constrained optimization. Experimental results as well as limitations are discussed in Section 5. Lastly, the paper ends with the conclusion section.

2. Preliminary

We first give necessary definitions and preliminaries.

Definition 1. *Feature curves* are the piecewise linear curves formed by polygonal edges on the given piecewise linear surface P to be flattened, where every segment on a feature curve is required to have the same length on P and the flattened piece D.

Definition 2. For a feature curve, if its planar shape on the flattened piece D has been predefined, it is named as *a key feature curve*; other feature curves are called *accessory feature curves*, whose planar shapes are determined by minimizing the variation between the surface angle and the planar angle at each endpoint of segments.

Definition 3. *Darts* are the curves defined on the piecewise linear surface P, which specify the place to be cut out.

For different products, different sets of feature curves are defined by industrial designers. Feature curves are adopted in the 3D design to control the shape of final product, and the role of these feature curves is like control curves for lofting in geometrical modelling systems. However, for products that are fabricated from 2D sheet materials, an important requirement of the feature curves is that they should be length invariant during flattening (i.e. work like tendon wires). The red and green curves shown in Fig. 1 are feature curves. In all examples of this paper, the key feature curves are coloured in red, the accessory feature curves are displayed in green and the boundary curves are shown in black. The boundary curves are classified into accessory feature curves if no special explanation is given. Darts are illustrated in blue in all examples, which will be converted into boundary curves at the beginning of flattening by iteratively introducing duplicated edges on those belonging to the dart curve. All feature curves are in general called *wires* in the rest of the paper.

Definition 4. Each region circled by feature curves on the given surface P is defined as a *wire-patch*.

For a given surface patch P, it can be segmented into several wire-patches. Fig. 2(a) illustrates the wire-patches on the shirt with different colours. The boundary of a wire-patch is recorded by a list of wire-nodes, where each wire-node is coincident with a vertex on wires of surface patch P. For the vertex v on a feature curve, it may have more than one wire-nodes attached (e.g. the vertex circled by blue curves in Fig. 2(b)). The number of wire-nodes associated with a surface vertex v is determined by the number of wire-patches adjacent to v.

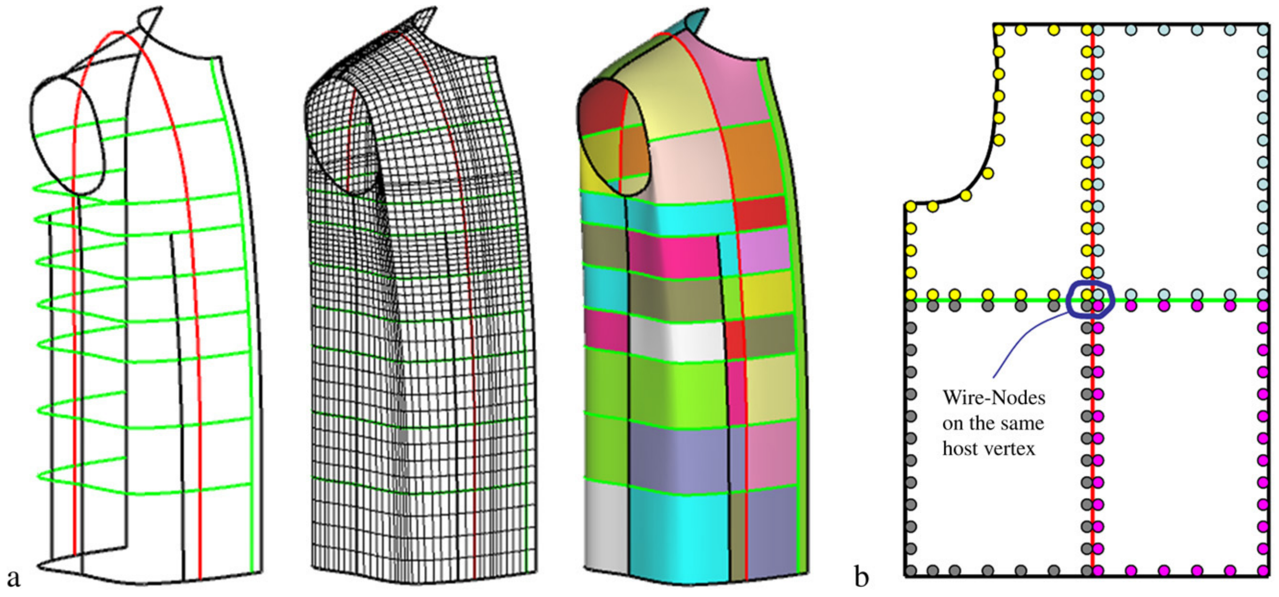

Fig. 2. The preliminary construction of wire-patches. (a) left — the wires (note that the darts have been converted into boundary curves), middle — the given piecewise linear surface, and right — the wire-patches that are visualized in different colors. (b) illustration for the wire-nodes (the wire-nodes belonging to different wire-patches are shown in different colors). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Definition 5. A *wire-node* is denoted by q^i_j with the superscript for the index of the wire-patch P_i holding it, and the subscript represents its index in P_i (ordered anticlockwise); $v(q^i_j)$ represents the vertex holding q^i_j that is named as the *host vertex* of q^i_j.

For three neighbouring wire-nodes q^i_{j-1}, q^i_j and q^i_{j+1} on the same wire-patch, α^i_j is employed to represent the surface angle on P_i formed by them. The value of α^i_j can be evaluated by summing the angles of polygons f_k with $f_k \in P_i$ at the host vertex $v(q^i_j)$. The 2D angle formed by these three wire-nodes after flattening is denoted by θ^i_j – details about how to compute θ^i_j will be addressed later. A data-structure entity is developed for wire-nodes so that we can easily find the host vertex $v(q^i_j)$ of a wire-node q^i_j in a constant time complexity. The wire-nodes in other adjacent wire-patches at $v(q^i_j)$ is also stored in this entity. Also, in order to travel among neighboring wire-patches, the following wire-curve is defined.

Definition 6. A *wire-curve* is defined by an ordered list of directional edges on P which separates two neighbouring wire-patches.

It is obvious that the wire-curves are coincident to feature curves, and the boundary of a wire-patch consists of several wire-curves in general. A wire-patch stores a collection of its wire-curves and a wire-curve entity records its left/right wire-patches in the data structure.

Before discussing details of the *WireWarping* approach, we still need to figure out one more problem — how to specify the shape of key feature curves in $\Re^2$. As will be described later, the shape of wires in $\Re^2$ are determined by the computed planar angles associated with wire-nodes (i.e. the value of θ^i_j at q^i_j). Therefore, users can specify the shape of a key feature curve in $\Re^2$ by assigning θ^i_j of wire-nodes on the key feature curve. Users sometime do not explicitly give the shape of a key feature curve, but just wish that it remains the shape on the given surface P. To reflect the intention of this, we assign the 2D angle of a wire-node q on key feature curves as $\theta(q) \equiv \alpha(q)$ if $v(q)$ is a boundary vertex. If $v(q)$ is an interior vertex on P, the value of $\theta(q)$ is given as

$$\theta(q) \equiv \frac{2\pi \cdot \alpha(q)}{\sum_{q_k \in v(q)} \alpha(q_k)}, \tag{1}$$

where q_ks are the wire-nodes associated with $v(q)$. By Eq. (1)), the 2D angles of wire-nodes associated with an interior vertex are proportional to their values in 3D, and the sum of these 2D angles is restricted to 2π — to be locally flattenable (cf. [37]). Based on the above definitions and preliminaries, two schemes of *WireWarping* will be presented in the following sections.

3. Progressive warping scheme

The progressive warping scheme of *WireWarping* will be presented in this section, which simulates the warping of wire-patches from $\Re^3$ into $\Re^2$ one by one. After addressing the warping problem of a single wire-patch, we consecutively present the progressive warping algorithm, the method for placing feature curves in $\Re^2$, and the method to compute interior meshes of wire-patches.

3.1. Computing length-preserved optimal boundary of a wire-patch

A single wire-patch P_i is in fact a piecewise linear surface patch with the disk-like topology. The method of flattening it

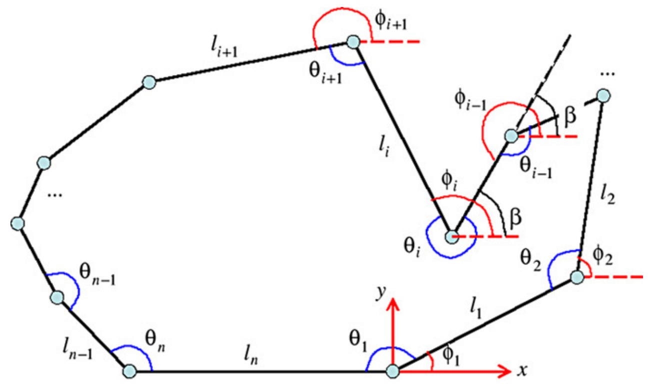

Fig. 3. The computation of length-preserved optimal boundary (from [38]).

into a planar patch retains the length of edges on its boundary (i.e. on the feature curves of P) will be given below. More specifically, we introduce an algorithm that simulates the edges on boundary of wire-patches as tendon wires and flattens the wires onto plane by warping them. During the flattening, we retain the lengths of edges on wires and also minimize the surface-angle variation between edges on wires. Therefore, the shape of a warped wire-patch in 2D is similar to its shape on the given surface patch P.

Based on the above requirements, we can compute the optimal planar boundary of a wire-patch under a constrained optimization framework. The angle variation term is set as the soft constraint in the objective function, and the length invariant term is assigned as the hard constraint. Study in [38] shows that formulating this problem in the angle space as follows can greatly simplify the computation.

$$\min_{\theta_i} \sum_{i=1}^{n} \frac{1}{2}(\theta_i - \alpha_i)^2 \quad \text{s.t. } n\pi - \sum_{i=1}^{n} \theta_i \equiv 2\pi,$$
$$\sum_{i=1}^{n} l_i \cos\phi_i \equiv 0, \quad \sum_{i=1}^{n} l_i \sin\phi_i \equiv 0 \tag{2}$$

where θ_i is the 2D angle associated with the wire-node q_i, α_i represents its 3D surface angle, l_i denotes the length of an edge on the boundary, and n is the number of wire-nodes on the boundary. Since all variables are for the same wire-patch, to be simple the superscript index is neglected. From the *closed-path theorem* (Ref. [26]), we know that: for a simple non-self-intersection planar closed path, if its path is anti-clockwise, the total turning must be 2π. As shown in Fig. 3, the total turning by accumulating vertex turning angles can be computed by $\sum_{i=1}^{n}(\pi - \theta_i)$, which leads to the first constrain in Eq. (2) that $n\pi - \sum_i \theta_i \equiv 2\pi$. The later two constraints in Eq. (2) are derived from the position coincidence requirement. By giving the inner turning angles θ_is and placing the wire-node q_1 at the origin, the planar coordinate (x_i, y_i) of a wire-node q_i becomes $x_i = \sum_{k=1}^{i-1} l_k \cos\phi_k$ and $y_i = \sum_{k=1}^{i-1} l_k \sin\phi_k$. As illustrated in Fig. 3, we have $\theta_i = 2\pi - (\phi_i - \beta)$ at the wire-node q_i and $\beta = \phi_{i-1} - \pi$ at the wire-node q_{i-1}, which yields

$$\phi_i = \pi - \theta_i + \phi_{i-1}. \tag{3}$$

Together with $\phi_1 = \pi - \theta_1$, the general formula for ϕ_i can be derived as $\phi_i = i\pi - \sum_{b=1}^{i} \theta_b$. In order to ensure the boundary of a wire-patch being closed, we must let (x_{n+1}, y_{n+1}) be coincident with the origin, which leads to the last two constraints in Eq. (2).

Algorithm 1 Newton's Method

```
1: while $\|\delta_\theta\|^2/n > 10^{-8}$ do
2:     Solve $\nabla^2 J(X)\delta = -\nabla J(X)$;
3:     $X \leftarrow X + \delta$;
4: end while
```

With the Lagrange multiplier $\lambda = (\lambda_\theta, \lambda_x, \lambda_y)$, the constrained optimization problem defined in Eq. (2) can be converted into an augmented objective function

$$J(X) = \sum_{i=1}^{n} \frac{1}{2}(\theta_i - \alpha_i)^2 + \lambda_\theta \left((n-2)\pi - \sum_{i=1}^{n} \theta_i \right)$$
$$+ \lambda_x \sum_{i=1}^{n} l_i \cos\phi_i + \lambda_y \sum_{i=1}^{n} l_i \sin\phi_i, \tag{4}$$

which can be minimized using Newton's method [29] as shown in Algorithm 1 with $\delta = [\delta_\theta \quad \delta_\lambda]^T$. The size of Hessian matrix $\nabla^2 J(X)$ is $n+3$. To speed up step 2 in the Newton's routine, the sequential linearly constrained programming is used to minimize $J(X)$ by neglecting the terms coming from the second derivatives of the constraints in the Hessian matrix $\nabla^2 J(X)$. The equation

$$\nabla^2 J(X)\delta = -\nabla J(X)$$

solved at each iteration is simplified into

$$\begin{bmatrix} I & \Lambda^T \\ \Lambda & 0 \end{bmatrix} \begin{bmatrix} \delta_\theta \\ \delta_\lambda \end{bmatrix} = \begin{bmatrix} B_\theta \\ B_\lambda \end{bmatrix} \tag{5}$$

with $X = (\theta_1, \ldots, \theta_n, \lambda_\theta, \lambda_x, \lambda_y)$. In this equation, Λ, B_θ and B_λ can all be efficiently evaluated in recursive forms (see [38]). Eq. (5) can then be solved by

$$\Lambda\Lambda^T \delta_\lambda = \Lambda B_\theta - B_\lambda \tag{6}$$
$$\delta_\theta = B_\theta - \Lambda^T \delta_\lambda \tag{7}$$

where $\Lambda\Lambda^T$ is 3×3 (as Λ is $3 \times n$). After computing the optimal values of θ_i, the values of ϕ_i can be determined by Eq. (3). Therefore, the optimal position of every wire-node is given by

$$q_{i+1} = q_i + (l_i \cos\phi_i, l_i \sin\phi_i)^T. \tag{8}$$

In all our tests, Newton's routine reaches the terminal condition in less than 10 iteration steps.

Computation with locked 2D angles. Different from the surface warping in [38], here the computation of the optimal planar boundary for a wire-patch may need to lock the values of 2D angles at some wire-nodes. For example, if the wire-node q_i is located on a key feature curve with its 2D shape specified by designers – an optimal value $\hat{\theta}_i$ at q_i is given. We embed this constraint into the above numerical computation by modifying the linear equation system in Eq. (5). For a wire-node q_i with the specified optimal 2D angle $\hat{\theta}_i$, we firstly let $\theta_i = \hat{\theta}_i$ before starting the Newton's algorithm. The ith column in Λ and the

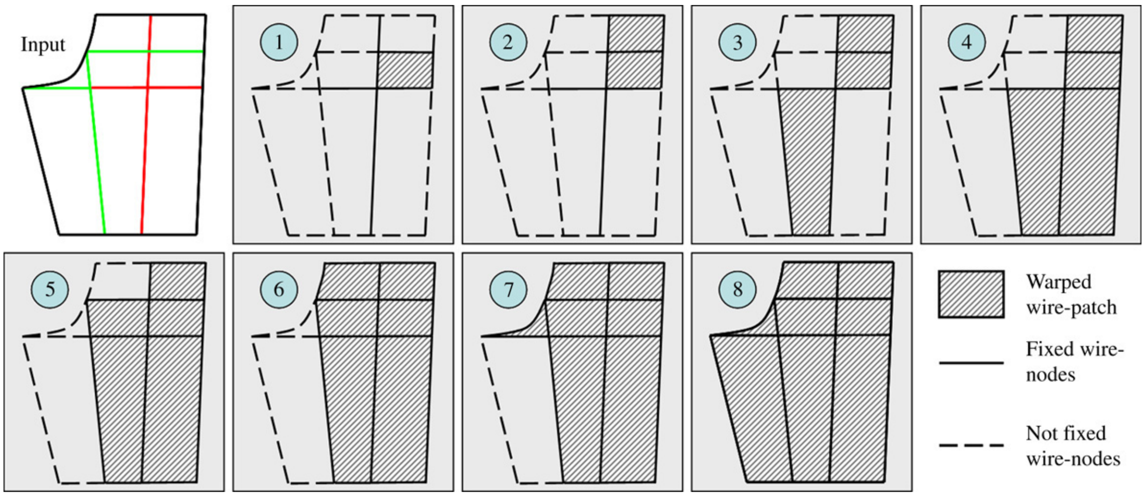

Fig. 4. Illustration for the warping order of wire-patches determined by Algorithm 2. Note that wire-nodes on the key feature curves have been fixed at the beginning of the algorithm.

ith row in Λ^T are then replaced by zeros. Lastly, the ith element in B_θ are assigned to zero. When modifying the linear equation system by this way, $\Lambda\Lambda^T$ in Eq. (6) may become singular if too many 2D angles are locked. Therefore, to be numerically stable, we conduct the *Singular Value Decomposition* (SVD) [31] to solve Eq. (6).

Besides the wire-nodes on key feature curves, there is also another reason that the 2D angle on wire-node q needs to be locked. Let $\hat{Q}(q) = \{q_k \in v(q)\} \setminus \{q\}$ represents the set of other wire-nodes associated with the host vertex $v(q)$ except q, and Γ is defined as the set of all wire-patches holding the wire-nodes in $\hat{Q}(q)$. If the host vertex $v(q)$ of q is an *interior* vertex on the given surface patch, when all wire-patches in Γ have been warped into plane (i.e. all nodes in $\hat{Q}(q)$ have their 2D angles determined), we need to lock the 2D angle of q as

$$\theta(q) = 2\pi - \sum_{q_k \in \hat{Q}(q)} \theta(q_k). \tag{9}$$

Otherwise, the flattened wire-patches around $v(q)$ will not be compatible to each other.

3.2. *Progressive warping algorithm*

The basic idea of the progressive warping algorithm is to warp the wire-patches into plane progressively by using the angle-based method presented above. The strategy of our progressive warping is somewhat similar to [8]. After computing the optimal 2D angles of wire-nodes on all wire-patches, we can iteratively determine the position of feature curves (i.e. the host vertices of wire-nodes) in $\Re^2$ followed by placing the interior mesh of each wire-patch. To flatten wire-patches progressively, we need to determine the warping order. Starting from positioning the wire-patches adjacent to key feature curves, we wish the first warp is always the wire-patch having more 2D angles locked wire-nodes. Therefore, we defined the *shape-evidence factor* on each wire-patch P_i as

$$\varrho(P_i) = n'/n \tag{10}$$

where n' is the number of wire-nodes with their 2D angles known and n is the number of all wire-nodes on P_i. We employ a maximum heap keyed by the shape-evidence factor to determine the warping order of wire-patches. Pseudo-code of the progressive warping algorithm is shown in Algorithm 2. Fig. 4 shows an illustration for the warping order of wire-patches determined by this algorithm.

Algorithm 2 Progressive Warping

```
1:  Construct wire-patches by the feature curves on a given surface patch $P$;
2:  Initialize the shape-evidence factors of all wire-patches;
3:  Insert all wire-patches into a maximum heap $\Upsilon$ keyed by the shape-evidence factors;
4:  **while** $\Upsilon$ is not empty **do**
5:    Remove the top wire-patch $P_t$ from $\Upsilon$;
6:    Warp the boundary wires of $P_t$ into $\Re^2$ (by the method in Section 3.1);
7:    **for** the host vertex $v(q_b)$ of every wire-node $q_b \in P_t$ **do**
8:      **if** $v(q_b)$ is an interior vertex **AND** all wire-nodes in $v(q_b)$ but $q$ ($q \neq q_b$) have their 2D angle determined **then**
9:        Assign the value of $\theta(q)$ by the method of Eq. (9);
10:     **end if**
11:   **end for**
12:   Update $\varrho(\ldots)$ of all wire-patches neighbouring to $P_t$ (by Eq. (10)) and thus their positions in $\Upsilon$;
13: **end while**
14: Lay out the feature curves in $\Re^2$;
15: Computing interior meshes of each wire-patch.
```

3.3. *Laying out feature curves and interior mesh vertices of wire-patches*

After computing the optimal 2D angles on all wire-nodes, we need to lay out the wire-patches and their interior mesh vertices in $\Re^2$. The feature curves are placed first, and the

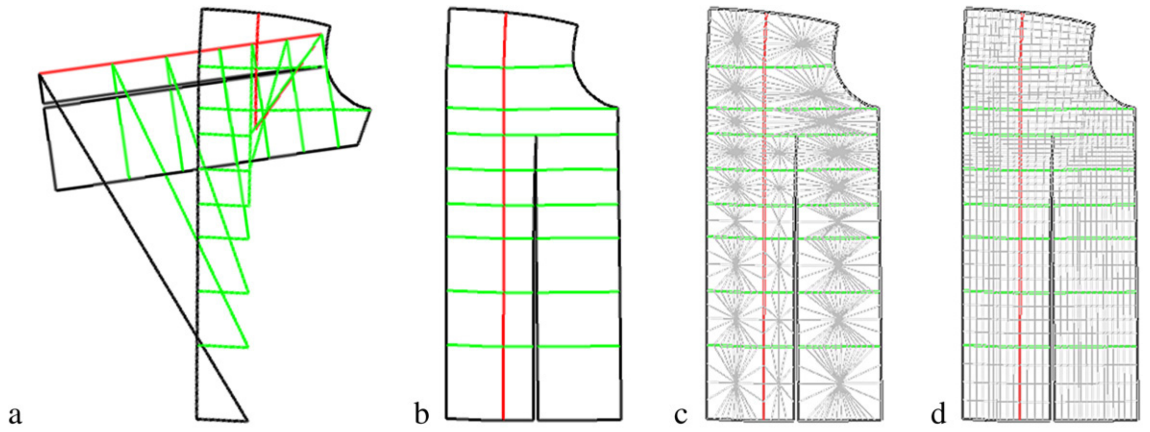

Fig. 5. Laying out feature curves and computing interior meshes of wire-patches: (a) a failure example for placing wire-patches in $\Re^2$ with an incorrect order, (b) a successful positioning with the new order of wire-patches, (c) the interior mesh nodes of every wire-patch are first placed at its centre, and (d) the final mesh.

positions of interior mesh vertices are computed with the fixed boundary of wire-patches.

For placing feature curves, we need to reorder wire-patches by propagation and to store them in a list Ψ. Starting from a seed wire-patch P_s, after inserting P_s into Ψ we check all neighbouring wire-patches of P_s. If any neighbor P_r not in Ψ is found, we insert P_r into Ψ and check the neighbours of P_r recursively. With the wire-patches ordered in Ψ, we can place the host vertices of wire-nodes patch by patch. The wire-nodes in a wire-patch are classified into two types: *fixed nodes* — those with their host vertices' 2D positions known, and *free nodes* — those whose host vertices have not been placed yet. For the wire-nodes in a wire-patch P_i, we search them anticlockwise and find the first free node q_e which is next to a fixed node q_f. The position of q_e's host vertex $v(q_e)$ can be determined by Eq. (8) with the positions of $v(q_f)$ and its previous fixed node $v(q_{f-})$, the edge length of $v(q_f)v(q_e)$ in $\Re^3$, and the optimal 2D angle $\theta(q_f)$ at q_f. Similarly, the latter free nodes on this wire-patch P_i can be placed consecutively. No fixed wire-node can be found on the first wire-patch in Ψ. We thus randomly choose two neighbouring wire-nodes and fix them in $\Re^2$ by reserving the distance between them. In this way, we can lay out all feature curves (i.e. the boundaries of all wire-patches).

The reason, why we do not employ the order of wire-patches determined in Algorithm 2, is because this could lead to a wire-patch being placed incorrectly. For instance in Fig. 4, after placing the second wire-patch, the third wire-patch has only one fixed wire-node that leads to an unsuccessful placement of the next free wire-node. Fig. 5(a) shows an example of such failure.

The mesh vertices not associated with any wire-node (i.e. the interior mesh vertices of wire-patches) finally need to be positioned in $\Re^2$ to generate a correct mesh surface representation. Every vertex v_i is first placed at the average position of the boundary vertices of the wire-patch holding it (e.g. Fig. 5(c)). Next, the positions of v_i are moved iteratively by the operator

$$v_i^{\text{new}} \leftarrow \frac{1}{w(v_i)} \sum_{j \in N(v_i)} \|v_i v_j\|^{-1} v_j, \tag{11}$$

where $\|\cdots\|$ denotes the distance of two vertices on the given surface, $N(v_i)$ represents the 1-ring neighbours of the vertex v_i, and $w(v_i)$ is the summed weights as

$$w(v_i) = \sum_{j \in N(v_i)} \|v_i v_j\|^{-1}.$$

This is in fact the iterative version to solve a Laplacian-like system [27], which has been proved to be very stable. The iteration stops when the movement of all vertices are less than 10^{-5}, and the number of iterations is in the range between 10 to 100. To further speed up the computation, we introduce a relaxing factor $\tau = 1.5$ (like the improvement of convergency using relaxation for the Gauss-Seidel solver in [7]) to let

$$v_i^{\text{new}} \leftarrow v_i + \tau \left(\left(\frac{1}{w(v_i)} \sum_{j \in N(v_i)} \|v_i v_j\|^{-1} v_j \right) - v_i \right), \tag{12}$$

so that the number of iteration steps can be reduced by about two-thirds in most examples. An example result has been shown in Fig. 5(d).

4. Global warping scheme

The progressive warping scheme works well on those surface patches that are nearly developable; however, the industrial application sometimes wishes to flatten highly curved surfaces that are far from developable (e.g. the wetsuit shown in Figs. 9 and 10) where the progressive warping scheme meets great difficulty in giving a satisfactory result. The major reason is that great distortion will occur on a given warped wire-patch if the surface is far from developable. The progressive warping scheme will accumulate the distortions on the warped wire-patches from the very beginning to the last warped one (i.e. the wire-patch that is lastly popped from the heap Υ). Therefore, the last one has the greatest distortion. The order of warping wire-patches can be adjusted by defining different sets of key feature curves. However, the accumulated distortion error of all wire-patches has never been optimized in this way. Therefore, the global warping scheme is developed for highly curved surfaces.

4.1. Formulation

To compute a flattening of wire-patches by the means of global warping, we integrated the subsystems of constrained optimization (i.e. the ones presented in Eq. (2)) into a consistent system to warp all wire-patches together. In addition to the closed path constraint and the position coincident constraints, the compatibility constraint is introduced so that the sum of 2D angles of the wire-nodes associated with an interior host vertex v is 2π.

Without loss of generality, if there are total m wire-patches constructed on the given surface P, we have $\sum_{p=1}^{m} n_p$ wire-nodes where n_p represents the number of wire-nodes for the wire-patch P_p whose index is p. As all wire-patches will be warped together, every wire-node will have one local index in the wire-patch and another global index. To simplify the expression, we define a permutation function $\Gamma_p(b)$ for returning the global index of a wire-node on the wire patch P_p with the local index b, and its inverse function $\Gamma_p^{-1}(j)$ that gives the local index of a wire-node q_j on the wire-patch P_p. The goal of global warping is to find constrained optimal 2D angles for wire-nodes so that the global distortion of flattening is minimized, which can be formulated as follows:

$$
\begin{aligned}
&\min_{\theta_i} \sum_i \frac{1}{2}(\theta_i - \alpha_i)^2 && \\
s.t.\quad & n_p\pi - \sum_{b=1}^{n_p} \theta_{\Gamma_p(b)} \equiv 2\pi && (\forall p = 1,\dots,m) \\
& \sum_{b=1}^{n_p} l_b \cos\phi_b \equiv 0, \quad \sum_{b=1}^{n_p} l_b \sin\phi_b \equiv 0 && (\forall p = 1,\dots,m) \\
& \sum_{q_k \in v} \theta_k \equiv 2\pi && (\forall v \in \Phi)
\end{aligned}
\tag{13}
$$

where Φ represents the collection of interior vertices on accessory feature curves, θ_i is the 2D angle associated with the wire-node q_i, α_i represents its 3D surface angle, and l_b denotes the length of an edge on wires. In summary, if there are l wire-nodes with their 2D angles locked by the key feature curves, the number of variables for the above problem is

$$n_{\text{var}} = \left(\sum_{p=1}^{m} n_p\right) - l,$$

and the set of free wire-nodes is defined as Q_{act}. If there are r interior vertices on the accessory feature curves, the total number of constraints is

$$n_{\text{con}} = 3m + r.$$

Using the Lagrange multipliers [29], the constrained optimization in Eq. (13) can be converted into an augmented objective function that

$$
\begin{aligned}
J = &\sum_{i \in Q_{\text{act}}} \frac{1}{2}(\theta_i - \alpha_i)^2 \\
&+ \sum_p \left[\lambda_{\theta_p}\left((n_p - 2)\pi - \sum_{b=1}^{n_p} \theta_{\Gamma_p(b)}\right) \right. \\
&\left. + \lambda_{x_p}\left(\sum_{b=1}^{n_p} l_b \cos\phi_b\right) + \lambda_{y_p}\left(\sum_{b=1}^{n_p} l_b \sin\phi_b\right) \right] \\
&+ \sum_{v \in \Phi} \lambda_v \left(2\pi - \sum_{q_k \in v} \theta_k\right).
\end{aligned}
\tag{14}
$$

The objective function is again minimized by Newton's method (i.e. Algorithm 1) with the sequential linearly constrained programming, where in each iteration the following linear equation system (derived from $\nabla^2 J\delta = -\nabla J$) is solved.

$$
\begin{bmatrix} I & \Lambda^T & D^T \\ \Lambda & & \\ D & & \end{bmatrix}
\begin{bmatrix} \delta_\theta \\ \delta_\lambda \\ \delta_v \end{bmatrix}
=
\begin{bmatrix} B_\theta \\ B_\lambda \\ B_v \end{bmatrix}.
\tag{15}
$$

The dimensions of Λ and D are $3m \times n_{\text{var}}$ and $r \times n_{\text{var}}$ respectively, where

$$
D = \{d_{v,j}\} = \left\{ \frac{\partial^2 J}{\partial \lambda_v \partial \theta_j} \right\} = \left\{ \frac{\partial}{\partial \theta_j}\left(2\pi - \sum_{q_k \in v} \theta_k\right) \right\}
= \begin{cases} -1 & (q_j \in v) \\ 0 & (\text{otherwise}), \end{cases}
\tag{16}
$$

and

$$
\Lambda = \{\Lambda_{p,j}\} = \left\{ \frac{\partial^2 J}{\partial \lambda_{\theta xy} \partial \theta_j} \right\}
= \begin{bmatrix}
\frac{\partial}{\partial \theta_j}\left((n_p - 2)\pi - \sum_{b=1}^{n_p} \theta_{\Gamma_p(b)}\right) \\
\frac{\partial}{\partial \theta_j} \sum_{b=1}^{n_p} l_b \cos\phi_b \\
\frac{\partial}{\partial \theta_j} \sum_{b=1}^{n_p} l_b \sin\phi_b
\end{bmatrix}.
\tag{17}
$$

$\Lambda_{p,j} = 0$ if the wire-node q_j with θ_j is not on the wire-patch P_p, and

$$
\Lambda_{p,j} = \left(-1, \sum_{b=\Gamma_p^{-1}(j)}^{n_p} \sin\phi_b, -\sum_{b=\Gamma_p^{-1}(j)}^{n_p} \cos\phi_b\right)^T
$$

when $q_j \in P_p$. In order to evaluate $\Lambda_{p,j}$ efficiently, we can conduct the following recursive formulas.

$$\Lambda_{p,\Gamma_p(n_p)} = (-1, l_{n_p} \sin\phi_{n_p}, -l_{n_p} \cos\phi_{n_p})^T \tag{18}$$

$$\Lambda_{p,\Gamma_p(j)} = \Lambda_{p,\Gamma_p(j+1)} + (0, l_j \sin\phi_j, -l_j \cos\phi_j)^T. \tag{19}$$

For the right-hand side vector, since $\frac{\partial \phi_k}{\partial \theta_b} = \begin{cases} 0 & (k < b) \\ -1 & (k \geq b) \end{cases}$, we have

$$
B_\theta = \left\{ -\frac{\partial J}{\partial \theta_i} \right\} = (\alpha_i - \theta_i) + \lambda_{\theta_p}
- \sum_{b=\Gamma_p^{-1}(i)}^{n_p} l_b(\lambda_{x_p} \sin\phi_b - \lambda_{y_p} \cos\phi_b) + \Pi(i)
\tag{20}
$$

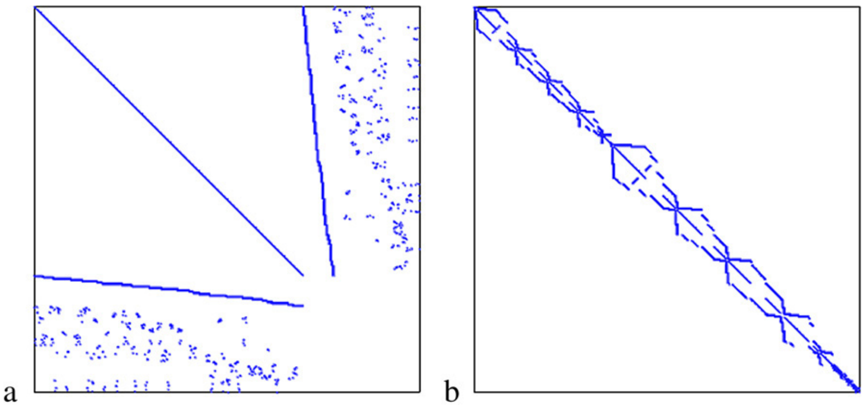

Fig. 6. Patterns of the linear equation system in Eq. (15): (a) the pattern without optimization, and (b) the envelope minimized pattern by the *Cuthill-McKee* Algorithm [11].

with $\Pi(i) = \begin{cases} \lambda_v & (v(q_i) \in \Phi) \\ 0 & \text{(otherwise)}, \end{cases}$

$$B_\lambda = \left\{ -\frac{\partial J}{\partial \lambda_{\theta xy_p}} \right\} = \begin{bmatrix} \left(\sum_{b=1}^{n_p} \theta_{\Gamma_p(b)} \right) - (n_p - 2)\pi \\ -\sum_{b=1}^{n_p} l_b \cos \phi_b \\ -\sum_{b=1}^{n_p} l_b \sin \phi_b \end{bmatrix}, \tag{21}$$

and

$$B_v = \left\{ -\frac{\partial J}{\partial \lambda_v} \right\} = \left\{ \left(\sum_{q_k \in v} \theta_k \right) - 2\pi \right\}. \tag{22}$$

All of these can be evaluated very efficiently.

We start the computation by letting $\theta_i = \alpha_i$, and the Newton's algorithm always stops in less than 10 iterations. After determining the optimal θ_is, the planar coordinates of vertices on feature curves can be computed by the method presented in Section 3.3. Then, the interior mesh vertices can be placed in $\Re^2$ in the same way. Note that when using Eq. (3) to compute the value of ϕ_i from θ_is, the 2D angles of locked wire-nodes should be included since they also contribute to the shape of every wire-patch. Nevertheless, they do not show up in the linear system of Eq. (15) since it only depends on the differentiation with variables (i.e. the angles on free wire-nodes).

4.2. Analysis of numerical computations

The linear equation system above can be solved only if $n_{\text{con}} \leq n_{\text{var}}$; otherwise, it becomes singular. However, we never met the case with $n_{\text{con}} > n_{\text{var}}$ in practice. Even if this occurs, we can still employ the *Singular Value Decomposition* (SVD) [31] to find a reasonable approximation of its solution. In the following discussion, we only concentrate on the cases that $n_{\text{con}} \leq n_{\text{var}}$. Letting

$$H = \begin{bmatrix} \Lambda \\ D \end{bmatrix}, \quad \delta_h = \begin{bmatrix} \delta_\lambda \\ \delta_v \end{bmatrix}, \quad B_h = \begin{bmatrix} B_\lambda \\ B_v \end{bmatrix},$$

we have (from Eq. (15))

$$\begin{bmatrix} I & H^T \\ H & 0 \end{bmatrix} \begin{bmatrix} \delta_\theta \\ \delta_h \end{bmatrix} = \begin{bmatrix} B_\theta \\ B_h \end{bmatrix},$$

which leads to a linear equation system with smaller size and a followed substitution as

$$HH^T \delta_h = HB_\theta - B_h, \tag{23}$$

$$\delta_\theta = B_\theta - H^T \delta_h. \tag{24}$$

The dimension of HH^T is $(3m + r) \times (3m + r)$. Eq. (23) can only be efficiently solved by *Gaussian Elimination* [31] when $(3m + r) < 100$, which cannot always be satisfied.

After studying the pattern of the linear equation system in Eq. (15), we find that it is in general sparse (e.g. the one shown in Fig. 6(a)) and symmetric (see Eq. (15)). When using the famous *Cuthill-McKee* Algorithm [11] to reorder the elements, it can be converted into a matrix with narrower band (see Fig. 6(b)). In our tests, the semi-band-width falls in the range between 10 to more than 100 – in other words, using the optimized narrow-band matrix [31] can speed up the computation somewhat but not too much. We then move to the sparse direct solver – SuperLU (cf. [17]) based on a sparse LU factorization, by which all examples shown in this paper can be finished in less than 100 microseconds per iteration on a PC with standard configuration. The computational complexity is approximately linear to $3m + r + n_{\text{var}}$.

5. Experimental results and discussion

Several examples are tested in this section to demonstrate the performance of our *WireWarping* approach. Our first example is a 3D shirt which is generated from the design automation system of 3D garment. Fig. 1 shows the given 3D surfaces with darts, key feature curves and accessory feature curves. The flattened 2D pieces shown in Fig. 1 are generated by the global warping scheme of *WireWarping* — the lengths of edges on feature curves are strongly preserved during the flattening. Fig. 7(a) shows the colour map for illustrating the developability on the main-body surfaces of the 3D shirt, where

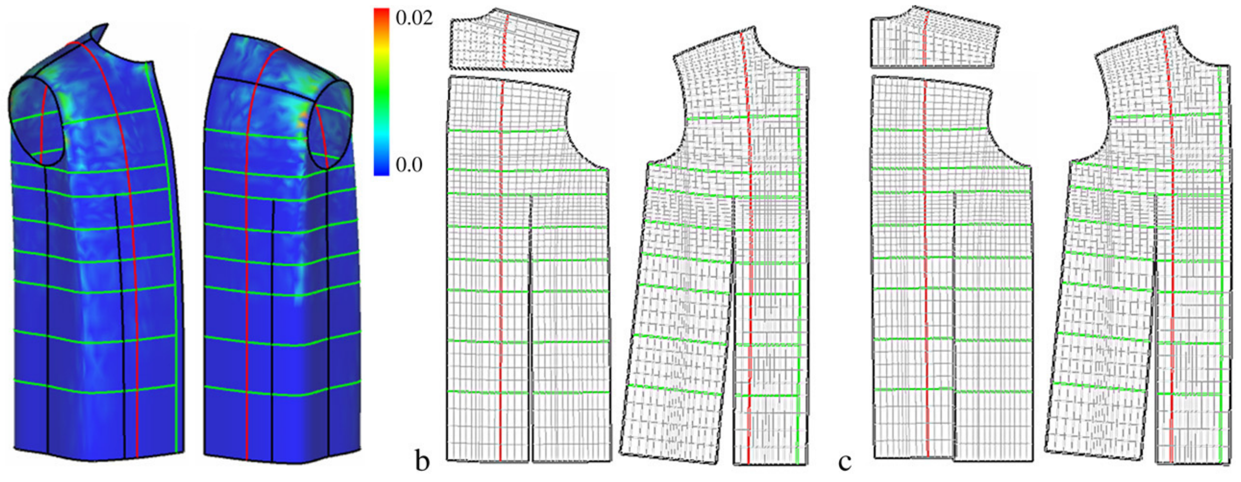

Fig. 7. Examples for flattening the main-body of 3D shirt: (a) the colour map shows the local developability on the given surface, (b) the flattening results from the progressive warping scheme of *WireWarping*, and (c) the results from the *Least Squares Conformal Map* (LSCM) [20]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

the local developability at an iterative vertex v is measured by

$$E_{\text{dev}}(v) = |\varphi(v) - 2\pi| \tag{25}$$

with $\varphi(v)$ denoting the sum of angles at v for the polygons adjacent to v. When $E_{\text{dev}}(v) \equiv 0$, v is locally developable; otherwise, the greater value of $E_{\text{dev}}(v)$, the higher stretching is given at the surface around v during flattening. It is easy to find that there are many non-developable regions on the 3D shirt, therefore flattening results of the 3D shirt should contain distortion in some sense. Fig. 7(b) gives the results from the progressive warping scheme. For comparing our results with the state-of-the-art approach existing in the literature, we choose the *Least Squares Conformal Map* (LSCM) [20] which is the core algorithm of the texture mapping function in Maya [24]. Note that as the method of LSCM does not preserve the scale of a flattening, we add a post-processing step to scale the flattening result so that it has the same perimeter as the given surface in $\Re^3$. The results from LSCM are shown in Fig. 7(c). It is obvious that the boundaries at two-sides of the dart on the back piece in Fig. 7(c) have different lengths, which will yield annoying wrinkles when sewing a shirt from this 2D pattern. The following two error terms are computed in our tests for measuring the results of flattening.

Edge-length error. The length variation of edges on the feature curves (wires) is measured by

$$E_{\text{len}} = \frac{1}{N(\Omega_e)} \sum_{e \in \Omega_e} \frac{|l_e^0 - l_e|}{l_e^0}, \tag{26}$$

where Ω_e is the set of edges on all feature curves, $N(\ldots)$ defines the number of elements in a set, l_e^0 is the length of the edge e in $\Re^3$, and l_e is its length in $\Re^2$.

Angle error. The angle variation of all polygons on the given piecewise linear surface is measured as

$$E_{\text{ang}} = \frac{1}{N(\Omega_a)} \sum_{a \in \Omega_a} \frac{|\vartheta_a^0 - \vartheta_a|}{\vartheta_a^0} \tag{27}$$

with Ω_a as the collection of all polygonal angles on the given mesh surface P, and ϑ_a^0 and ϑ_a are the values of the polygonal angle a in $\Re^3$ and $\Re^2$ respectively.

For an ideal flattening result, it should let both E_{len} and E_{ang} be zero. However, these two error terms in general are not compatible to each other on a non-developable surface. The results from our *WireWarping* preserve $E_{\text{len}} \equiv 0$ and try to minimize the value of E_{ang}.

Homogeneity of distortion and **aspect ratio.** The global homogeneity of distortion E_h and the global aspect ratio E_r, which were proposed by Azariadis and Sapidis in [5], are also measured in our tests. The ideal values of both E_h and E_r are *one*, which is only shown on an isometric mapping. Details of their calculation can be found in [5] and are neglected here. The colour maps for displaying the homogeneity of distortion and the aspect ratio on faces are also shown in some results below.

Our second test is to flatten a pair of 3D pants by using different sets of key-feature curves (see Fig. 8). Obviously, when giving different sets of key feature curves, we can get different flattening results. The progressive warping scheme is easier to be affected by setting different key feature curves. In other words, it is more flexible so that we can have the ability to control the local shape of flattened pieces. The third and fourth examples are a piece of wetsuit with highly curved non-developable surfaces (see Figs. 9 and 10). Computational statistics of all the examples are listed in Table 1. From the statistics, we can conclude that: both the progressive warping scheme and the global warping scheme of *WireWarping* can compute a flattened patch while strongly preserving the length of edges on feature curves. The global warping scheme gives less angle distortion since it actually distributes the distortion error to all wire-patches but the progressive warping scheme accumulates the error. Also, both schemes of *WireWarping* can be finished in an interactive speed. The results from LSCM always give great variation of lengths on the boundaries to be sewed and the feature curves, which can be found from both the resultant shape and the computational statistics. The

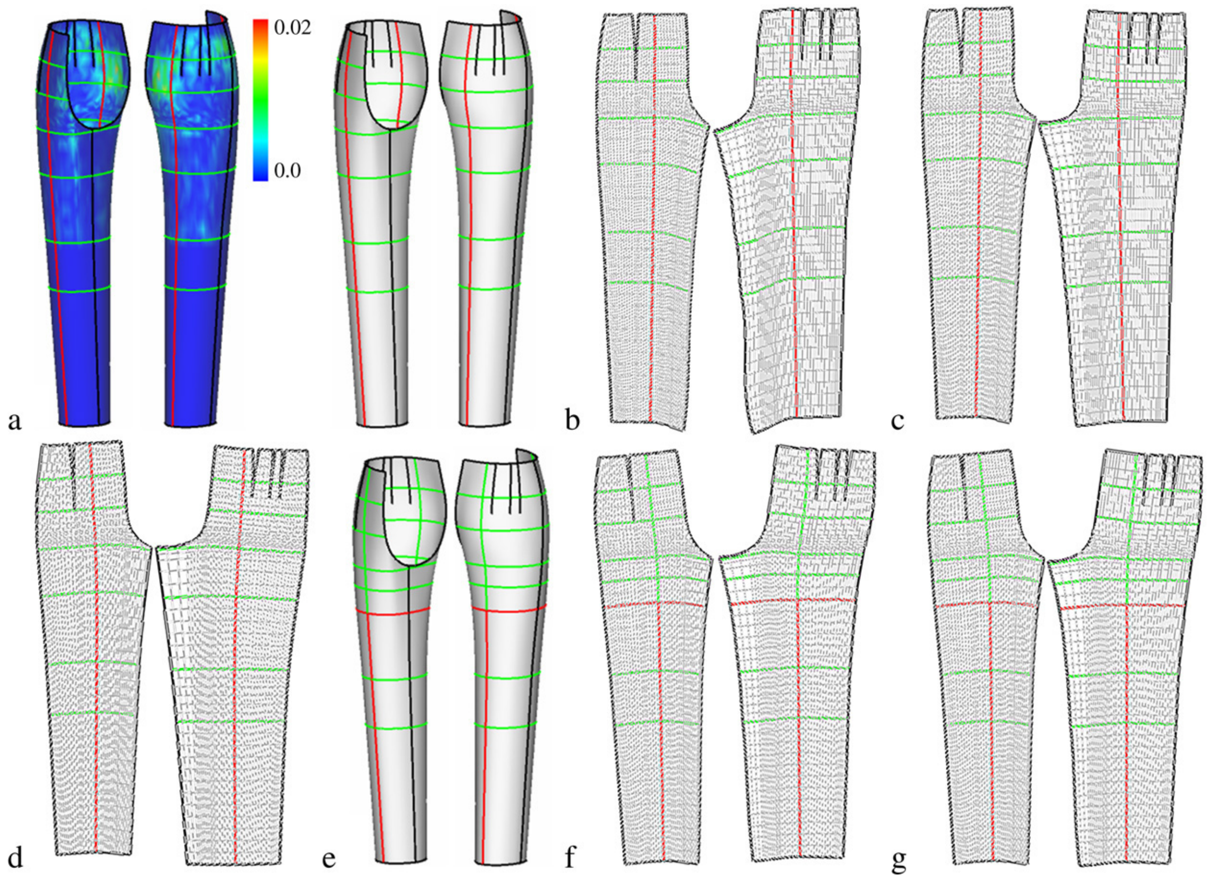

Fig. 8. Example for flattening 3D pants: (a) the given 3D surface of pants and the colour map for showing the developability — the front and back grainlines are defined as key feature curves, (b) the results from the progressive warping scheme, (c) the results from the global warping scheme, (d) the results from LSCM [20], (e) setting mid-thigh and the grainlines below as key feature curves, (f) the results from the progressive warping scheme, and (g) the results from the global warping scheme.

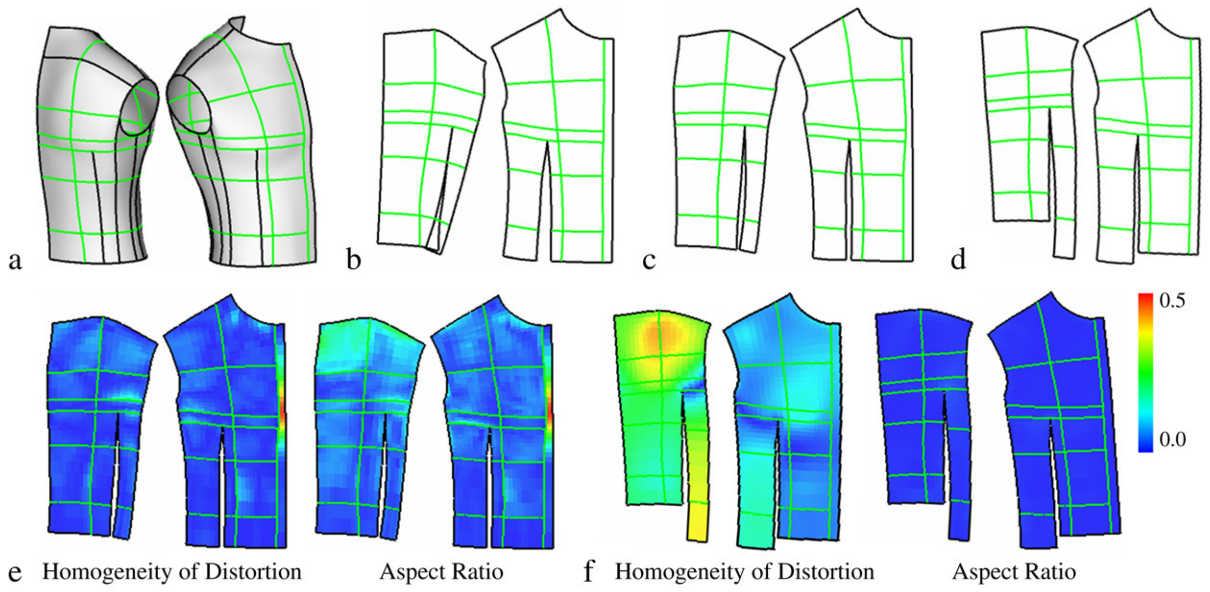

Fig. 9. Example for flattening wetsuit — the upper body: (a) given surface with darts and feature curves, (b) the results from the progressive warping scheme, (c) the results from the global warping scheme, (d) the results of LSCM [20] that the boundaries to be sewed are with different lengths, and the colour map for showing the *Homogeneity of Distortion* and the *Aspect Ratio* [5] on the results from (e) the global warping scheme versus (f) LSCM [20]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Computational statistics

Example	Piece	Method	Fig.	E_{len} ($\times10^{-3}$)	E_{ang} ($\times10^{-3}$)	E_h	E_r	T_{opt}	T_{all}
Shirt	Front (1414)	Progressive	7(b)	0.00	7.72	0.725	0.721	31	47
		Global	1	0.00	6.71	0.727	0.731	109	110
		LSCM	7(c)	5.36	0.486	0.952	0.970	Nil	46
	Back (1234)	Progressive	7(b)	0.00	10.4	0.922	0.921	46	47
		Global	1	0.00	4.64	0.922	0.931	125	125
		LSCM	7(c)	8.85	0.392	0.913	0.955	Nil	63
	Shoulder (182)	Progressive	7(b)	0.00	21.5	0.821	0.830	~0	~0
		Global	1	0.00	21.5	0.821	0.830	~0	~0
		LSCM	7(c)	4.29	1.41	0.938	0.965	Nil	16
Pants (with set 1)	Front (1729)	Progressive	8(b)	0.00	41.8	0.775	0.717	31	78
		Global	8(c)	0.00	23.1	0.832	0.703	93	140
		LSCM	8(d)	8.71	0.737	0.852	0.911	Nil	125
	Back (1729)	Progressive	8(b)	0.00	72.4	0.716	0.649	47	63
		Global	8(c)	0.00	30.4	0.671	0.626	94	125
		LSCM	8(d)	42.5	1.56	0.701	0.835	Nil	125
Pants (with set 2)	Front (1729)	Progressive	8(f)	0.00	15.2	0.796	0.745	31	63
		Global	8(g)	0.00	13.8	0.801	0.730	109	156
	Back (1729)	Progressive	8(f)	0.00	38.1	0.794	0.746	31	63
		Global	8(g)	0.00	36.3	0.786	0.711	110	125
Wetsuit upper body	Front (658)	Progressive	9(b)	0.00	41.1	0.912	0.951	16	32
		Global	9(c)	0.00	35.2	0.915	0.948	63	94
		LSCM	9(d)	113.	2.82	0.985	0.992	Nil	31
	Back (928)	Progressive	9(b)	0.00	22.1	0.792	0.892	15	31
		Global	9(c)	0.00	19.9	0.799	0.875	31	78
		LSCM	9(d)	39.4	1.77	0.960	0.980	Nil	31
Wetsuit pants	Front (1729)	Progressive	10(b)	0.00	65.4	0.446	0.443	32	94
		Global	10(c)	0.00	40.0	0.464	0.452	140	187
		LSCM	10(d)	47.9	4.44	0.587	0.736	Nil	109
	Back (1729)	Progressive	10(b)	0.00	69.8	0.398	0.377	31	62
		Global	10(c)	0.00	65.7	0.343	0.335	125	156
		LSCM	10(d)	81.7	5.17	0.456	0.664	Nil	109
Quarter-sphere	(56)	Progressive	11(b)	0.00	159.1	0.394	0.511	~0	~0
		Global	11(c)	0.00	148.8	0.404	0.518	~0	~0
		LSCM	11(d)	64.3	42.1	0.582	0.672	Nil	~0

The numbers listed in the brackets are the number of polygons on the piece. All the examples are tested with our implementation on a PC with PIV 3.0 GHz CPU + 1GB RAM. The time is measured in microseconds. The numbers in the T_{opt} column are the time for WireWarping without placing the interior vertices, and the numbers in the T_{all} column are the overall time for computation (i.e. including the time to place interior mesh vertices).

measurements on the global homogeneity of distortion E_h and the global aspect ratio E_r (from [5]) are also listed and compared in Table 1. On these two measurements, the results from our *WireWarping* approach do not show more advantages than LSCM (see Table 1). This is because E_h and E_r relate to the shape of all triangles after flattening. However, our approach does not optimize the shape of each element.

The last test is given on a symmetrical non-developable surface — a quarter-sphere patch (see Fig. 11. The results from both the progressive warping scheme (Fig. 11(b)) and the global warping scheme (Fig. 11(c)) of *WireWarping* are symmetrical horizontally. The result from global warping is also symmetrical in the vertical direction but the progressive warping is not. The 2D piece generated by LSCM is neither horizontally nor vertically symmetrical (see Fig. 11(d)).

Fig. 12 shows an application of our *WireWarping* approach in the shoe industry. After designing the cover of a shoe based on the shape of shoe last, the industry looks for a tool that can automatically compute the shape of leather pieces so that the shoe can be fabricated from them. In Fig. 12, the 2D pieces in Fig. 12(b) are generated from the 3D surface patches given in Fig. 12(a) by the global warping scheme of *WireWarping*. Lengths of feature curves are retained and the surface-angle distortions along the feature curves are optimized.

5.1. Limitations

By our experience of applying the technique presented in this paper to the garment industry, the most difficult part of using *WireWarping* is how to choose appropriate key feature curves. As demonstrated in Fig. 8, Fig. 8(b)–(g) show that significantly different flattening results can be obtained when choosing different key feature curves. The selection of key feature curves does very much depend on the experience of users. Therefore, for the given model with new set of feature curves or new topology, we usually start the study by applying

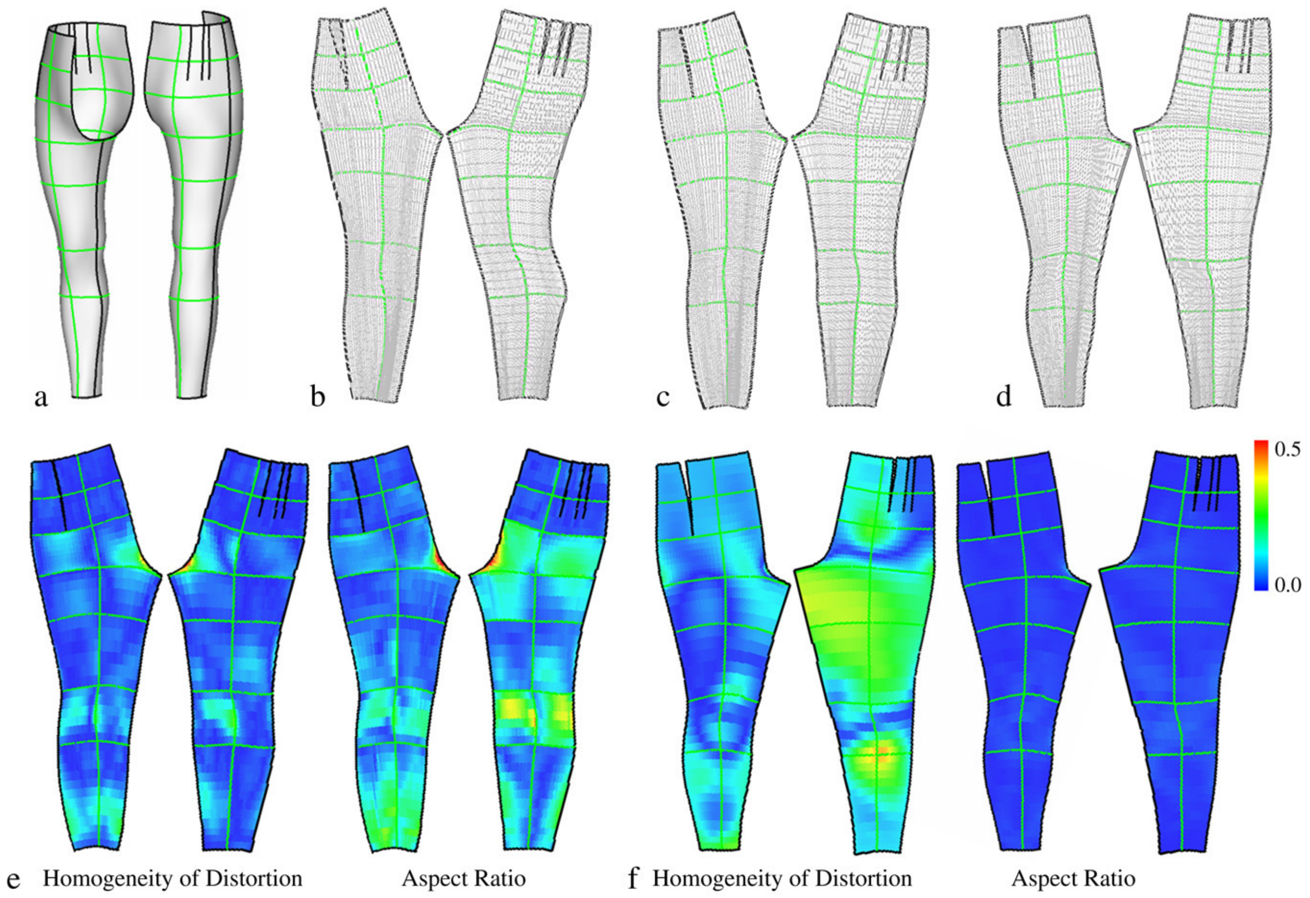

Fig. 10. Example for flattening wetsuit — the pants: (a) given surface with darts and feature curves, (b) the results from the progressive warping scheme, (c) the results from the global warping scheme, (d) the results of LSCM [20] that the boundaries to be sewed are with different lengths, and the colour map for showing the *Homogeneity of Distortion* and the *Aspect Ratio* [5] on the results from (e) the global warping scheme versus (f) LSCM [20]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

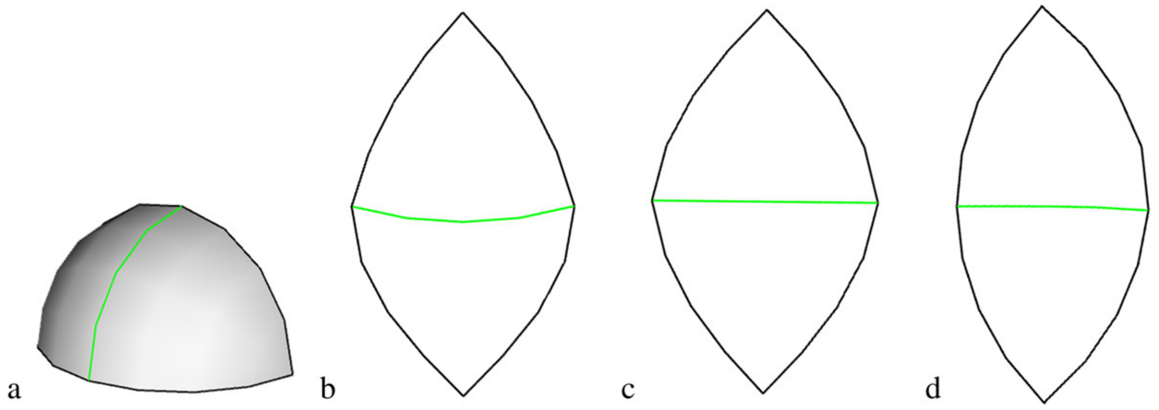

Fig. 11. Example for flattening a quarter-sphere: (a) the given surface, (b) the result from the progressive warping scheme, (c) the result from the global warping scheme, and (d) the result of LSCM [20].

the global warping scheme with all feature curves as accessory feature curves only. After getting a first flattening result, we can then change some accessory feature curves to key feature curves by setting their planar shapes.

Key feature curves on a given surface may not always be compatible to each other. For example, the 3D pants which were previously shown in Fig. 8, if both the curves at the mid-thigh and at the mid-calf are set to be key feature curves, they will distort other feature curves between them (see Fig. 13). Therefore, in order to get satisfactory results, we need to release some key feature curves into accessory feature curves as the setup shown in Fig. 8(e). One of our future work is to study the conditions for setting compatible key feature curves.

There are also some extreme cases that the given surface patch cannot be effectively flattened into a planar piece by preserving the lengths of all feature curves — even if all of

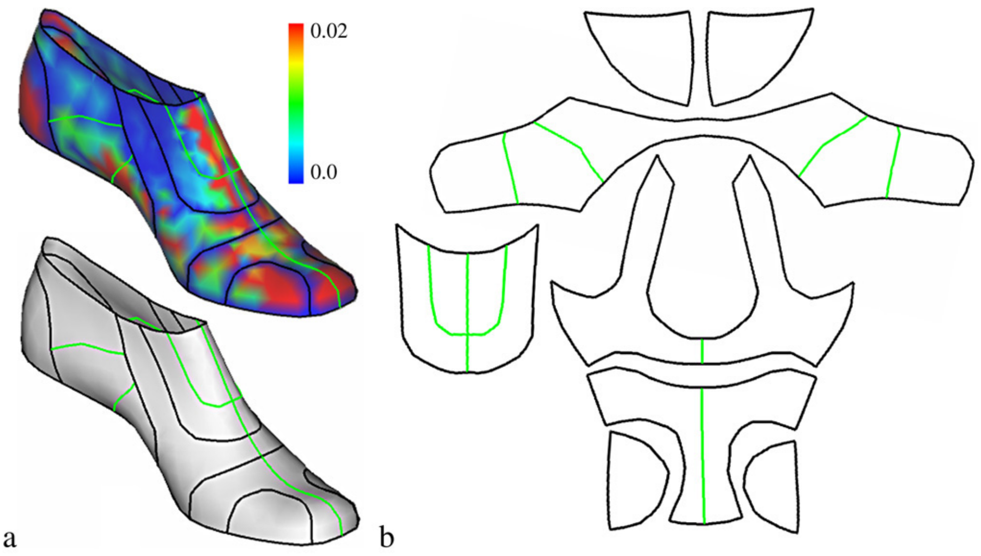

Fig. 12. The application of *WireWarping* in the shoe industry: (a) the colour map for the developability of surfaces on a designed shoe (top) and the designed model, and (b) the flattened 2D patterns that can be used to fabricate the shoe by leather (the lengths of feature curves are preserved). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

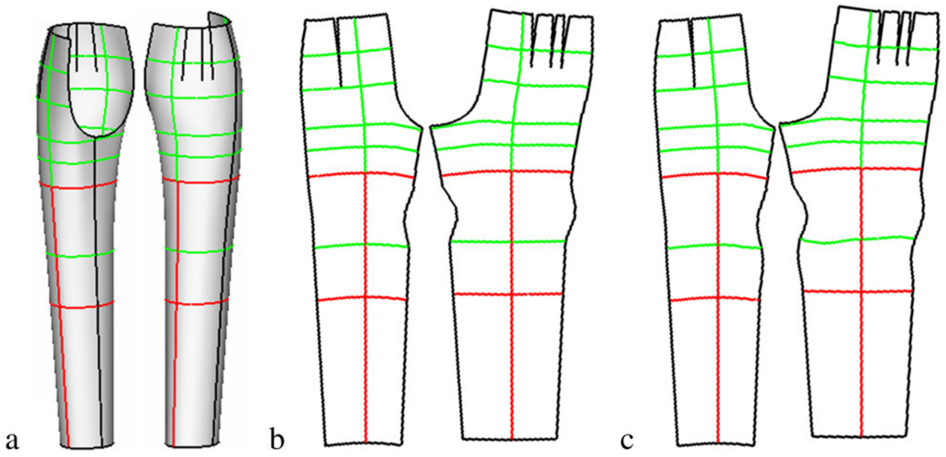

Fig. 13. Too many key feature curves may be incompatible to each other such that unexpected distortions are given on the flattening results: (a) the given 3D pants with key feature curves at the mid-thigh, the mid-calf and the lower part of grainline, (b) results of the progressive warping scheme, and (c) results of the global warping scheme which gives more distorted feature curves.

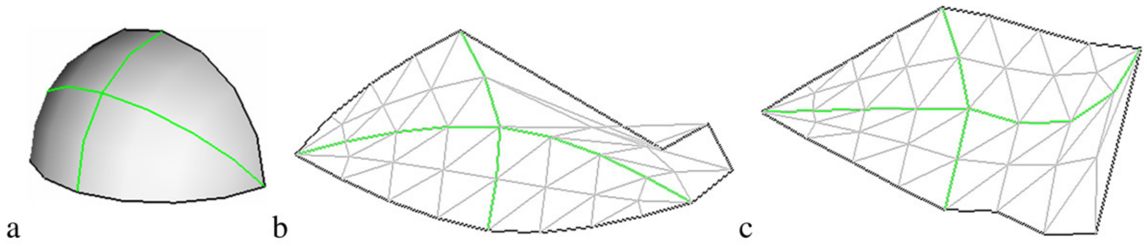

Fig. 14. Too many feature curves (even if only serve as accessory feature curves) on a surface far from developable can make the *WireWarping* approach fail: (a) the quarter-sphere with two accessory feature curves, (b) the result from the progressive warping scheme, and (c) the result from the global warping scheme.

them are defined as accessory feature curves only (e.g. the quarter-sphere in Fig. 14 with two feature curves). For these cases, we may release some of the feature curves (e.g. as shown in Fig. 11) so that the patch can be flattened. We could also conduct the method presented in [37] to deform a given surface into the shape that is more developable.

6. Conclusion

A novel approach, *WireWarping*, for computing the flattened planar surface patch from a given piecewise linear surface has been presented in this paper, which can fully preserve the length of edges on feature curves — this is very important for many industrial applications. *WireWarping* simulates warping a given 3D surface patch into 2D with the boundaries and feature curves as tendon wires to preserve the lengths of edges on them. During warping, the surface-angle variation between edges on wires are minimized so that the shape of a piece in 2D is similar to its corresponding 3D patch. Two schemes — the progressive warping and the global warping schemes are developed, where the former one is more flexible for local shape control and the latter one gives better performance on highly curved patches. Experimental results in this paper show that *WireWarping* can successfully flatten given piecewise linear surface patches into 2D pieces while preserving the lengths of edges on feature curves.

Acknowledgments

The author would like to thank TPC (HK) Ltd for providing the 3D surface patches of garment suits, and thank Dr Penelope Watkins for proofreading the manuscript.

References

[1] Azariadis PN, Aspragathos NA. Design of plane development of doubly curved surface. Computer-Aided Design 1997;29:675–85.
[2] Azariadis PN, Aspragathos NA. Geodesic curvature preservation in surface flattening through constrained global optimization. Computer-Aided Design 2001;33(8):581–91.
[3] Aono M, Breen DE, Wozny MJ. Modeling methods for the design of 3D broadcloth composite parts. Computer-Aided Design 2001;33:989–1007.
[4] Aono M, Breen DE, Wozny MJ. Fitting a woven-cloth model to a curved surface: Mapping algorithms. Computer-Aided Design 1994;26:278–92.
[5] Azariadis PN, Sapidis NS. Planar development of free-form surfaces: quality evaluation and visual inspection. Computing 2004;72(1–2):13–27.
[6] Belkin M, Niyogi P. Laplacian Eigenmaps for dimensionality reduction and data representation. Neural Computation 2003;15:1373–96.
[7] Chapra SC, Canale RP. Numerical methods for engineers: With software and programming applications. 4th ed. 2003. p. 289–95.
[8] Bennis C, Vezjen J-M, Iglesias G. Piecewise surface flattening for non-distorted texture mapping. Computer Graphics 1991;24(4):237–46.
[9] Chen H-Y, Lee I-K, Leopoldseder S, Pottmann H, Randrup T, Wallner J. On surface approximation using developable surfaces. Graphical Models and Image Processing 1999;61:110–24.
[10] Chu CH, Séquin CH. Developable Bézier patches: Properties and design. Computer-Aided Design 2002;34(7):511–27.
[11] Duff IS, Erisman AM, Reid JK. Direct methods for sparse matrices. Oxford: Clarendon Press; 1989.
[12] Decaudin P, Julius D, Wither J, Boissieux L, Sheffer A, Cani M-P. Virtual garments: A fully geometric approach for clothing design. Computer Graphics Forum 2006;25(3):625–34.
[13] Desbrun M, Meyer M, Alliez P. Intrinsic parameterizations of surface meshes. Computer Graphics Forum 2002;21(3):209–18.
[14] do Carmo MP. Differential geometry of curves and surfaces. Englewood Cliffs (NJ): Prentice-Hall.
[15] Floater MS, Hormann K. Surface parameterization: A tutorial and survey. In: Dodgson NA, Floater MS, Sabin MA, editors. Advances in multiresolution for geometric modelling. Heidelberg: Springer-Verlag; 2005. p. 157–86.
[16] Karni Z, Gotsman C, Gortler SJ. Free-boundary linear parameterization of 3D meshes in the presence of constraints. In: Proceedings of shape modeling international. 2005. p. 266–75.
[17] Li S, Demmel J, Gilbert J. SuperLU, http://crd.lbl.gov/xiaoye/SuperLU/, February 2006.
[18] Lee Y, Kim H-S, Lee S. Mesh parameterization with a virtual boundary. Computers & Graphics 2002;26:677–86.
[19] Leopoldseder S, Pottmann H. Approximation of developable surfaces with cone spline surfaces. Computer-Aided Design 1998;30:571–82.
[20] Lévy B, Petitjean S, Ray N, Maillot J. Least squares conformal maps for automatic texture atlas generation. ACM Transactions on Graphics 2002; 21:362–71.
[21] Liu Y, Pottmann H, Wallner J, Yang Y-L, Wang W. Geometric modeling with conical meshes and developable surfaces. ACM Transactions on Graphics 2006;25(3):681–9.
[22] Liu YS, Yu PQ, Du MC, Yong JH, Zhang H, Paul JC. Mesh parameterization for an open connected surface without partition. In: Proceedings of the ninth international conference on computer aided design and computer graphics. 2005. p. 306–10.
[23] Manning JR. Computerized pattern cutting: Methods based on an isometric tree. Computer-Aided Design 1980;12(1):43–7.
[24] Autodesk Maya, http://www.autodesk.com/maya.
[25] McCartney J, Hinds BK, Seow BL. The flattening of triangulated surfaces incorporating darts and gussets. Computer-Aided Design 1999; 31:249–60.
[26] Mortenson ME. Geometric modeling. 2nd ed. New York: Wiley; 1997.
[27] Meyer M, Desbrun M, Schröder P, Barr AH. Discrete differential-geometry operators for triangulated 2-manifolds. In: Visualization and mathematics III. Springer; 2003. p. 35–58.
[28] Nutbourne AW, McLellan PM, Kensit RML. Curvature profiles for plane curves. Computer-Aided Design 1972;4(4):176–84.
[29] Nocedal J, Wright SJ. Numerical optimization. Springer-Verlag; 1999.
[30] Peternell M. Recognition and reconstruction of developable surfaces from point clouds. In: Proceedings of geometric modeling and processing. 2004. p. 301–10.
[31] Press WH, Flannery BP, Teukolsky SA, Vetterling WT. Numerical recipes in C: The art of scientific computing. 2nd ed. Cambridge: Cambridge University Press; 1995.
[32] Peternell M, Steiner T. Reconstruction of piecewise planar objects from point clouds. Computer-Aided Design 2004;36:333–42.
[33] Pottmann H, Wallner J. Approximation algorithms for developable surfaces. Computer Aided Geometric Design 1999;16:539–56.
[34] Sun X, Hancock ER. Fast isometric parametrization of 3D triangular mesh. In: Proceedings of british machine vision conference. 2005.
[35] Sheffer A, Lévy B, Mogilnitsky M, Bogomjakov A. ABF++: fast and robust angle based flattening. ACM Transactions on Graphics 2005;24(2): 311–30.
[36] Sheffer A, de Sturler E. Parameterization of faceted surfaces for meshing using angle based flattening. Engineering with Computers 2001;17(3): 326–37.
[37] Wang CCL. Towards flattenable mesh surfaces. Computer-Aided Design 2007; doi:10.1016/j.cad.2007.06.001.
[38] Wang CCL. Computing length-preserved free boundary for quasi-developable mesh segmentation. IEEE Transactions on Visualization and Computer Graphics 2008;14(1):25–36.
[39] Wang CCL, Smith SSF, Yuen MMF. Surface flattening based on energy model. Computer-Aided Design 2002;34(11):823–33.
[40] Wang CCL, Tang K. Achieving developability of a polygonal surface by minimum deformation: A study of global and local optimization approaches. The Visual Computer 2004;20(8–9):521–39.
[41] Wang CCL, Tang K, Yeung BML. Freeform surface flattening by fitting a woven mesh model. Computer-Aided Design 2005;37:799–814.
[42] Wang CCL, Wang Y, Yuen MMF. On increasing the developability of a trimmed NURBS surface. Engineering with Computers 2004;20(1): 54–64.
[43] McCartney J, Hinds BK, Chong KW. Pattern flattening for orthotropic materials. Computer-Aided Design 2005;37:631–44.
[44] Zigelman G, Kimmel R, Kiryati N. Texture mapping using surface flattening via multi-dimensional scaling. IEEE Transactions on Visualization and Computer Graphics 2002;8:198–207.
[45] Zhong Y, Xu B. A physically based method for triangulated surface flattening. Computer-Aided Design 2006;38:1062–73.