Flattenable Mesh Surface Fitting on Boundary Curves

Charlie C.L. Wang
Member of ASME
Department of Mechanical and Automation Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
Email: cwang@mae.cuhk.edu.hk

This paper addresses the problem of fitting flattenable mesh surfaces in $\Re^3$ onto piecewise linear boundary curves, where a flattenable mesh surface inherits the isometric mapping to a planar region in $\Re^2$. The developable surface in differential geometry shows the nice property. However, it is difficult to fit developable surfaces to a boundary with complex shape. The technique presented in this paper can model a piecewise linear flattenable surface that interpolates the given boundary curve and approximates the cross-tangent normal vectors on the boundary. At first, an optimal planar polygonal region is computed from the given boundary curve $B \in \Re^3$, triangulated into a planar mesh surface, and warped into a mesh surface in $\Re^3$ satisfying the continuities defined on B. Then, the fitted mesh surface is further optimized into a Flattenable Laplacian (FL) mesh which preserves the positional continuity and minimizes the variation of cross-tangential normals. Assembled set of such FL mesh patches can be employed to model complex products fabricated from sheets without stretching.

1 Introduction

In many industries, the products with complex shapes are made by a set of assembled patches where each is warped from a planar sheet of material (e.g., the metal sheet in ship industry, the textile pattern in apparel industry, the leather in shoe industry and furniture industry). The sheet after warping is expected to have no stretch. This is because that embedding stretch easily leads to material fatigue and failure. At present, geometry design in these industries is conducted in a trial-and-error manner. A designer will first figure 2D pieces on an alternative material and then make a prototype by the patterns to check whether the fitting is good. If the result is not satisfied, the designer modifies the 2D pieces by his own experiences and makes prototype once more. The cycle of revising and prototyping will be repeated, which is very inefficient. Besides, it is very difficult to find a 2D piece that fits a given 3D boundary curve. This in fact leads to a more serious problem that the product made from the patterns generated in this way may give some unwanted shape or give gaps along the boundaries. Therefore, a lot of designers in these industries are waiting for a three-dimensional modelling system, by which they can design some curves directly in 3D and fit patches onto the curves. The fitted patch must be a surface that can be flattened into a planar piece without stretching. The flattenable Laplacian mesh surface defined in [1] is employed here to fit boundary curves. The approach presented in this paper focuses on how to interpolate boundary curves while approximating cross-tangent normals, which has not been covered by [1].

1.1 Problem definition

Without lose of generality, we assume that the boundary of a surface patch to be fitted is defined as a piecewise linear curve $B = \{\mathbf{e}_i\}$ in $\Re^3$ coupled with a set of normal vectors $\Pi = \{\mathbf{n}_i\}$, where $\mathbf{n}_i$ is the normal vector of tangent plane defined on the line-segment $\mathbf{e}_i$. The problem we are going to solve is that: to find a piecewise linear mesh surface $M \in \Re^3$ interpolating the given boundary B and approximating the coupled normal vectors in Π, where M can be flattened into a planar patch without stretching.

This is actually an ill-posed problem; i.e., we cannot always find a flattenable mesh surface M satisfying any arbitrary given B and Π. Therefore, we formulate the computation as an optimization problem in this paper. Being a popular representation of piecewise linear surfaces, we employ triangular mesh patch to present M.

Property 1 For an inner triangular mesh vertex $\mathbf{v}_p$, if and only if the summed inner angle, $\theta(\mathbf{v}_p) = \sum_j \theta_j$, around it is identically 2π, the triangles around it can be flattened into a plane without distortion.

Figure 1 gives an illustration for this property. By which, we have the following definitions.

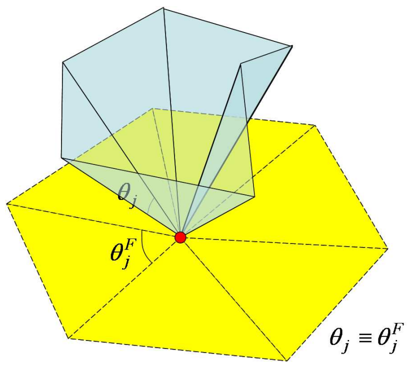

Fig. 1. Illustration for vertex flattenability – the inner angles before and after flattening the triangles around a vertex are identical.

Definition 1 An inner triangular mesh vertex $\mathbf{v}_p$ is named as *flattenable vertex* when $\theta(\mathbf{v}_p) = 2\pi$.

Definition 2 The value of $\varpi(\mathbf{v}_p) = |\theta(\mathbf{v}_p) - 2\pi|$ is defined as the *flattenability* at $\mathbf{v}_p$.

Definition 3 A triangular *flattenable mesh patch* is defined as a disk-like triangular mesh with all its interior vertices flattenable.

$\varpi(\mathbf{v}_p)$ is in fact similar to the discrete form of Gaussian curvature defined in [2], which is the smaller the better for our problem. The reason why we do not adopt the name *developable* (or *discrete developable*) as [3–5] is that: when discussing developable property, it is usually derived from differential geometry on regular surface points; for a sharp (or singular) vertex as shown in Fig.1, which is not differentiable, it is more appropriate to name it as *flattenable* (*unflattenable*) rather than *developable* (*undevelopable*).

1.2 Related work

There are a lot of mesh parameterization approaches in the computer graphics literature [6–11] and surface flattening approaches in the computer-aided design literature [12–17] which adopt various criteria to compute the planar shape from a 3D surface patch by minimizing their differences. However, few work tries to fit a stretch-free flattenable surface onto a given 3D boundary.

In order to segment a given model into disk-like patches with less stretch, Julius et al. developed an algorithm in [3] to separate a given model into quasi-conical proxies. Based on a similar idea, they deformed a given mesh surface in [18] instead of segmenting it so that the deformed surface locally approximates a conical surface. It is a sufficient (but not necessary) condition that a conical mesh surface is flattenable. A more general representation for flattenable surface (or discrete developable surface) is needed. In [5], Wang and Tang adopted the definition of Gaussian curvature in discrete differential geometry [2] to define the measurement for the discrete developablity on given polygonal mesh surfaces. They conducted a constrained optimization approach to deform mesh surfaces so that increase their discrete developability. Although [5] is akin to the second phase of our algorithm in this paper, its converging speed is much slower. Liu et al. in [4] presented a novel PQ meshes – quadrilateral meshes with planar faces, which is useful to the application of architecture design. The computation of PQ meshes is based on the constrained optimization with the position of mesh vertices as variables. The developable surface constructed by [4] is still simple (i.e., with the shape similar to ruled surfaces). Recently, in [1], we formulate the method to compute flattenable meshes also by using the constrained optimization, and can model the flattenable meshes with more complex shape on boundaries with complex shape (e.g., the surfaces in Fig.10). However, the cross-tangent constraints and the method to compute initial fitting to given boundary curves have not been addressed in [1], which will be the focus of this paper.

The study of flattenable mesh surfaces relates to the developable surface in differential geometry [19], where the definition of a developable surface is derived on 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. The key concept in characterizing the developability is Gaussian curvature — in general, a surface is developable if and only if the Gaussian curvature of every point on it is zero. Every surface enveloped by a one-parameter family of planes is a developable surface. By this idea, some researches in literature focused on modelling [20–22] or approximating [23–25] 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 (e.g., the surfaces in Fig.12). Another limitation of these approaches is that they can only model surface patches with 4-sided boundaries as the surfaces are usually defined on a squared parametric domain. Although trimmed surfaces were considered in [26], the modelling ability for freeform objects by these approaches is still very limited.

1.3 Contributions

To model flattenable mesh surfaces by given boundary curves, in this paper, we 1) introduce the cross-tangent approximated Flattenable Laplacian (FL) mesh as a novel freeform surface representation, and 2) develop a novel length-preserved optimal boundary computation methods for computing the initial shape of a surface under fitting. These two technical contributions yield the new functionality that we can fit stretch-free flattenable mesh surfaces onto a freeform boundary. This function has not been provided by existing approaches in literature.

The rest of this paper is organized as follows. We first give the overview of our two-steps algorithm in section 2. The first step of our method about how to construct the initial fitting surface will be addressed in section 3, and section 4 presents the second step about how to further process the surface into a cross-tangent approximated flattenable mesh surface. After giving results in section 5, our paper ends with the conclusion section.

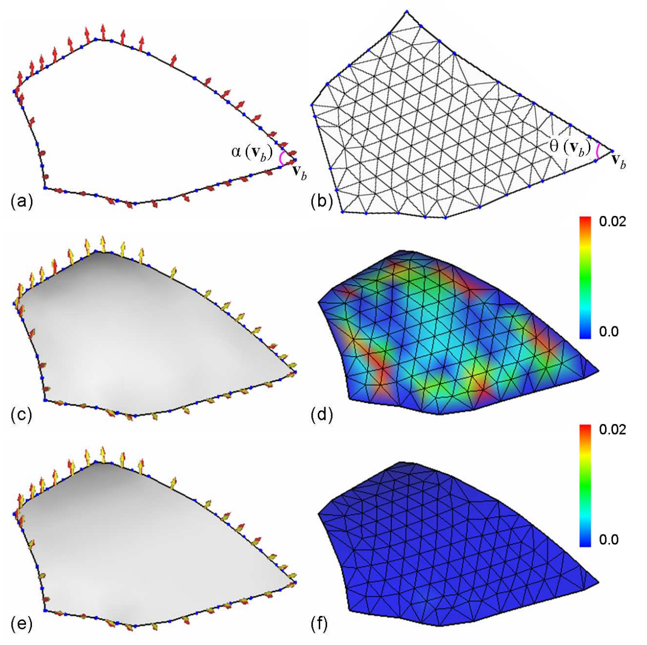

Fig. 2. Method overview: (a) the given boundary curve (blue knots are endpoints of line segments) and the normal vectors (red arrows) defined on segments, (b) the computed optimal planar boundary and tessellated planar mesh patch, (c) the warped mesh surface from the planar mesh as the initial fitting patch (where the yellow arrows represent the normal vectors on the boundary of the warped patch), (d) the color map of flattenablity at vertices of the initial fitting patch, (e) the final flattenable mesh computed from (c), and (f) the corresponding flattenability map of (e).

2 Method Overview

For a piecewise linear boundary curve $B \in \Re^3$ coupled with the normal constraints Π (e.g., in Fig.2(a), the blue knots represent the endpoints of line segments on B and the red arrows are the normal vectors), we develop an algorithm with two-steps to model a flattenable mesh surface fitting this boundary.

1. *Initial Patch Fitting*: Purpose of the algorithm in this phase is to construct an initial mesh surface, which interpolates the given boundary curve B and satisfies the normal constraints Π defined on the boundary line segments. A warping based method is developed for this. Firstly, a length-preserved optimal boundary B^p is computed in $\Re^2$ by B and Π, where all the line segments on B^p are with the same length as their corresponding segments on B and the shape differences between B^p and B is minimized (see the boundary of the patch in Fig.2(a) and (b)). The computation of B^p is formulated as a constrained optimization problem in the angle space. By $B^p \in \Re^2$, we then tessellate the region bounded by B^p into a planar triangular mesh patch M^p (as Fig.2(b)). After that, M^p is warped into a shape M^0 interpolating the endpoints of segments on B and the normal vectors in Π by radial-basis functions (RBF) (see Fig.2(c)).
2. *Modelling of Cross-Tangents Approximated FL Meshes*: The initial fitting patch, M^0, in general will have quite a few vertices with large flattenability values (see Fig.2(d)) – in other words, it is not flattenable. Therefore, we need to further process the mesh M^0 into a flattenable mesh surface M. This modelling task is formulated as a constrained optimization problem where the objective function is defined by a Laplacian smoothness term, a weak position term for trying to remain the shape of M^0, and a strong position term for approximating the boundary normal vectors. The constraints of the optimization problem are derived from the vertex flattenability. To speed up the convergency of computation, a multi-level optimization algorithm is introduced to generate the final flattenable mesh (e.g., Fig.2(e) and (f)).

The following sections will present details for these two steps.

3 Initial Patch Fitting

3.1 Length-preserved planar boundary computation

The method for computing an optimal planar boundary B^p by the given piecewise linear boundary $B \in \Re^3$ is presented in this section, where B^p is also a piecewise linear curve but in $\Re^2$. The requirements on B^p are that

1. B^p and B have the same number of linear segments, and each linear segment on B^p have the same length with its corresponding segment on B.
2. The shape of B^p is similar to B, and is not self-intersected.

The first requirement is easy to understand — since the fitted mesh should be able to be flattened into a two-dimensional piece without stretching, no length variation is allowed on the boundaries in $\Re^3$ and $\Re^2$. However, only constraining the edge length is not enough, we also need some morphological term to control the similarity between B^p and B.

We employ angles to measure the morphological difference between B^p and B. For a vertex $\mathbf{v}_b \in B$, suppose that its surface inner angle on the constructed surface M is $\alpha(\mathbf{v}_b)$ and its inner turning angle on B^p is $\theta(\mathbf{v}_b)$ (see the illustration in Fig.2(a) and Fig.2(b)), the shape similarity between B^p and B can be evaluated by the following boundary angle error once the edge length consistency is preserved between B^p and B.

$$\varepsilon_\theta = \frac{1}{n} \sum_{\mathbf{v}_b \in B} (\alpha(\mathbf{v}_b) - \theta(\mathbf{v}_b))^2 \qquad (1)$$

In ε_θ, n is the number of vertices on B which is actually the same as the number of line segments (i.e., edges) on B and B^p. The value of $\alpha(\mathbf{v}_b)$ has not been stated. As illustrated in Fig.3(a), considering about a boundary vertex $\mathbf{v}_b \in B$ and its previous and next vertices $\mathbf{v}_{b-}$ and $\mathbf{v}_{b+}$ on B, without loss of generality, we can assume that the normal vectors $\mathbf{n}_{e+}$ and $\mathbf{n}_{e-}$ defined on the edges $\mathbf{e}_-$ and $\mathbf{e}_+$ are with small difference. Then, the value of angle $\alpha(\mathbf{v}_b)$ can be predicted as $\alpha(\mathbf{v}_b) =$

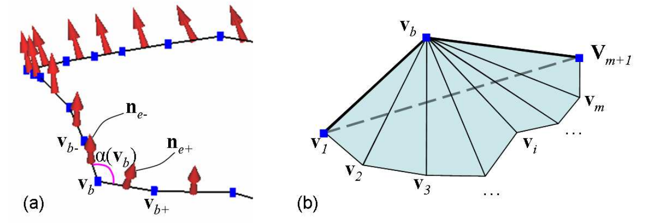

Fig. 3. Angles on a boundary vertex $\mathbf{v}_b$: (a) the value of $\alpha(\mathbf{v}_b)$ can be predicted by the projected angle on the plane defined by $\mathbf{v}_b$ and the normal vector $\frac{1}{2}(\mathbf{n}_{e+}+\mathbf{n}_{e-})$, and (b) the value of an optimal $\theta(\mathbf{v}_b)\in\Re^2$ has a lower bound as been addressed in Lemma 1.

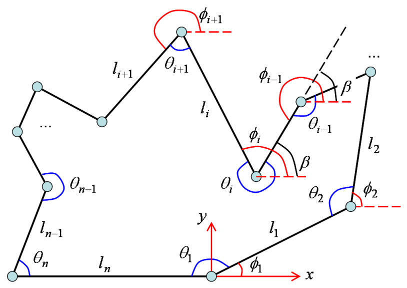

Fig. 4. The closed-path constraint and the position coincident constraint on the planar boundary.

$\angle\prod(\mathbf{v}_{b-})\prod(\mathbf{v}_b)\prod(\mathbf{v}_{b+})$ where $\prod(...)$ defines a projection of points onto the plane passing $\mathbf{v}_b$ and with the normal vector $\frac{1}{2}(\mathbf{n}_{e+}+\mathbf{n}_{e-})$. Note that the vertices on B are ordered anti-clockwise. The following lemma gives the lower bound of an optimal $\theta(\mathbf{v}_b)$.

Lemma 1 For a boundary vertex $\mathbf{v}_b\in B$, the optimal angle $\theta(\mathbf{v}_b)\in\Re^2$ should satisfy that $\theta(\mathbf{v}_b)\geq\angle\mathbf{v}_{b-}\mathbf{v}_b\mathbf{v}_{b+}$.

Proof If there is only one triangle at $\mathbf{v}_b$ on the final constructed patch M, we have $\theta(\mathbf{v}_b)=\angle\mathbf{v}_{b-}\mathbf{v}_b\mathbf{v}_{b+}$ as the triangle $\triangle\mathbf{v}_{b-}\mathbf{v}_b\mathbf{v}_{b+}$ should be flattened without distortion.

If there are m triangles adjacent to the vertex $\mathbf{v}_b$ on the final patch M — $\triangle\mathbf{v}_1\mathbf{v}_b\mathbf{v}_2$, $\triangle\mathbf{v}_2\mathbf{v}_b\mathbf{v}_3$, ..., $\triangle\mathbf{v}_{i-1}\mathbf{v}_b\mathbf{v}_i$, ..., $\triangle\mathbf{v}_m\mathbf{v}_b\mathbf{v}_{m+1}$ (see Fig.3(b)), as the optimal boundary B^p should let all these triangles to be flattenable without stretching, we thus have $\theta(\mathbf{v}_b)=\sum_{i=1}^m\angle\mathbf{v}_i\mathbf{v}_b\mathbf{v}_{i+1}$. Repeatedly, $\angle\mathbf{v}_1\mathbf{v}_b\mathbf{v}_2+\angle\mathbf{v}_2\mathbf{v}_b\mathbf{v}_3\geq\angle\mathbf{v}_1\mathbf{v}_b\mathbf{v}_3$, $\angle\mathbf{v}_1\mathbf{v}_b\mathbf{v}_3+\angle\mathbf{v}_3\mathbf{v}_b\mathbf{v}_4\geq\angle\mathbf{v}_1\mathbf{v}_b\mathbf{v}_4$, ..., $\angle\mathbf{v}_1\mathbf{v}_b\mathbf{v}_m+\angle\mathbf{v}_m\mathbf{v}_b\mathbf{v}_{m+1}\geq\angle\mathbf{v}_1\mathbf{v}_b\mathbf{v}_{m+1}$, we therefore have $\theta(\mathbf{v}_b)=\sum_{i=1}^m\angle\mathbf{v}_i\mathbf{v}_b\mathbf{v}_{i+1}\geq\angle\mathbf{v}_1\mathbf{v}_b\mathbf{v}_{m+1}$.

Q.E.D.

Lemma 1 can be expressed as the constraints defined for the optimal planar boundary B^p as that

$$\theta_i\geq\angle\mathbf{v}_{i-1}\mathbf{v}_i\mathbf{v}_{i+1},$$

with θ_i be the planar inner turning angle of a boundary vertex $\mathbf{v}_i\in B^p$. From the *closed-path theorem* (ref. [27]), we know that: for a simple non-self-intersected planar closed path, if its path is anti-clockwise, the total turning is 2π. As shown in Fig.4, the total turning by accumulating vertex turning angles can be computed by $\sum_{i=1}^n(\pi-\theta_i)$, which leads to another constraint for B^p that

$$n\pi-\textstyle\sum_{i=1}^n\theta_i\equiv2\pi.$$

Besides these two constraints, the position coincident constraints should also be added on B^p. By giving the inner turning angles θ_is and placing $\mathbf{v}_1$ at the origin, the planar coordinate (x_i,y_i) of a boundary vertex $\mathbf{v}_i$ 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 been illustrated in Fig.4, we have $\theta_i=2\pi-(\phi_i-\beta)$ at the vertex $\mathbf{v}_i$ and $\beta=\phi_{i-1}-\pi$ at the vertex $\mathbf{v}_{i-1}$, which yields $\phi_i=\pi-\theta_i+\phi_{i-1}$. Together with $\phi_1=\pi-\theta_1$, the general formula for ϕ_i can be derived in terms of θ_b as $\phi_i=i\pi-\sum_{b=1}^i\theta_b$. In order to ensure the planar boundary B^p be closed, we must let (x_{n+1},y_{n+1}) be coincident with the origin, which leads to

$$\textstyle\sum_{i=1}^n l_i\cos\phi_i\equiv0,\qquad\sum_{i=1}^n l_i\sin\phi_i\equiv0.$$

According to the morphological error term in Eq.1 and above three constraints, the computation of B^p from B and Π can be formulated as a constrained optimization problem defined in the angle space

$$\begin{array}{ll} & \arg\min_{\theta_i}\{\sum_i\frac{1}{2}(\theta_i-\alpha_i)^2\}\\ s.t. & n\pi-\sum_i\theta_i\equiv2\pi,\\ & \sum_i l_i\cos\phi_i\equiv0,\quad\sum_i l_i\sin\phi_i\equiv0,\\ & \theta_i\geq\angle\mathbf{v}_{i-1}\mathbf{v}_i\mathbf{v}_{i+1}.\end{array}\qquad(2)$$

To efficiently compute the optimal angles θ_is, the constrained optimization problem is firstly converted into an augmented objective function J_θ by using the Lagrange multipliers, where the inequality constraints are partitioned into an active set and an inactive set in each iteration — only the constraints in the active set are added into J_θ . We employ the sequential linear constrained programming (ref. [28]) to minimize J_θ. The computation converges in few iterations. Fig.5(b) gives an optimal boundary $B^p\in\Re^2$ computed from B and Π given in Fig.5(a).

The convergency of computation in Eq.(2) depends on the given boundary curves and coupled normal vectors. Firstly, if $\mathbf{n}_{e+}+\mathbf{n}_{e-}=0$ for normal vectors on two neighboring line segments, the predicted angle $\alpha(\mathbf{v}_b)$ cannot be determined by planar projection. Then we let $\alpha(\mathbf{v}_b)\geq\angle\mathbf{v}_{b-}\mathbf{v}_b\mathbf{v}_{b+}$. A more serious situation lead to the failure of optimizing B^p is described below.

Lemma 2 For a given boundary curve if the sum of its polygonal angles in $\Re^3$ satisfies $\sum_{i=1}^n\angle\mathbf{v}_{i-1}\mathbf{v}_i\mathbf{v}_{i+1}>(n-2)\pi$, the constrained optimization problem defined in Eq.(2) has no solution.

Proof The proof is very straight forward. If we have $\sum_{i=1}^n\angle\mathbf{v}_{i-1}\mathbf{v}_i\mathbf{v}_{i+1}>(n-2)\pi$, the constraints $n\pi-\sum_i\theta_i\equiv2\pi$ can never be satisfied when preserving another constraint $\theta_i\geq\angle\mathbf{v}_{i-1}\mathbf{v}_i\mathbf{v}_{i+1}$.

Q.E.D.

3.2 Tessellation

After determining the length-preserved optimal planar boundary B^p, we need to tessellate the region surrounded by it into a planar triangular mesh M^p. The tessellation consists of four steps:

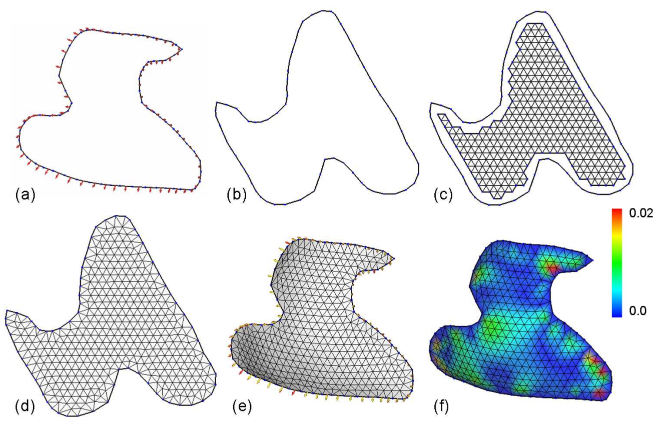

Fig. 5. The steps for computing an initial patch fitting to the given boundary: (a) the given boundary and normal vectors for cross-tangent planes, (b) the computed optimal planar boundary, (c) the inner region is filled with regular triangles, (d) the boundary region is tessellated by the Constrained Delaunay Triangulation (CDT) [29] and the inner vertices are re-positioned through an area-based smoothing, (e) the planar mesh is warped to interpolate the given boundary and cross-tangent planes, and (f) the color map of flattenability at vertices of the warped patch.

1. *Sampling* – Firstly, sampling points are uniformly inserted into the region surrounded by B^p. The sampling points will become the interior vertices of the resultant planar mesh M^p. In order to let the resultant triangles close to regular shape, we shift the position of sampling points in the odd rows by half of the target triangular edge length $\overline{L}$ according to the points in even rows. The points in the same row are sampled with the distance $\overline{L}$, and the distance between two neighboring rows is $\frac{\sqrt{3}}{2}\overline{L}$.
2. *Regular triangulation* – Since the sampling points are added rows by rows, we can fast construct the interior triangles by these sampling points using the layout information of points (e.g., the result shown in Fig.5(c)).
3. *Constrained Delaunay Triangulation* (CDT) – the region between the interior triangles and B^p is then tessellated into triangles by the Constrained Delaunay Triangulation (CDT) [29].
4. *Smoothing* – Lastly, the interior vertices are re-positioned through an area-based smoothing algorithm, where every interior vertex is iteratively moved to the area-weighted centric of its one-ring neighbors for several steps (around 10 in our implementation). In order to achieve a stable smooth, the vertices are moved with a damping factor 0.25.

One example of the tessellated planar patch M^p is shown in Fig.5(d).

3.3 Warping for patch fitting

As each vertex on B^p has a corresponding vertex on B, the problem for warping M^p into a shape M^0 that interpolates B and Π can be defined as: given a set of point V^p defined on B^p and a corresponding point set V on B, how to find a surface M^0 which is transformed from M^p and the deformation from M^p to M^0 is equivalent to the deformation from V^p to V.

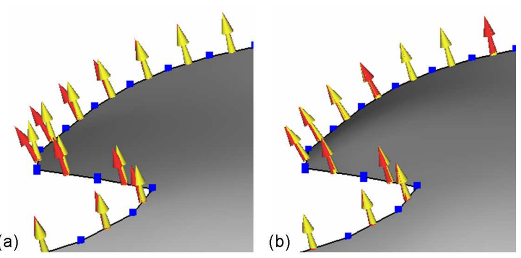

Fig. 6. Different constraints on the boundary yield different surfaces by the RBF-based warping: (a) only the positions and the normal vectors are constrained, and (b) all the positions, the normal vectors, and the cross-tangents are constrained.

Solving this problem is actually to find a deformation function $\Psi(\ldots)$ letting $V = \Psi(V^p)$ so that $M^0 = \Psi(M^p)$ can be determined. The radial basis function (RBF) is the most suitable candidate for this deformation function [30, 31]. A RBF is represented in the piecewise form as

$$\Psi(\mathbf{x}) = p(\mathbf{x}) + \sum_{i}^{n} \lambda_i \phi(\|\mathbf{x} - \tau_i\|) \tag{3}$$

where $p(\mathbf{x})$ is a linear polynomial that accounts for the rigid transformation, the coefficients λ_i are real numbers to be determined and $\|\cdot\|$ is the Euclidean norm on $\Re^3$. To achieve a global deformation, the basis function $\phi(t)$ is chosen as $\phi(t) = t^3$ (the triharmonic spline as [32]). The coefficients λ_i and the coefficients of $p(\mathbf{x})$ can be easily determined by letting $\Psi(\tau_i^p) \equiv \tau_i$ for all pairs of $\tau_i^p \in V^p$ and $\tau_i \in V$ plus the compatibility conditions $\sum_i^n \lambda_i = \sum_i^n \lambda_i \tau_i^p = 0$. The formulated linear equation system has been proven to be positive definite unless all the points in V^p or V are coplanar.

If only the boundary vertices in B^p are added into V^p, they are co-planar — the RBF cannot be successfully computed. Therefore, as suggested in [32], some more points are added into as normal constraints. V^p contains all vertices on B^p. For each edge in B^p, a point located at the middle of the edge but with the depth-coordinate as 0.5τ is also inserted into V^p. Correspondingly, V holds all the vertices on B and the points $\{\mathbf{c}_i + 0.5\tau\mathbf{n}_i\}$ where $\mathbf{c}_i$ is the middle point of the edge $\mathbf{e}_i$, $\mathbf{n}_i$ is the unit normal vector of cross-tangent plane defined on $\mathbf{e}_i$, and τ is the average edge length of M^p. A warping result by this setup is as shown in Fig.6(a). However, from the zoom-view of the result, it is easy to find that the normal vectors on the warped surface (in yellow color) do not fit the given normals (in red) quite well. Therefore, more points are added into the point-sets to constrain the normal vectors on the warped surface. More specifically, we add points $T_{out} = \{\mathbf{c}_i + 0.5\tau(\mathbf{t}(\mathbf{e}_i) \times \mathbf{n}_i)\}$ and $T_{in} = \{\mathbf{c}_i + 0.5\tau(\mathbf{n}_i \times \mathbf{t}(\mathbf{e}_i))\}$ into V for the edges on B (where $\mathbf{t}(\mathbf{e}_i)$ shows the unit vector of an edge $\mathbf{e}_i$). These two sets of points in fact constrain the cross-tangents of the warped surface, as $\mathbf{t}(\mathbf{e}_i) \times \mathbf{n}_i$ and $\mathbf{n}_i \times \mathbf{t}(\mathbf{e}_i)$ are the cross-tangents pointing outwards and inwards the surface boundary $\mathbf{e}_i$ respectively. The new surface warping result is as Fig.6(b), where the boundary normal vectors on M^p follows the given normals in Π.

The reason that we care about the normal constraints on M^0 so much is because that the FL mesh modelling presented in the following section can hardly correct the cross-tangent normal constraints once it is broken. This also reflects the property of numerical optimization technique that the computing result does rely on the initial values very much.

4 Modelling of Cross-Tangents Constrained FL Meshes

The warping defined by RBF is not an isometric mapping (ref. [19]) so that the flattenablity is not preserved on the warped surface. It is not difficult to find from the color map of flattenability (e.g., Fig.2(d) and Fig.5(f)) that many vertices on the warped patch are not flattenable. The second-phase of our approach presented in this section will further process the initial fitting surface M^0 into a final flattenable mesh surface M.

4.1 Formulation

By Definition 3, we have clearly stated the condition for a flattenable mesh patch. However, the geometry, more specifically, the smoothness of a flattenable mesh has not been controlled. Derived from the famous Laplacian operator, authors in [33–37] define the smoothness condition for a mesh vertex $\mathbf{v}_i \in \Re^3$ as

$$\mathbf{v}_i - \frac{1}{|N(\mathbf{v}_i)|} \sum_{j \in N(\mathbf{v}_i)} \mathbf{v}_j = 0 \tag{4}$$

where $N(\mathbf{v}_i)$ is the set of 1-ring neighboring vertices of $\mathbf{v}_i$, and $|...|$ denotes the number of elements in a set. All interior vertices of a Laplacian smooth mesh surface M^s satisfy Eq.(4), which leads to a linear equation system $L\mathbf{v} = 0$ with L known as the Laplacian operator

$$L_{i,j} = \begin{cases} 1 & (i = j) \\ -\frac{1}{|N(\mathbf{v}_i)|} & (j \in N(\mathbf{v}_i)) \\ 0 & (otherwise) \end{cases} .$$

Definition 4 A *Flattenable Laplacian* (FL) mesh is a mesh surface patch which can be flattenable into two-dimensional pieces without stretching, and at the meanwhile minimizes the fairness energy function defined by Laplacian operators.

The following part of this section will focus on the mathematical tool to process the initial fitting patch M^0 into a final FL mesh M. In short, the final mesh M should:

1. be a Flattenable Laplacian mesh (by Definition 4);
2. approximate the shape of M^0;
3. interpolate the given boundary curve B and approximate the normal vectors Π defined on B.

Note that, different from [1], the normal vectors Π on B are constrained. Above three requirements can be formulated into a constrained optimization problem

$$\arg\min_{p \in V_{act}} w_1 J_{fair} + w_2 J_{pos} + w_3 J_{tan} \quad s.t.\ \theta_p \equiv 2\pi, \tag{5}$$

where V_{act} is the set of interior vertices on the mesh patch. The smoothness term J_{fair} is defined on the integral of Laplacian operator all over the surface M

$$J_{fair} = \frac{1}{2} \sum_{p \in V_{act}} \psi(\mathbf{v}_p), \tag{6}$$

with $\psi(\mathbf{v}_p) = \|L\mathbf{v}_p\|^2$ being a piecewise function only defined on the Voronoi area of $\mathbf{v}_p$. The smoothness term is with a weight $w_1 = 1.0$. The gradient of a discretized J_{fair} to a vertex $\mathbf{v}_p \in V_{act}$ is

$$\frac{\partial J_{fair}}{\partial \mathbf{v}_p} = \mathbf{v}_p - \frac{1}{|N(\mathbf{v}_p)|} \sum_{k \in N(\mathbf{v}_p)} \mathbf{v}_k. \tag{7}$$

The position functional J_{pos} is defined to minimize the difference between the new surface M and the given surface M^0

$$J_{pos} = \frac{1}{2} \sum_{p \in V_{act}} \|\mathbf{v}_p - \mathbf{v}_p^0\|^2 \tag{8}$$

with $\mathbf{v}_p^0$ being the original position of $\mathbf{v}_p$ on M^0. J_{pos} is weakly weighted with $w_2 = 0.1$. The last term in the constrained optimization problem defined in Eq.(5) is for constraining the normal vectors of cross-tangent planes on the processed surface as

$$J_{tan} = \frac{1}{2} \sum_{p \in V_{tan}} \|\mathbf{v}_p - \mathbf{v}_p^p\|^2. \tag{9}$$

Letting V_{bnd} be the set of vertices in the triangles that are adjacent to the edges in B, the set V_{tan} in J_{tan} is then defined as $V_{tan} = \{\mathbf{v}_t | \mathbf{v}_t \in V_{bnd} \bigcap (M^0 \setminus B)\}$ (i.e., the vertices that are on the triangles adjacent to B but not on B). The tracking position $\mathbf{v}_p^p$ defined in J_{tan} is given in the way that:

> Suppose that $\mathbf{v}_p$ is on a triangle $\triangle \mathbf{v}_{i+1}\mathbf{v}_p\mathbf{v}_i$ where $\mathbf{v}_i\mathbf{v}_{i+1}$ is the edge $\mathbf{e}_i \in B$ associated with the normal vector $\mathbf{n}_i$, $\mathbf{v}_p^p$ is the projection of $\mathbf{v}_p^0$ on the plane passing the point $\frac{1}{2}(\mathbf{v}_i + \mathbf{v}_{i+1})$ with the normal $\mathbf{n}_i$.

Why we only add the normal constraints softly in the objective function but not hardly in the constraints set? This is because that if we define cross-tangent normals as hard constraints in the constraint set, the feasible region in the solution space defined by the flattenability (i.e., $\theta_p \equiv 2\pi$ for $\mathbf{v}_p \in$

V_{act}) and the feasible region defined by the cross-tangent normals interpolation may have no intersection. Therefore, there will be no solution for Eq.(5). We hence add the constraints softly (i.e., adding them in the objective function) but with a strong weight $w_3 = 100.0$. Once the functional $J_{tan} \equiv 0$ is satisfied, the normal constraints $\{\mathbf{n}_i\}$ are preserved.

4.2 Numerical scheme and a multiple-loop algorithm

For the constrained optimization problem defined in Eq.5, we can still conduct the Lagrange-Multiplier method to form an augmented objective function

$$J_{FL}(X) = w_1 J_{fair} + w_2 J_{pos} + w_3 J_{tan} + \sum_{p \in V_{act}} \lambda_p(\theta_p - 2\pi)$$

with X as the collection of positions of all vertices in V_{act}. When using the sequential linear constrained programming (ref. [28]) to minimize J_{FL} by neglecting the terms coming from the second derivatives of the constraints in the Hessian matrix $\nabla^2 J_{FL}(X)$, the equation $\nabla^2 J_{FL}(X)\delta = -\nabla J_{FL}(X)$ solved at each step is simplified into

$$\begin{pmatrix} H & \Lambda^T \\ \Lambda & 0 \end{pmatrix} \begin{pmatrix} \delta \\ \lambda \end{pmatrix} = \begin{pmatrix} B_p \\ B_\lambda \end{pmatrix}, \qquad (10)$$

where

$$H = \{h_{i,j}\} = \begin{cases} w_1 + w_2 + w_3 & (i = j,\ i \in V_{tan}) \\ w_1 + w_2 & (i = j,\ i \in V_{act} \setminus V_{tan}) \\ -\frac{w_1}{|N(\mathbf{v}_i)|} & (\mathbf{v}_j \in N(\mathbf{v}_i)) \\ 0 & (otherwise) \end{cases},$$

$$\Lambda = \{\frac{\partial^2}{\partial \lambda_i \partial \mathbf{v}_j} \sum_{p \in V_{act}} \lambda_p(\theta_p - 2\pi)\} = \{\frac{\partial \theta(\mathbf{v}_i)}{\partial \mathbf{v}_j}\}.$$

It is easy to find that the matrix H is invertible and the vectors in Λ are linear independent. Therefore, the linear equation system defined in Eq.(10) can always be solved. In the sequential linear constrained programming, the system variables are updated by the vector δ determined from Eq.(10) in a Newton's routine (ref. [38]), where we can use a soft-linear search strategy [28] to determine the actual update step size $\alpha\delta$ with $0 < \alpha \le 1$ to ensure stable convergence and sufficient descent.

However, directly solving the constrained optimization problem in Eq.(5) by the sequential linear constrained programming plus the soft-linear search strategy does not converge fast enough. We find that if the position constraint functional J_{pos} and the normal vector constraint functional J_{tan} are released in some sense, the convergency will be speeded up. Therefore, a multiple-loop optimization algorithm is developed. In this multiple-loop algorithm, the two statements after finishing the inner loop are employed to release the position constraint J_{pos} and the cross-tangent constraint J_{tan} respectively. Note that for a vertex in V_{tan} associated with more than one cross-tangent planes, the average plane is used in the projection. Fig.7 gives a comparison for computing a FL mesh by the direct Newton's routine plus soft-linear search vs. the proposed multi-loop algorithm here. It is easy to find that the multiple-loop algorithm converges much faster.

Algorithm 1 Multiple-loop Optimization

```
repeat
    $i \leftarrow 1$;
    repeat
        Solve Eq.(10);
        Soft-linear search for find an optimal $\alpha$;
        $X \leftarrow X + \alpha\delta$;
        $i \leftarrow i + 1$;
    until $\|\delta\| < 10^{-5}$ OR $i > 5$
    Update the tracking position $v_p^0$ for every vertex $v_p \in V_{act}$ by its current position;
    Update the tracking position $v_b^p$ for every vertex $v_b \in V_{tan}$ by projecting its current position onto the associated cross-tangent plane;
until the maximal flattenability on the mesh is less than $\varepsilon$
```

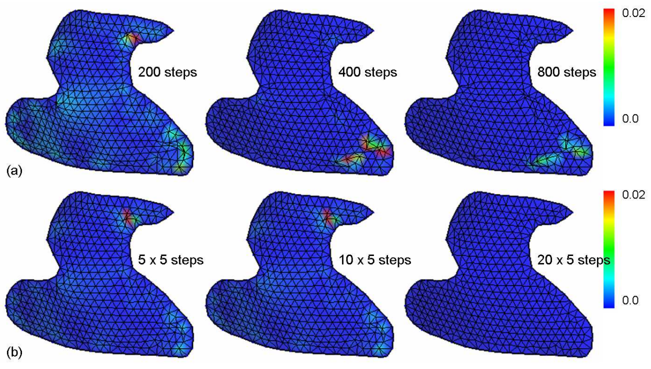

Fig. 7. The comparison of (a) the direct Newton's routine vs. (b) our multiple-loop algorithm for computing a FL mesh surface from an input of Fig.5(f), where the results after 200, 400 and 800 iterations of the direct Newton's routine are shown in (a) and the results after 5, 10 and 20 outer loop iterations (i.e., accumulated 25, 50 and 100 steps) are shown in (b).

5 Results and Discussion

We have implemented the proposed method using C++ plus OpenGL in a prototype system. Several examples have been tested in our prototype system on a PC with Intel Pentium M PIV 1.86GHz CPU + 1GB RAM. The numerical solver for the linear system used in the algorithm *Multiple-Loop Optimization* is the public available SuperLU library from [39].

Our first example is to fit a flattenable mesh surface onto a given boundary curve with double-curved wrinkles. Fig.8 shows the result. The flattenable mesh surface surface with complex double curved wrinkles are constructed, which cannot be modelled by exiting approaches for modelling developable surfaces.

The second example is to fill a hole on a given model. As shown in Fig.9(a), on the back of the moai model, there is a big hole which needs to be filled by a metal sheet. The boundary of the hole is used as the input curve of our algorithm. We use the normal of triangles on the remaining surface adjacent to the boundary as the input normal vectors of cross-tangent planes. Fig.9(b) gives the two-dimensional

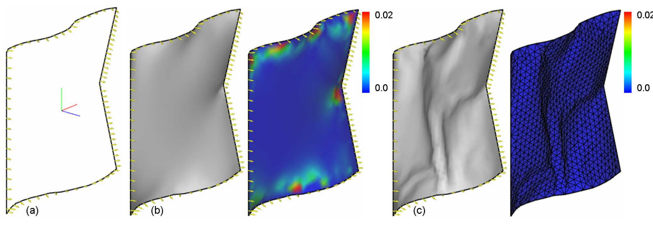

Fig. 8. The wrinkle patch example: (a) the given boundary curve and the cross-tangent normals defined on it, (b) the initial warped patch M^0 and its color map for flattenability, and (c) the result FL mesh surface patch its the color map.

Table 1. Computational Statistic

Example	T_{2D}	T_{warp}	T_{FL}	V_{num}	F_{num}
Wrinkle-Patch	0.4s	0.3s	15.9s	654	1,226
Moai	0.9s	2.5s	13.9s	1,335	2,514
Pants	1.7s	2.3s	4.2s	5,010	8,572
Shoe-I	0.7s	1.5s	2.2s	1,459	2,302
Shoe-II	0.7s	1.4s	1.7s	1,460	2,276

mesh patch before warping, Fig.9(c) shows the final reconstructed FL mesh surface, and the color map in Fig.9(d) illustrates the flattenability on the reconstructed FL patch.

The third example is from the apparel industry, where the modelling in three-dimensional becomes the major trend. By a given virtual human body generated from the 3D laser scanners, the curve-network of a garment (with the cross-tangent normals) can be generated automatically from the mannequin. The most critical work is how to generate the patches fitting this curve-network. Here the patches are required to be flattenable since garments are fabricated from 2D patterns and allows almost no extension. As shown in Fig.10, flattenable mesh patches are constructed by our method piece by piece.

The last example is from the shoe industry. The curve-network associated with normal vectors is generated from the shoe last of a human body. Fig.11(a) shows the curve-network and the normal vectors on it. After determining the initial fitting surface by the RBF-based surface warping (see Fig.11(b)), FL mesh surface patches can be constructed to fit the given curve-network as shown in Fig.11(c). However, unexpected wrinkles are generated at the region near arch (which easily leads to a material failure on the product made by these patches). Therefore, a refined design is given in Fig.12 where the unexpected wrinkles have been successfully removed by adding one more cut near arch. A 2D layout of the pieces that can be used to make this shoe has been given in Fig.13, where the pieces are generated by the flattening method in [16] from the 3D patched in Fig.12(c).

Table 1 gives the computational statistics of the four examples, where T_{2D} is the time needed for the construction of 2D patch, T_{warp} is the time for RBF-based surface warping, T_{FL} represents the time for FL-mesh processing, and V_{num} and F_{num} give the number of vertices and triangles on the result surfaces. All the tests can be finished in less than 20 seconds on a PC with standard configuration, which is an acceptable speed by interactive modelling applications.

5.1 Limitations

The first limitation of the presented FL mesh fitting algorithm is that there are some cases that the computed optimal boundary B^p has global self-intersection as shown in Fig.14(a) although the constraint derived from the closed-path theorem [27] prevents the local self-intersections. If

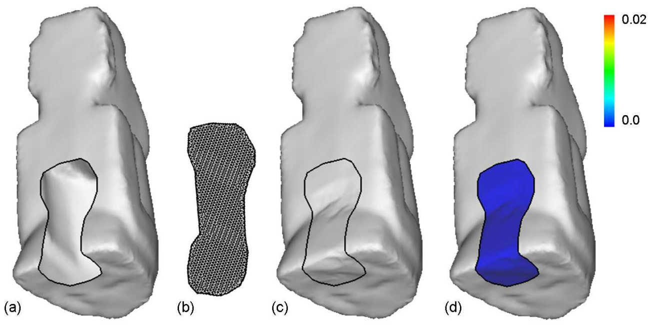

Fig. 9. The hole filling example: (a) the given moai model with a large hole at the back of the model, (b) the 2D mesh patch computed from the hole boundary, (c) the result FL mesh surface patch for filling, and (d) the color map for flattenability on the result FL mesh.

B^p is self-intersected, the region bounded by it cannot be tessellated correctly. Besides, the computation of B^p fails at the cases mentioned in Lemma 2. When meeting these problems, we use a simple but practical method to solve this – a message will be given to users to ask them change the boundary curves B. Subdividing the failed piece into several smaller pieces can always solve the problem.

The second limitation of the FL mesh fitting algorithm proposed in this paper is that the FL mesh representation is not a connectivity invariant representation. Therefore, the computation of FL meshes can be stuck by some topological obstructions although it seldom happens in practice. During the optimization of FL meshes, if high value of $\varpi(\mathbf{v}_p)$ keeps showing on a vertex $\mathbf{v}_p$, we locally refine the triangles around $\mathbf{v}_p$ by the strategy similar to the $\sqrt{3}-$subdivision [40] to add more degree-of-freedom on the mesh under processing.

The last limitation of the presented FL mesh fitting technique is that the G^1 continuity is only approximated but not fully preserved on the given boundary curve B. The situation becomes significant near the region with high curvatures. For example, the surface fitting result of the shoe example shown in Fig.14(b), only G^0 is preserved on the boundary at heel (i.e., the normal vectors on the left patch and the right patch are not consistent). However, for the boundaries falling in the low curvature regions, the normal vectors of the left and right patches are the same (see other boundaries in Fig.14(b)). Although this is a limitation somewhat, it in fact follows the reality when we use flattenable materials with strong tensile stiffness (e.g., leather) to fit a high-curved wire-frame.

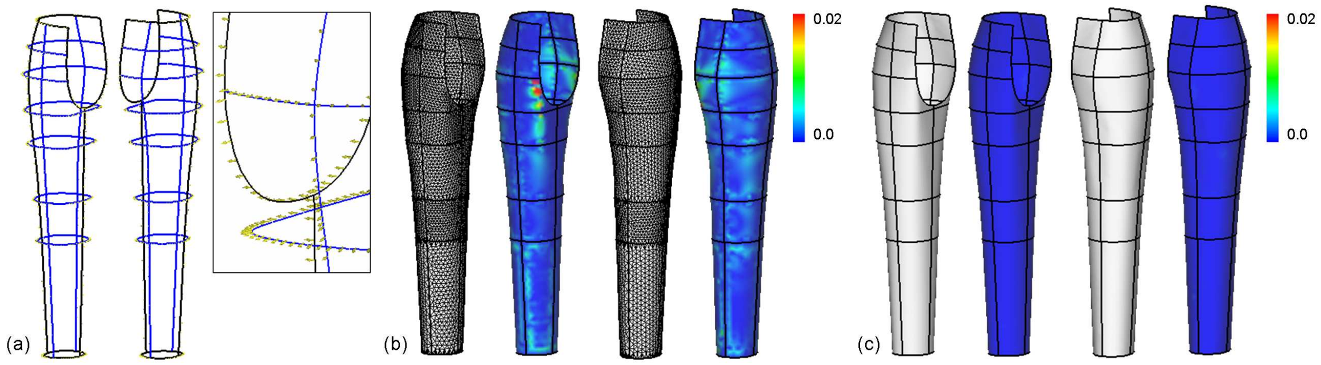

Fig. 10. An example for the flattenable surface modelling in apparel industry: (a) the curve network for fitting flattenable mesh surfaces — the normals of cross-tangent surfaces are defined on the curves (see the yellow arrows), (b) the mesh surface after initial fitting and its flattenability map, and (c) the final FL mesh surface patches fitting the given curve network.

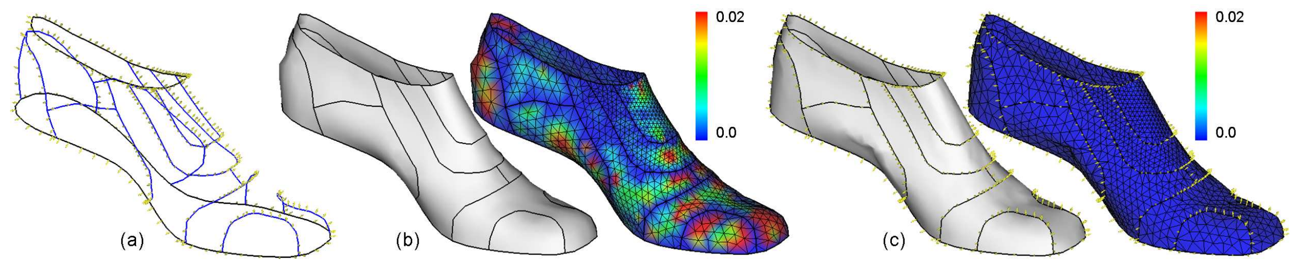

Fig. 11. An example for modelling a shoe: (a) the input curve network with normal vectors, (b) the surface after initial fitting and the color map for illustrating vertex flattenability, and (c) the final FL mesh surface patches (unexpected wrinkles are generated at the region near arch).

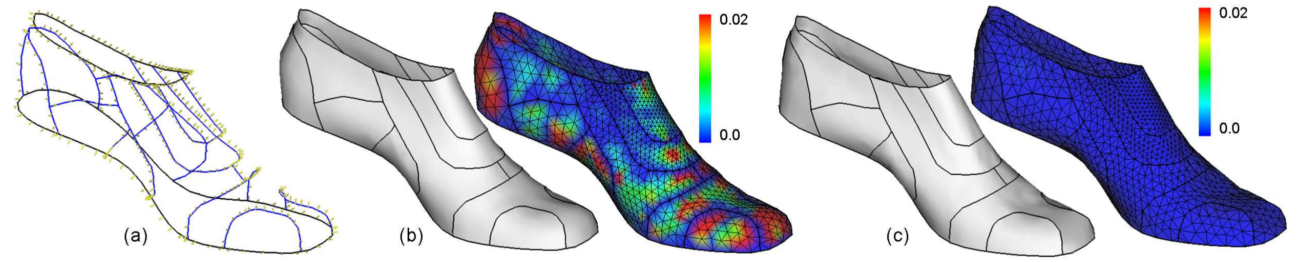

Fig. 12. A refined design for the shoe cover: (a) the input curve networks with normal vectors, (b) the surface after RBF warping (i.e., the initial fitting result), and (c) the final FL mesh patches that are automatically constructed by our approach.

6 Conclusion

In this paper, based on the Flattenable Laplacian (FL) mesh representation [1], we have presented a novel method for fitting a FL mesh surface onto a given boundary piecewise linear curve coupled with normal vectors for cross-tangent planes. The constructed FL mesh patch interpolates the given boundary curve and approximates the cross-tangent normal vectors. Every FL mesh surface can be flattened into a 2D piece without stretching. With this new technique, we can fit stretch-free flattenable mesh surfaces onto a boundary with complex shape. To the best of our knowledge, this is a modelling task that cannot be finished by existing approaches in literature.

References

[1] Wang, C. "Towards flattenable mesh surfaces". *Computer-Aided Design*. accepted.

[2] Meyer, M., Desbrun, M., Schroder, P., and Barr, A., 2002. "Discrete differential-geometry operators for triangulated 2-manifolds". In Proceeding of Visualization and Mathematics.

[3] Julius, D., Kraevoy, V., and Sheffer, A., 2005. "D-charts: quasi-developable mesh segmentation". *Computer Graphics Forum,* **24**, pp. 581–590.

[4] Liu, Y., Pottmann, H., Wallner, J., Yang, Y.-L., and Wang, W., 2006. "Geometric modeling with conical meshes and developable surfaces". *ACM Transactions on Graphics,* **25**(3), pp. 681–689.

[5] Wang, C., and Tang, K., 2004. "Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches". *The Visual Computer,* **20**, pp. 521–539.

[6] Desbrun, M., Meyer, M., and Alliez, P., 2002. "Intrinsic parameterizations of surface meshes". *Computer*

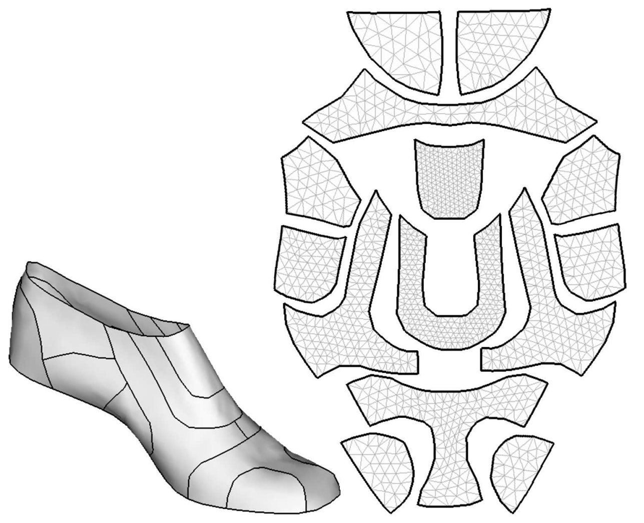
Fig. 13. The 2D layout of leather patterns for making the shoe shown in the left, where the patterns are generated by [16] from the patches shown in Fig.12(c).

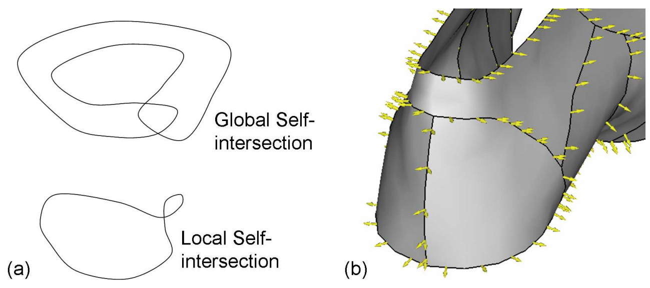

Fig. 14. Discussion: (a) the computed optimal planar boundary may globally self-intersected (top) although the local self-intersection (bottom) has been prevented, and (b) tangential continuity cannot always be preserved on the resultant patches (comparing normal vectors on the boundary curves at heel and other regions).

Graphics Forum, **21**(3), pp. 209–218.

[7] Karni, Z., Gotsman, C., and Gortler, S., 2005. "Free-boundary linear parameterization of 3d meshes in the presence of constraints". In Proceedings of Shape Modeling International, pp. 266–275.

[8] Lee, Y., Kim, H.-S., and Lee, S., 2002. "Mesh parameterization with a virtual boundary". *Computers & Graphics,* **26**, pp. 677–686.

[9] Lévy, B., Petitjean, S., Ray, N., and Maillot, J., 2002. "Least squares conformal maps for automatic texture atlas generation". *ACM Transactions on Graphics,* **21**(3), pp. 362–371.

[10] Sheffer, A., Lévy, B., Mogilnitsky, M., and Bogomjakov, A., 2005. "Abf++: fast and robust angle based flattening". *ACM Transactions on Graphics,* **24**(2), pp. 311–330.

[11] Sheffer, A., and de Sturler, E., 2001. "Parameterization of faceted surfaces for meshing using angle based flattening". *Engineering with Computers,* **17**(3), pp. 326–337.

[12] Azariadis, P., and Aspragathos, N., 1997. "Design of plane development of doubly curved surface". *Computer-Aided Design,* **29**, pp. 675–685.

[13] Aono, M., Breen, D., and Wozny, M., 2001. "Modeling methods for the design of 3d broadcloth composite parts". *Computer-Aided Design,* **33**, pp. 989–1007.

[14] Aono, M., Breen, D., and Wozny, M., 1994. "Fitting a woven-cloth model to a curved surface: mapping algorithms". *Computer-Aided Design,* **26**, pp. 278–292.

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

[16] Wang, C., Smith, S., and Yuen, M., 2002. "Surface flattening based on energy model". *Computer-Aided Design,* **34**(11), pp. 823–833.

[17] Wang, C., Tang, K., and Yeung, B., 2005. "Freeform surface flattening by fitting a woven mesh model". *Computer-Aided Design,* **37**, pp. 799–814.

[18] Decaudin, P., Julius, D., Wither, J., Boissieux, L., Sheffer, A., and Cani, M.-P., 2005. "Virtual garments: a fully geometric approach for clothing design". *Computer Graphics Forum,* **25**(3), pp. 625–634.

[19] do Carmo, M., 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, NJ.

[20] Leopoldseder, S., and Pottmann, H., 1998. "Approximation of developable surfaces with cone spline surfaces". *Computer-Aided Design,* **30**, pp. 571–582.

[21] Pottmann, H., and Wallner, J., 1999. "Approximation algorithms for developable surfaces". *Computer Aided Geometric Design,* **16**, pp. 539–5562.

[22] Chu, C., and Séquin, C., 2002. "Developable bézier patches: properties and design". *Computer-Aided Design,* **34**(7), pp. 511–527.

[23] Chen, H.-Y., Lee, I.-K., Leopoldseder, S., Pottmann, H., Randrup, T., and Wallner, J., 1999. "On surface approximation using developable surfaces". *Graphical Models and Image Processing,* **61**, pp. 110–124.

[24] Peternell, M., 2004. "Recognition and reconstruction of developable surfaces from point clouds". In Proceedings of Geometric Modeling and Processing 2004, pp. 301–310.

[25] Peternell, M., and Steiner, T., 2004. "Reconstruction of piecewise planar objects from point clouds". *Computer-Aided Design,* **36**, pp. 333–342.

[26] Wang, C., Wang, Y., and Yuen, M., 2004. "On increasing the developability of a trimmed nurbs surface". *Engineering with Computers,* **20**(1), pp. 54–64.

[27] Mortenson, M., 1997. *Geometric Modeling (2nd Edition)*. Wiley, New York.

[28] Madsen, K., Nielsen, H., and Tingleff, O., 2004. Optimization with constraints. Lecture Notes.

[29] Chew, L., 1987. "Constrained delaunay triangulations". In Proceedings of the third annual symposium on Computational geometry table of contents, pp. 215–222.

[30] Botsch, M., and Kobbelt, L., 2005. "Real-time shape editing using radial basis functions". *Computer Graphics Forum,* **24**(3), pp. 611–621.

[31] Turk, G., and O'Brien, J., 2002. "Modelling with implicit surfaces that interpolate". *ACM Transactions on Graphics,* **21**(4), pp. 855–873.

[32] Yngve, G., and Turk, G., 2002. "Robust creation of implicit surfaces from polygonal meshes". *IEEE Transactions on Visualization and Computer Graphics,* **8**(4), pp. 346–359.

[33] Sorkine, O., 2006. "Differential representations for mesh processing". *Computer Graphics Forum,* **25**(4).

[34] Chen, D., Cohen-Or, D., Sorkine, O., and Toledo, S., 2005. "Algebraic analysis of high-pass quantization". *ACM Transactions on Graphics,* **24**(4), pp. 1259–1282.

[35] Sorkine, O., Cohen-Or, D., Irony, D., and Toledo, S., 2005. "Geometry-aware bases for shape approximation". *IEEE Transactions on Visualization and Computer Graphics,* **11**(2), pp. 171–180.

[36] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rossl, C., and Seidel, H.-P., 2004. "Laplacian surface editing". In Eurographics/ACM SIGGRAPH Symposium on Geometry Processing 2004, pp. 179–188.

[37] Sorkine, O., and Cohen-Or, D., 2004. "Least-squares meshes". In Proceedings of Shape Modeling International 2004, pp. 191–199.

[38] Nocedal, J., and Wright, S., 1999. *Numerical Optimization*. Springer-Verlag.

[39] Li, S., Demmel, J., and Gilbert, J., 2006. SuperLU. http://crd.lbl.gov/ xiaoye/SuperLU/, February.

[40] Kobbelt, L., 2000. "$\sqrt{3}$-subdivision". In Proceedings of SIGGRAPH 2000, pp. 103–112.