A Least-norm Approach to Flattenable Mesh Surface Processing

Charlie C.L. Wang*
Department of Mechanical and Automation Engineering
The Chinese University of Hong Kong

ABSTRACT

Following the definition of developable surface in differential geometry, the flattenable mesh surface, a special type of piecewise-linear surface, inherits the good property of developable surface about having an isometric map from its 3D shape to a corresponding planar region. Different from the developable surfaces, a flattenable mesh surface is more flexible to model objects with complex shapes (e.g., cramped paper or warped leather with wrinkles). Modelling a flattenable mesh from a given input mesh surface can be completed under a constrained nonlinear optimization framework. In this paper, we reformulate the problem in terms of estimation error. Therefore, the shape of a flattenable mesh can be computed by the least-norm solutions faster. Moreover, the method for adding shape constraints to the modelling of flattenable mesh surfaces has been exploited. We show that the proposed method can compute flattenable mesh surfaces from input piecewise linear surfaces successfully and efficiently.

Index Terms: I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations—Physically based modeling; J.6 [COMPUTER-AIDED ENGINEERING]: Computer-aided design (CAD)—Computer-aided design (CAD)

1 INTRODUCTION

In sheet manufacturing industries, the products are fabricated from two-dimensional patterns of sheet materials (e.g., metal in ship industry, fabric in apparel industry and toy industry, and leather in shoe industry and furniture industry). The final products are fabricated by warping and stitching 2D patterns together. The traditional design process in these industries is conducted in a trial-and-error manner. A designer will draft 2D pieces on a paper and then make a prototype to check whether the fitting is good. If the result is not satisfactory, the designer needs to modify the patterns by his experience and make another prototype. The prototyping and the modification steps will be applied repeatedly, which is very inefficient. Designers in these industries wish to have a geometric modelling tool to model the products by surfaces that can be flattened into 2D pieces without stretching, i.e., holding an isometric map to some 2D regions. In the rest of the paper, we simply call it the *stretch-free flattening property*. The well-known *developable surface* in differential geometry [10] inherits such an elegant property. The form of developable surfaces could be planes, generalized cylinders, conical surfaces (away from the apex), or tangent developable surfaces. However, the shape of products in practice could be more complex (e.g., as shown in Fig.1), which can hardly be modelled by the conventional developable surfaces. The approach presented in this paper provides an efficient method to deform a user-defined 3D piecewise linear surface S into a new mesh surface M that will induce minimized distortion error in flattening.

*e-mail:cwang@mae.cuhk.edu.hk

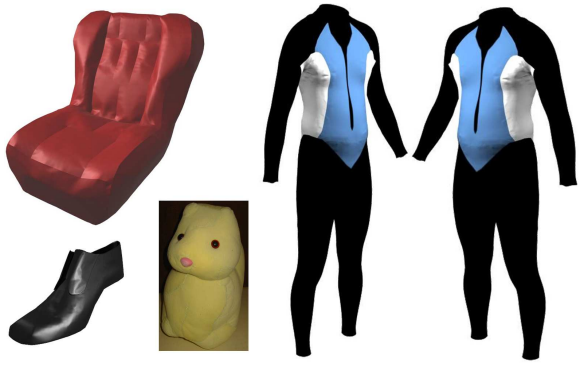

Figure 1: Products made from 2D sheets. Their shape can hardly be modelled by the conventional developable surfaces.

1.1 Problem definition

Without lose of generality, we assume that the user-defined 3D piecewise linear surface S is with the disk-like topology. This is because only the surface patches with such a topology can easily be fabricated from 2D pieces. For example, without adding a cut between the two loops, generalized cylinders cannot be made from a piece of sheet material. Here, there are two general requirements for the mesh surface M deformed from S:

- M is a mesh surface that shows the stretch-free flattening property;
- The difference between M and S is minimized.

A mesh surface satisfying the stretch-free flattening property is named as *a flattenable mesh surface*. The difference between two mesh surface M and S can be measured by some shape error (e.g., [21]). Therefore, the problem we are going to solve is to find a flattenable mesh surface M in $\Re^3$ to approximate the input piecewise linear surface S. Note that this is different from mesh surface parameterization [11] in computer graphics, where the mesh is computed in $\Re^2$ so that the shape deformation in 3D cannot be explicitly controlled.

1.2 Related work

The mesh processing method for flattenable mesh surface relates to the developable surface in differential geometry [10]. In general, a surface is developable if and only if the Gaussian curvature of every surface point is zero. To satisfy this, there are many approaches modelling [7, 16, 28] or approximating [6, 26, 27] 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.

Julius et al. developed an algorithm in [12] to separate a given model into quasi-conical proxies. Based on a similar idea, in [8] the authors processed a given mesh surface instead of segmenting it, where a deformation process is applied to let the surface locally approximate a conical surface. It is a sufficient (but not necessary) condition for a conical mesh surface to be flattenable. In other

words, there are many flattenable mesh surfaces that are not conical. Wang and Tang [39] adopted the discrete definition of Gaussian curvature in [23] to process the given mesh surface through a constrained optimization to make it more flattenable. This is recently further developed into the Flattenable Laplacian (FL) mesh processing method in [37], which has a more stable computation. A least-norm solution based flattenable mesh processing approach will be exploited in this paper to speed up the computation. Another related work is the PQ meshes presented by Liu et al. in [21]. The computation of PQ meshes is also under the constrained optimization framework; however, the developable surface modelled in [21] is with relatively simple shape. A fast approach to compute flattenable meshes with complex shapes is needed. In [29], developable (discrete) surfaces are generated by triangulating 3D boundary curves, which is different from our purpose.

The parameterization of a given three-dimensional surface computes its corresponding 2D parametric domain, usually via surface flattening. Therefore, a surface parameterization always introduces distortion in either angles or areas. All parameterization approaches in literature focus on how to minimize the distortions in some sense (see [11] for a detail review). Among the methods of mesh parameterization, only a few schemes [9, 14, 15, 18, 30, 31, 41] generate a planar domain with a free boundary so that it can be employed to compute the shape of 2D patterns. The recent linear ABF presented in [41] balances the speed and the quality of parameterization excellently. We borrow their idea of reformulating constrained optimization to a least-norm numerical solution in this paper. In the area of computer-aided design, the surface flattening for pattern design has been studied in various industries (cf. [1, 2, 3, 22, 24, 38, 40]). Nevertheless, neither mesh parameterization nor mesh flattening approaches provides a tool for modelling flattenable freeform mesh surfaces in $\Re^3$.

1.3 Main result

Based on the method [37] for computing flattenable mesh surfaces through constrained optimization, we reformulate the problem in terms of estimation error. Therefore, the shape of a flattenable mesh can be computed by the least-norm solutions of the estimation error with a faster speed. The resulting algorithm is faster than the method in [37]. Moreover, the method for adding constraints to the modelling of flattenable mesh surfaces has been exploited in this new approach. We show that the proposed method can successfully compute flattenable mesh surfaces from the input piecewise linear surfaces.

The rest of the paper is organized as follows. After briefly describing the method for modelling flattenable mesh surfaces in section 2, the problem is reformulated in terms of estimation error and linearized in section 3. The least-norm solution of the reformulated problem is presented in section 4. Also, in order to let the computation be more stable, the least-square scheme is adopted to update the position of vertices. Lastly, experimental results are given in section 5 together with the applications in freeform design of wetsuit and furniture.

2 Flattenable mesh surface

The properties about modelling flattenable mesh surfaces [37] are briefly summarized as follows. To be simple, the definition of flattenable mesh surfaces refers to triangular mesh surface only.

- A flattenable mesh surface M is a triangular mesh surface patch which can be flattened into a two-dimensional region D without stretching any triangle on it.
- For any 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 (see the illustration shown in Fig.2).

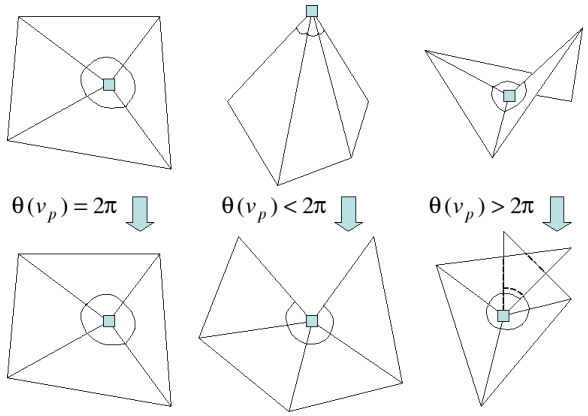

Figure 2: Stretch-free flattening of triangles around a vertex in different cases.

- For a triangular mesh patch M in $\Re^3$ with only one boundary loop, if the summed inner angles at all its inner vertices are 2π, M can be deformed into a patch D in $\Re^2$ without stretching any triangle (Proof can be found in [37]).
- When computing a flattenable mesh surface M_F from a given surface M, M_F should approximate the shape of M.

Let M be a given triangular mesh with the graph $G = (V, E)$ where V and E are the set of vertices and edges respectively. The computation of M_F from M is formulated as a constrained optimization problem [37]

$$\min_{p \in V_{int}} J_{pos} \quad s.t. \;\; \theta(\mathbf{v}_p) \equiv 2\pi \tag{1}$$

where V_{int} is the collection of interior vertices. J_{pos} is defined as $J_{pos} = \sum_{p \in V_{int}} \|\mathbf{v}_p - \mathbf{v}_p^0\|^2$ with $\mathbf{v}_p^0$ being the closest position of $\mathbf{v}_p$ on the given surface M (or simply being the original position of $\mathbf{v}_p$). Note that only interior vertices are allowed to be moved, and the boundary vertices are fixed to preserve the G^0 continuity with other assembled patches.

When solving Eq.(1) under a constrained optimization framework (e.g., the Lagrange multiplier method), the position objective function J_{pos} enforces the new position $\mathbf{v}_p$ of every vertex be close to their original position $\mathbf{v}_p^0$ on M. This prevents the deformation of M into a flattenable mesh surface in some cases. Therefore, the convergency of computation is slow in [37]. To overcome this shortcoming, we reformulate the minimization problem in Eq.(1) to a minimum-norm problem in terms of estimation error as follows.

3 Reformulation

After linearizing the constraints in Eq.(1), the flattenable mesh modelling problem can be converted into a minimum-norm problem and solved by a least-norm solution. As discussed in [41], carefully selecting alternative variables could make the linearization more accurate so that the computation converges faster than the Newton steps. Here, we choose the update vectors of interior vertices' position as variables.

First of all, we assume that the updated positions $\mathbf{v}_p$ of a vertex be an ideal position satisfying the flattenable constraint (i.e., the 2π constraint on its summed inner angles). The relationship between $\mathbf{v}_p$ and $\mathbf{v}_p^0$ can be presented by

$$\mathbf{v}_p = \mathbf{v}_p^0 + \mathbf{d}_p \tag{2}$$

The current position $\mathbf{v}_p^0$ can be considered as an initial guess, and $\mathbf{d}_p$ represents the estimation error. Therefore, the problem in Eq.(1)

can be rephrased as: we need to determine a set of minimal update vectors $\mathbf{d}_p$ on interior vertices so that the flattenable constraint is satisfied on every interior vertex. Let n_{int} be the number of vertices in V_{int}, there are in total $3n_{int}$ variables to be computed as each $\mathbf{d}_p$ has three components. However, only n_{int} flattenable constraints are given. Therefore, this is an underdetermined system of equations. Among the infinitely many solutions, we wish to obtain the solution with minimal-norm of estimation error.

The flattenable constraint defined on every interior vertex $\mathbf{v}_p$ is

$$\theta(\mathbf{v}_p) = 2\pi \tag{3}$$

where the value of $\theta(\mathbf{v}_p)$ is in terms of positions of $\mathbf{v}_p$ and $\mathbf{v}_q$ ($\forall q \in N(p)$). Here, $N(p)$ denotes the one-ring neighboring vertices of $\mathbf{v}_p$. To linearize the nonlinear expression in Eq.(3), the Taylor expansion for multi-variables is applied to $\theta(\mathbf{v}_p)$ as

$$\theta(\mathbf{v}_p) = \theta(\mathbf{v}_p^0) + \nabla\theta(\mathbf{v}_p^0)^T(\mathbf{v}_p - \mathbf{v}_p^0, \mathbf{v}_q - \mathbf{v}_q^0)^T + \ldots \tag{4}$$

by neglecting the high-order terms. More specifically, the nonlinear flattenable constraint in Eq.(3) can be rewritten as

$$\theta_p(\mathbf{v}_p^0)\cdot\mathbf{d}_p + \sum_{q\in N(p)} \theta_q(\mathbf{v}_p^0)\cdot\mathbf{d}_q \approx 2\pi - \theta(\mathbf{v}_p^0) \tag{5}$$

where $\theta_p(\mathbf{v}_p^0)$ and $\theta_q(\mathbf{v}_p^0)$ are the derivatives of $\theta(\mathbf{v}_p)$ in terms of $\mathbf{v}_p$ and $\mathbf{v}_q$ at current positions. Formulas for the gradients θ_p and θ_q have been derived in [37], which can be efficiently calculated. The error introduced by this approximation is quadratical to the estimation error $\mathbf{d}_p$ and $\mathbf{d}_q$, thus the computation converges fast.

4 Flattenable Mesh Surface Processing

The reformulation in previous section leads to the method of flattenable mesh processing here. By linearizing the flattenable constraint into Eq.(5), the set of flattenable constraints is converted into n_{int} linear equations with only $3n_{int}$ unknown variables – an underdetermined linear equations system. Letting $\mathbf{d}$ denote a vector with $3n_{int}$ components formed by the n_{int} update vectors $\mathbf{d}_p$, together with Eq.(5), the flattenable mesh processing problem becomes

$$\min \|\mathbf{d}\|^2 \quad s.t.\ \mathbf{A}\mathbf{d} = \mathbf{b} \tag{6}$$

where $\mathbf{A}\mathbf{d} = \mathbf{b}$ is from Eq.(5). Clearly, it now becomes a least-norm problem. The matrix $\mathbf{A}$ has full rank as the flattenable constraints are independent. From the literature of mathematics, we know that for a full rank coefficient matrix $\mathbf{A}$, the above least-norm problem has a unique solution (cf. [36])

$$\mathbf{d} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b} \tag{7}$$

The value of $\mathbf{d}$ can be solved by finding a solution to the normal equation

$$(\mathbf{A}\mathbf{A}^T)\mathbf{x} = \mathbf{b} \tag{8}$$

followed by a substitution that $\mathbf{d} = \mathbf{A}^T\mathbf{x}$. The matrix $\mathbf{A}$ is sparse, so the solution in Eq.(8) can be efficiently solved. In our implementation, the *SuperLU* solver [19] packaged by the *OpenNL* interface [17] is employed.

The squared norm $\|\mathbf{d}\|^2$ of the vector $\mathbf{d}$ is exactly the shape approximation term J_{pos} in Eq.(1). Therefore, when the current positions $\mathbf{v}_p^0$ of vertices are close to the optimal positions $\mathbf{v}_p$ (i.e., the approximation error introduced in Eq.(5) is small), the optimal solution can directly be obtained. Although it is not easy to find a good initial guess, the estimation error will become smaller and smaller if the positions of vertices are repeatedly updated by Eq.(2) after computing the least-norm solution of $\mathbf{d}_p$. The newly updated positions will be adopted as the current position in the next round of evaluation. The outline of the flattenable mesh processing method is listed below.

Algorithm 1 Flattenable Mesh Processing

1: **repeat**
2: Setup the linearized constraints of Eq.(5) in $\mathbf{A}^T$;
3: Compute $\mathbf{x}$ by Eq.(8);
4: Compute the update vectors $\mathbf{d} = \mathbf{A}^T\mathbf{x}$;
5: Update the position of each vertex $\mathbf{v}_p \in V_{int}$;
6: **until** the terminal condition is satisfied.

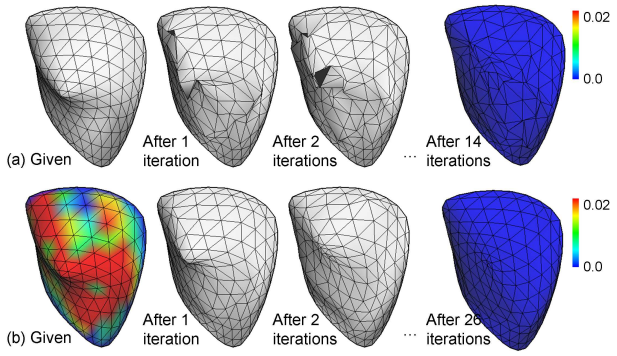

Figure 3: Example of flattenable mesh processing: (a) the progressive results by the direct update of vertex positions – the computation even diverges in some examples, and (b) the results by the least-square update. Colors represent the difference between each vertex's summed inner angle and 2π.

The terminal condition consists of two parts. We stop the iteration if 1) $\|\mathbf{d}\|^2 < \varepsilon$ or 2) the iteration has been repeated for more than 50 times. $\varepsilon = \max\{n_{int}\times 10^{-8}, 10^{-5}\}$ is usually chosen in our implementation.

4.1 Least-square update of shape

The above outline of flattenable mesh surface processing works well when the given surface M is nearly flattenable. However, if M is far from flattenable, the movement of vertices during the above flattenable mesh processing routine may break the regularity of vertex distribution on the surface. One example is shown in Fig.3(a). There are two causes of this instability: 1) the formulation of flattenable mesh processing does not consider the distribution of vertices, and 2) the linearized flattenable constraints in Eq.(5) are only in the first order of accuracy. As no consideration is given for the distribution of vertices, they are freely moved during the mesh processing. This gives more degree-of-freedom to process mesh, but it easily leads to instability when M is far from flattenable. Furthermore, the first order approximation of flattenable constraints may drive the vertices to some inaccurate places when M is far from flattenable.

In order to improve the robustness of the flattenable mesh processing, we update the shape of mesh surface through a least-square solution, which is stimulated by the Laplacian mesh editing technique (cf. the state-of-the-art report in [32] and its relevant technical papers [5, 33, 34, 35]). Let $\mathbf{v}_p^* = \mathbf{v}_p^0 + \mathbf{d}_p$ where $\mathbf{d}_p$ is the update vector determined in Eq.(7), the updated position of vertices are computed by

$$\begin{pmatrix} \mathbf{L} \\ \mathbf{I} \end{pmatrix}\mathbf{v} = \begin{pmatrix} \mathbf{g} \\ \mathbf{v}^* \end{pmatrix} \tag{9}$$

where $\mathbf{L}$ denotes the discrete graph Laplacian operator, $\mathbf{v}$ is a vector consisting of all vertices in V_{int}, $\mathbf{v}^*$ is a collection of their corresponding $\mathbf{v}_p^*$, and $\mathbf{g}$ is a vector containing those static boundary

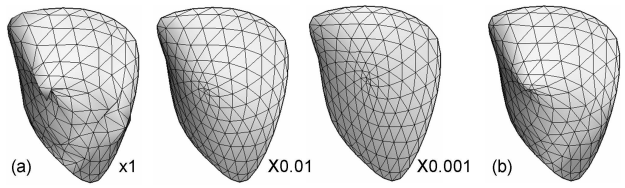

Figure 4: The influence of mixed area on the results of least-square update: (a) without normalization, the least-square update is effected by the dimension of input surface (when scaling the surface by 0.01 or 0.001), and (b) the consistent result is given by using the normalized mixed area.

vertices. Solving the above least-square problem is in fact

$$\min \{ \sum_{p \in V_{int}} \|\mathbf{L}(\mathbf{v}_p)\|^2 + \sum_{p \in V_{int}} \|\mathbf{v}_p - \mathbf{v}_p^*\|^2 \} \tag{10}$$

We can choose the uniform Laplacian operator for $\mathbf{L}(\cdots)$, or choose the cotan weighted discrete Laplacian operator proposed in [23] to balance the irregular meshes, which is as

$$\mathbf{L}(\mathbf{v}_p) = \frac{1}{2\bar{A}(\mathbf{v}_p)} \sum_{q \in N(p)} (\cot \alpha_{pq} + \cot \beta_{pq})(\mathbf{v}_p - \mathbf{v}_q) \tag{11}$$

with α_{pq} and β_{pq} the opposite angles in two triangles adjacent to edge $\mathbf{v}_p\mathbf{v}_q$. The static boundary vertices in $\mathbf{L}(\mathbf{v}_p)$ are moved to the right-hand side vector $\mathbf{g}$. Note that here $\bar{A}(\mathbf{v}_p)$ is a normalized mixed area at $\mathbf{v}_p$ as

$$\bar{A}(\mathbf{v}_p) = A_m(\mathbf{v}_p) / (\frac{1}{n_{int}} \sum_{p \in V_{int}} A_m(\mathbf{v}_p)). \tag{12}$$

The formula for $A_m(\mathbf{v}_p)$ has been given in [23]. Without this normalization, the value of coefficients in $\mathbf{L}(\mathbf{v}_p)$ will be much greater if the dimension of mesh surface is small, and vice versa. Once the coefficients are large which means the weight on the Laplacian term in Eq.(10) is great, $\mathbf{v}_p$ will hardly converge to the position $\mathbf{v}_p^*$. On the other hand, if the coefficients are small, the Laplacian term in Eq.(10) becomes useless. Fig.4 illustrates the influence of normalization on mixed area for the mesh surface given in Fig.3, where the listed results are after the first iteration.

This least-square position update is adopted in step 5 of **Algorithm** *Flattenable Mesh Processing* with a fixed Laplacian matrix $\mathbf{L}$ computed at the beginning of iteration. However, the positions of vertices computed by Eq.(9) will never equal to $\mathbf{v}_p^*$. Therefore, when the mesh surface under processing becomes nearly flattenable, the positions are updated by moving $\mathbf{v}_p$ to $\mathbf{v}_p + \mathbf{d}_p$ directly. In our implementation, a hybrid condition is employed to change from the least-square update to the direct update – either 1) $\theta_{err}^i < 0.001$ or 2) $(\theta_{err}^{i-1} - \theta_{err}^i) < 0.01$ after 20 iteration steps.

$$\theta_{err}^i = \max\{|\theta(\mathbf{v}_p) - 2\pi|\} \tag{13}$$

defines the flattenable error after the ith iteration. One example by using this scheme to update positions has already been given in Fig.3(b). More examples will be given below.

4.2 Weights on vertices

The formulas of least-norm solution assume that all vertices are with the same importance during the flattenable mesh processing. However, we need to define different weights on vertices in practice. For example, the vertices adjacent to the boundary of a given surface are expected to have less movement than other interior vertices so that the tangential continuity across the boundary can be approximated. In the garment design applications, semantic features

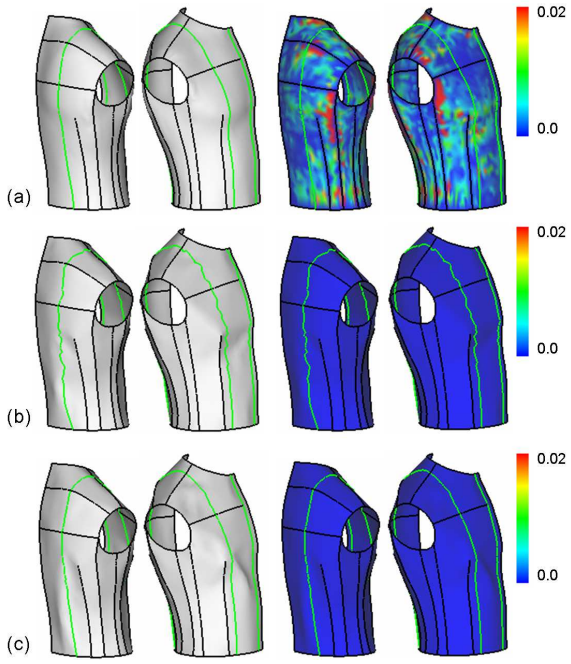

Figure 5: Example by using weights on vertices to preserve the shape of semantic feature curves: (a) the input surface for wetsuit with feature curves in green, (b) the flattenable mesh processing result without using weight on vertices ($\theta_{err} = 1.10 \times 10^{-5}$), (c) the result by adding higher weight on the vertices on feature curves ($\theta_{err} = 1.15 \times 10^{-5}$). Note that the uniform Laplacian is used in this test, the distortion of features in (b) will not be such significant if the cotan weighted Laplacian is employed.

are defined on some vertices. These vertices are often required to have much smaller movement than others to preserve the shape of feature curves (see Fig.5). All these can be implemented by introducing different weights on vertices.

Without loss of generality, let w_p denote the weights we wish to add on the vertex $\mathbf{v}_p$ where a greater w_p will lead to a smaller movement $\mathbf{d}_p$. By a simple change of variables

$$\mathbf{r}_p = w_p \mathbf{d}_p \tag{14}$$

we then compute the least-norm solution of $\mathbf{r}_p$ as

$$\min \|\mathbf{r}\|^2 \quad s.t. \ \mathbf{AWr} = \mathbf{b} \tag{15}$$

where $\mathbf{r}$ is the vector containing all $\mathbf{r}_p$ and $\mathbf{W}$ is a diagonal matrix with $1/w_p$ as the diagonal element. After solving $\mathbf{r}$ by

$$\mathbf{r} = (\mathbf{AW})^T((\mathbf{AW})(\mathbf{AW})^T)^{-1}\mathbf{b} \tag{16}$$

the value of $\mathbf{d}$ can be determined by $\mathbf{d} = \mathbf{Wr}$. Similarly, the least-square update is changed to

$$\begin{pmatrix} \mathbf{L} \\ \mathbf{W}^{-1} \end{pmatrix} \mathbf{v} = \begin{pmatrix} \mathbf{g} \\ \mathbf{W}^{-1}\mathbf{v}^* \end{pmatrix} \tag{17}$$

to let the vertices with larger weight to be less free.

The weight w_p should be carefully selected. Too small value will have no effect on the computational results, but too high weights can cause numerical problems. Choosing $w_p = 10$ for vertices to be constrained and $w_p = 1$ for other vertices works well for all tests in our experience. Fig.5 shows an example from garment industry for keeping the semantic feature curves by using different weights on vertices.

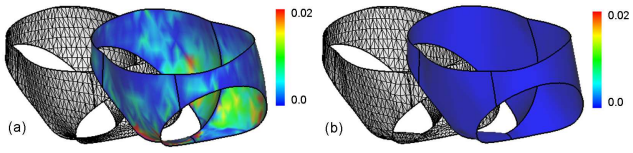

Figure 6: Underwear example: (a) the given mesh surface that the direct update by the least-norm solution diverges, and (b) the computation using the least-square update scheme with cotan weighted Laplacian converges ($\theta_{err} = 1.04 \times 10^{-5}$).

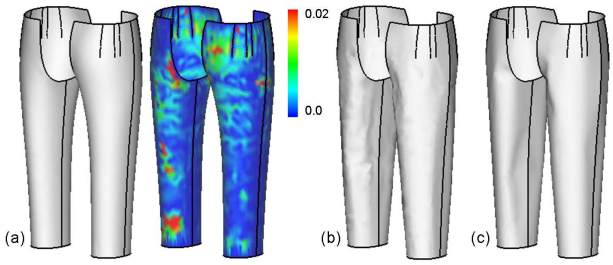

Figure 7: Pants example: (a) the given mesh surface, (b) the result by directly updating the positions has unwanted wrinkles ($\theta_{err} = 3.19 \times 10^{-5}$), and (c) the result using the least-square update scheme is very smooth ($\theta_{err} = 2.96 \times 10^{-6}$) and with more natural wrinkles.

Table 1: Mesh Size Statistics

Example	Face Number	Vertex Number
Patch (Fig.3)	250	142
Body (Fig.5)	2,122	2,388
Underwear (Fig.6)	1,122	1,271
Pants (Fig.7)	3,458	3,644

Table 2: Computational Statistics

Example	Method	Time$^+$	Steps	θ_{err}
Patch	FL meshes	0.75	43	2.11×10^{-4}
(Fig.3)	Least-norm	0.52	26	1.29×10^{-4}
Body	FL meshes	13.01	24-100	1.38×10^{-5}
(Fig.5)	Least-norm	8.95	26-31	1.15×10^{-5}
Underwear	FL meshes	4.13	37-100	3.29×10^{-3}
(Fig.6)	Least-norm	2.84	26-28	1.02×10^{-5}
Pants	FL meshes	37.92	100	1.31*
(Fig.7)	Least-norm	30.03	28-31	2.96×10^{-6}

* The computation has not converged yet.
$^+$ The time is reported in second, and all tests are evaluated on a PC with 3.0GHz Pentium IV CPU.

5 Results

Experimental tests and some applications are given in this section to demonstrate our least-norm approach to flattenable mesh processing. The discussion about limitation is also included.

5.1 Experimental tests and discussion

The first test has been shown in Fig.3 to compare the direct position update strategy and the least-square update scheme. The numerical computation of the least-square update scheme is more stable than directly updating the positions of vertices by the least-norm solution. However, the cost of such an improvement is the computing time, which is twofold – 1) each iteration takes more computing time and 2) usually a few more iteration steps are needed. This slow-down of computation makes up the inaccuracy introduced in the linear approximation of flattenable constraints. More specifically, without the least-square update, the computation of few examples diverges. For example, the underwear shown in Fig.6, using the direct update strategy on the pieces will lead to a diverged result; however, the computation using the least-square update succeeds. For the surfaces that the direct update scheme converges, the smoothness on the resultant surfaces is not as good as the results using the least-square update scheme. Fig.7 shows such an example. This is because the least-square update scheme in fact acts as a second-order filter that minimizes the thin-plate energy (cf. [25]).

The least-norm approach for flattenable mesh processing is an extension of the Flattanable Laplacian (FL) meshes [37], which solves the following problem

$$\begin{aligned} &\min \{\textstyle\sum_{p\in V_{int}} \|\mathbf{L}(\mathbf{v}_p)\|^2 + \sum_{p\in V_{int}} \|\mathbf{v}_p - \mathbf{v}_p^0\|^2\} \\ &s.t.\ \ \theta(\mathbf{v}_p) \equiv 2\pi \end{aligned} \tag{18}$$

by the sequential linearly constrained programming. In this paper, after introducing the least-norm solution and the least-square update scheme, both the convergency speed and the stability of computation have been improved. Tests have been given to verify the improvement. Comparisons between the FL mesh processing approach and the least-norm approach have been given in Table 1 and 2. From the statistics, it is easy to find that the least-norm approach always converges in fewer steps and in shorter time. The sparse linear equations in both are solved by the LU-decomposition solver [19]. Also, the final results from the least-norm approach have smaller flattenable error θ_{err} (which is defined in Eq.(13)) – i.e., the resultant meshes are more flattenable. The steps of iteration are not a fixed number because there are multiple pieces of patches in these examples.

After the flattenable mesh processing, all processed surface patches can be flattened into plane with almost no distortion. The comparison of flattening results for surfaces before vs. after processing has been given in Fig.8. Here, we conduct the least square conformal mapping method in [18] to compute planar meshes. To better illustrate the distortion introduced during flattening, the color maps for the homogeneity of distortion E_h and the aspect ratio E_r (see the detail computation formulas in [4]) are also included. Both E_h and E_r should be equal to zero for an isometric mapping.

Limitation The least-norm approach for flattenable mesh processing presented in this paper also has some limitations.

- First, although the least-square update scheme improves the stability of numerical computation, it however does not really converge to the solution for a flattenable mesh. This is the reason why we need to adopt the direct update strategy in the last few steps of computation. About the problem when we should stop the least-square update, the current method is based on experimental tests. A better condition for changing the update scheme needs to be studied.

- Many engineering applications wish to preserve the G^1 continuity across the boundary of assembled patches. In the current setting of the flattenable mesh processing, the constraint can only be added softly by using higher weights on the vertices adjacent to boundaries. There will have numerical problems if making G^1 continuity constraints hard by fixing these vertices. Moreover, the numerical problem occurs if choosing too great weights on these vertices. The G^1 continuity preservation is still a problem to be solved.

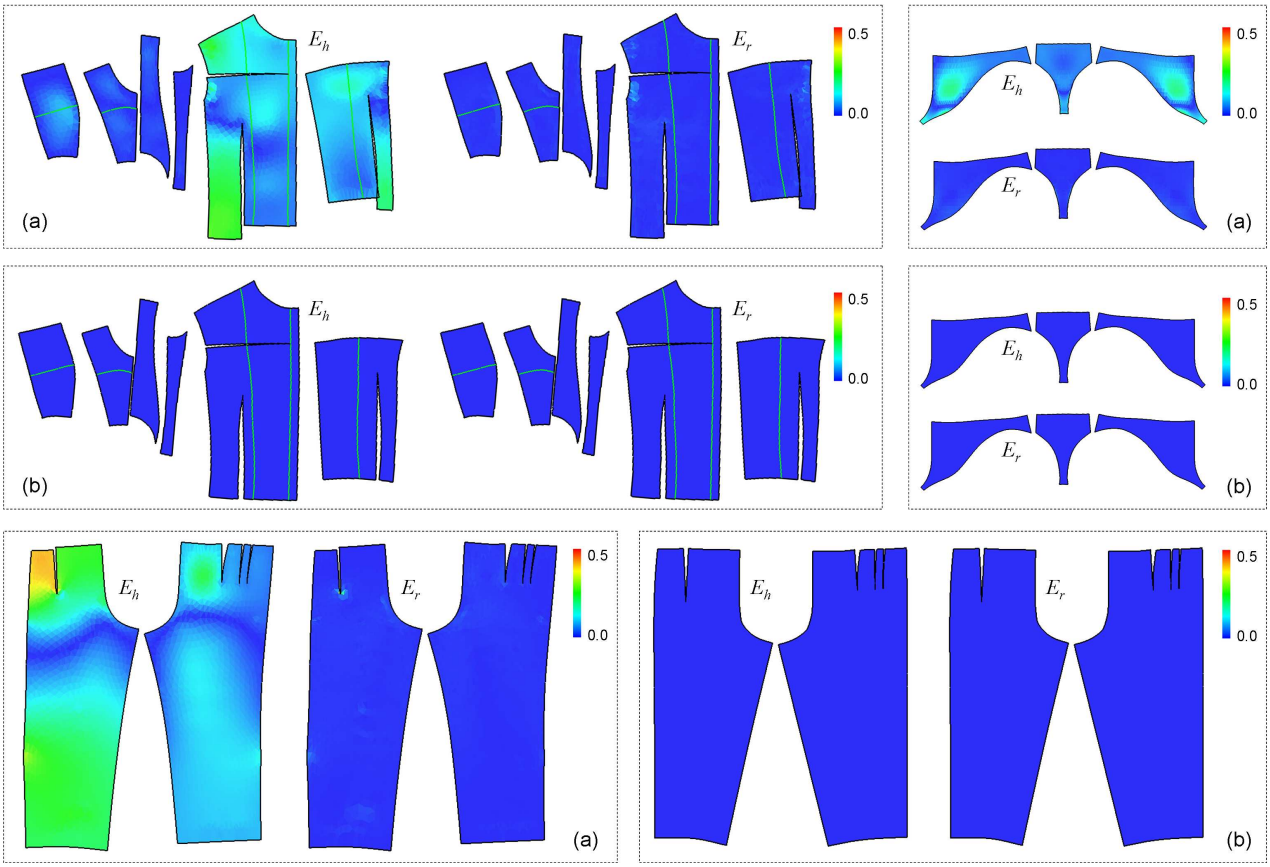

Figure 8: The comparison of planar pieces flattened from (a) the original meshes and (b) the processed flattenable meshes. The least-square conformable mapping [18] is adopted to generate the planar pieces, and the color maps illustrate the values of the homogeneity of distortion E_h and the aspect ratio E_r (cf. [4]).

- Lastly, by the Laplacian operator employed in the least-square update scheme, the processed mesh surfaces always shrink in some sense. This is fine in the applications where the 2D material pieces used to fabricate the final product can slightly be stretched. However, it will become a problem in the applications where no stretch is allowed. The control of shrinking or inflating effect (cf. [25]) needs to be considered.

These current limitations are the possible research topics in our future research.

5.2 Application I: wetsuit design

We have applied the flattenable mesh processing technique presented in this paper to the freeform wetsuit design application. An illustration is given in Fig.9. To freeform design wetsuit for a customer, the human body of the customer is firstly scanned and the mesh surface of human is reconstructed (see Fig.9(a)). After defining the freeform cutting lines, the 3D patterns of wetsuit are generated by trimming the mesh surface of human body (cf. [20]). The cutting result is given in Fig.9(b), and Fig.9(c) shows the color map for the flattenable error on the 3D patterns. After applying the flattenable mesh processing on the 3D patterns, its shape and flattenable errors are as shown in Fig.9(d). It is not difficult to find that, although all the patterns have become flattenable, the shoulder patterns cave in too much. If the final wetsuit is fabricated by these patterns, the user will feel too much pressure on his shoulder. To overcome this, we add a dart at the back of the shoulder patterns – see Fig.9(e). Then, the processed shape of shoulder patterns is more similar to its original shape (as shown in Fig.9(f)), so less pressure will be given on the shoulder of user by these patterns.

5.3 Application II: furniture design

The flattenable mesh processing technique presented in this paper can also be applied in the furniture design. For example, the sofa-chair shown in Fig.10. After defining the wire-frame, the mesh surfaces can be generated by the method presented in [13] and then be processed into flattenable mesh surfaces by our method. The stretch-free planar patterns generated from the flattenable mesh patches can then be easily determined by [18] (or even simpler greedy flattening). The final product can be fabricated by the leather sheets in the determined planar shape – the seam-allowance also needs to be considered.

6 Conclusions

In this paper, we reformulate the problem of modelling a flattenable mesh from a given input mesh surface in terms of estimation error so that the original nonlinear optimization problem is linearized and can be solved by a least-norm solution. Working together with a least-square update scheme, the computation of a flattenable mesh can be numerically stable and converge faster. Furthermore, the method for adding shape constraints by vertex weights has also been exploited. Several experimental tests have been conducted to demonstrate the performance of this approach, and the applications in garment industry and furniture industry have been shown.

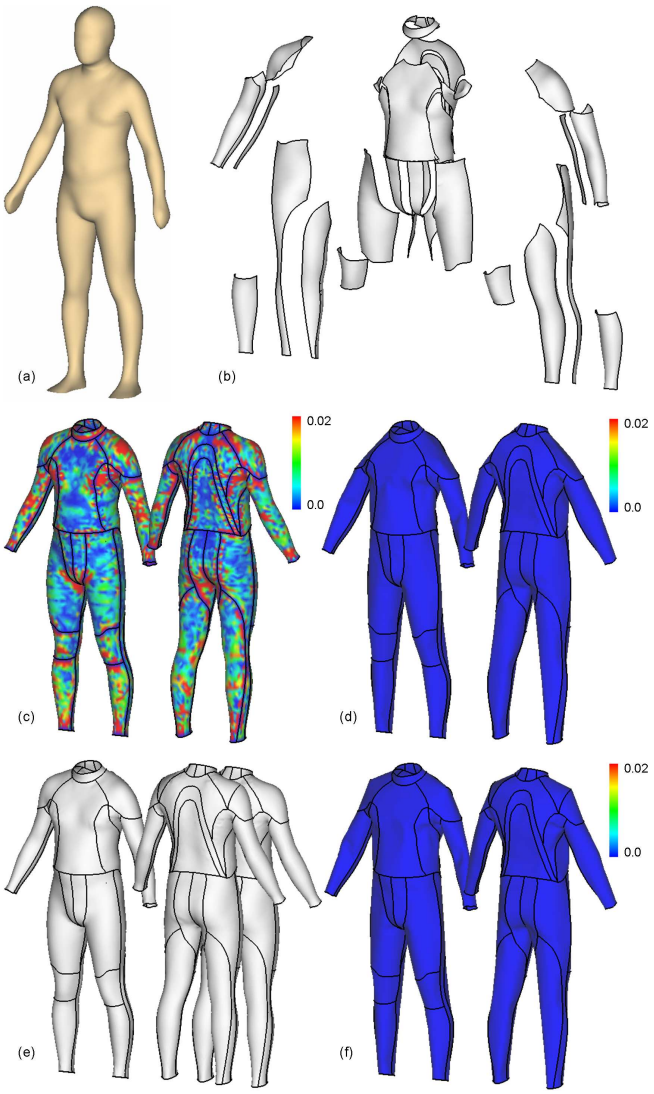

Figure 9: Application of wetsuit freeform design: (a) the given human model, (b) the 3D patterns for wetsuit cut out from the human body's surface, (c) color map for showing the flattenable error on 3D patterns, (d) the resultant patterns of flattenable mesh processing, (e) a modification with darts added at the back of shoulder patterns, and (f) the shape of flattenable mesh surfaces has been improved at the shoulder.

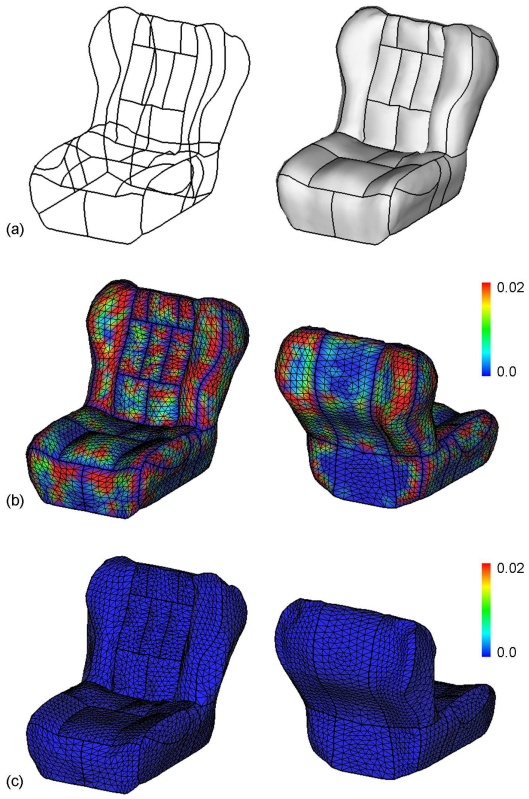

Figure 10: Application of furniture design: (a) the sofa-chair is designed by fitting freeform mesh surfaces onto the wire-frame, (b) the color map to show the flattenable error on the surface patches, and (c) the processed flattenable mesh surface patches.

ACKNOWLEDGEMENT

The work presented in this paper is partially supported by the Hong Kong Innovation and Technology Fund (ITF) ITS/026/07. The author would like to thank TPC (HK) Ltd for providing the data of wetsuit garment and human body.

REFERENCES

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

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

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

[4] P. Azariadis and N. Sapidis. Planar development of free-form surfaces: quality evaluation and visual inspection. *Computing*, 2004.

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

[6] H. Chen, I. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. On surface approximation using developable surfaces. *Graphical Models and Image Processing*, 61(2):110–124, 1999.

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

[8] P. Decaudin, D. Julius, J. Wither, L. Boissieux, A. Sheffer, and M.-P. Cani. Virtual garments: a fully geometric approach for clothing design. *Computer Graphics Forum*, 25(3):625–634, 2006.

[9] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2002.

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

[11] K. Hormann, A. Sheffer, B. Levy, M. Desbrun, and K. Zhou. *Mesh Parameterization: Theory and Practice (SIGGRAPH 2007 Course Notes)*. ACM, 2007.

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

[13] L. B. Kara, C. M. D'Eramo, and K. Shimada. Pen-based styling design of 3d geometry using concept sketches and template models. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, pages 149–160. 2006.

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

[15] Y. Lee, H.-S. Kim, and S. Lee. Mesh parameterization with a virtual

boundary. *Computers & Graphics*, 26(5):677–686, 2002.

[16] S. Leopoldseder and H. Pottmann. Approximation of developable surfaces with cone spline surfaces. *Computer-Aided Design*, 30(7):571–582, 1998.

[17] B. Lévy. *OpenNL*. http://www.loria.fr/ levy/software/, 2005.

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

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

[20] W. C. Li, B. Levy, and J.-C. Paul. Mesh editing with an embedded network of curves. In *Proceedings of Shape Modeling International*, pages 62–71. 2005.

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

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

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

[24] Y. Mori and T. Igarashi. Plushie: an interactive design system for plush toys. *ACM Transactions on Graphics*, 26(3):45, 2007.

[25] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Fibermesh: designing freeform surfaces with 3d curves. *ACM Transactions on Graphics*, 26(3):41, 2007.

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

[27] M. Peternell and T. Steiner. Reconstruction of piecewise planar objects from point clouds. *Computer-Aided Design*, 36(4):333–342, 2004.

[28] H. Pottmann and J. Wallner. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design*, 16(6):539–556, 1999.

[29] K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert. Developable surfaces from arbitrary sketched boundaries. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, pages 163–172. 2007.

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

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

[32] O. Sorkine. Differential representations for mesh processing. *Computer Graphics Forum*, 25(4):789–807, 2006.

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

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

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

[36] L. Vandenberghe. *Applied Numerical Computing (course-notes)*. http://www.ee.ucla.edu/ vandenbe/, 2007.

[37] C. Wang. Towards flattenable mesh surfaces. *Computer-Aided Design*, 2007.

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

[39] C. Wang and K. Tang. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer*, 20(8-9):521–539, 2004.

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

[41] R. Zayer, B. Lévy, and H.-P. Seidel. Linear angle based parameterization. In *Proceedings of Eurographics Symposium on Geometry Processing*. 2007.