Towards Flattenable Mesh Surfaces

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

Abstract

In many industries, products are constructed by assembled surface patches in $\Re^3$, where each patch is expected to have an isometric map to a corresponding region in $\Re^2$. The widely investigated developable surfaces in differential geometry show this property. However, the method to model a piecewise-linear surface with this characteristic is still under research. To distinguish from the continuous developable surface, we name them as flattenable mesh surfaces since a polygonal mesh has the isometric mapping property if it can be flattened into a two-dimensional sheet without stretching. In this paper, a novel flattenable mesh surface (Flattenable Laplacian mesh) is introduced and the relevant modelling tool is formulated. Moreover, for a given triangular mesh which is almost flattenable, a local perturbation approach is developed to improve its flattenability. The interference between the meshes under process and their nearby objects has been prevented in this local flattenable perturbation. Both the computations of Flattenable Laplacian meshes and the flattenable perturbation are based on the constrained optimization technology.

***Keywords*:** flattenable, freeform mesh surfaces, nonlinear subdivision, geometry processing, constrained optimization.

1. Introduction

The research presented in this paper is motivated by the development of geometric modelling systems for freeform products in sheet manufacturing industries, where the products are fabricated from two-dimensional patterns of sheet material (e.g, textile in apparel industry and leather in shoe industry). During fabrication, the 2D patterns are warped and stitched together to build the final product. Ideally, the warping and the stitching should be stretch-free since the stretch will produce some elastic energy in the final product which debases the fitness and creates material fatigue. 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 by the paper patterns to check whether the fitting is good. If the result is not satisfied, the designer needs to modify the patterns by his experiences and make another prototype. The prototyping and the modification steps will be repeatedly applied, which is very inefficient. Another more serious problem occurs when designing in 2D instead of 3D space – the product made from the patterns may not satisfy the desired 3D shape. Industrial designers find that designing in three-dimensional space is the best way to solve this problem. By a three-dimensional CAD systems (e.g., [47]), the product shapes are designed

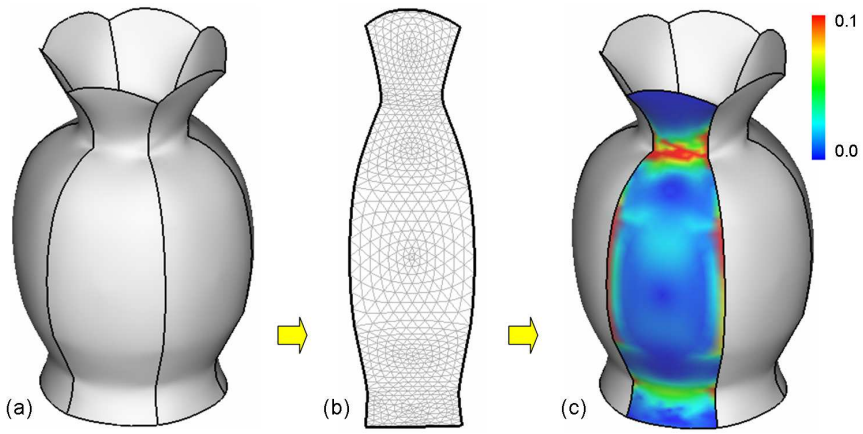

Figure 1: Stretches happen on mesh surfaces: (a) the original design, (b) the 2D pattern generated by [43], and (c) the color map for illustrating stretches in terms of elastic energy if the model is fabricated by the patterns in (b).

using polygonal meshes in 3D around the virtual body of customers. The modelling interface is more or less similar to the surface lofting function in state-of-the-art geometric modelling systems based on parametric surfaces, but is able to create models with more complex shapes and has more freedom on topological structures (e.g., the patch is not necessary to be 4-sided). After modelling the 3D surface patches for the shape of products, their corresponding 2D patterns for fabrication are computed through a surface flattening process. However, the flattening is stretch-free only if a surface mesh under flattening holds the isometric mapping property (i.e., the Geodesic distance between any two point on the 3D surface and the Euclidean distance between their corresponding planar points are equal [15]); otherwise, distortion will be introduced. Therefore, when using these 2D patterns to build up the designed 3D product shape, patterns need to be stretched to form the final 3D shape. The stretched region easily leads to material fatigue. For example, Fig.1 shows a color map to illustrate the elastic energy on the mesh surfaces of a 3D product, where the 2D patterns are computed by [43].

Studies for cloth simulation [4, 6, 7, 9, 41] show that cloth is an assembly of surface patches which strongly resist stretching but allow a lot of bending. Therefore, flattenable mesh surface is the best candidate for representing the products in apparel industry. At present, the cloth simulation approaches usually conduct a mass-spring system to mimic the physical characteristic of fabrics by choosing great stiffness coefficients for stretching springs but small coefficients for bending and shearing springs. Their works are based on the assumption that these surface patches are warped from 2D patterns without stretching. However, this is no more true for a system which allows users to design products directly in 3D, where the surfaces in general are not flattenable.

A lot of mesh parameterization approaches in the computer graphics literature [13, 19, 23, 25, 37, 38] and surface flattening approaches in the computer-aided design literature [1, 2, 3, 28, 43, 45] adopt various criteria to minimize the difference between the 3D surface patch and its corresponding 2D region. However, none of them provides the tools for directly modelling flattenable freeform mesh surfaces in $\Re^3$, which is the purpose of the research presented in this paper. We will firstly introduce a new type of mesh surfaces – Flattenable Laplacian meshes (in short, FL meshes), and then develop the modelling tool for FL meshes under the framework of constrained numerical optimization and the variational subdivision scheme. Furthermore, if a given polygonal mesh is almost flattenable, slight adjustments can improve its flattenability. For this, a new local flattenable perturbation

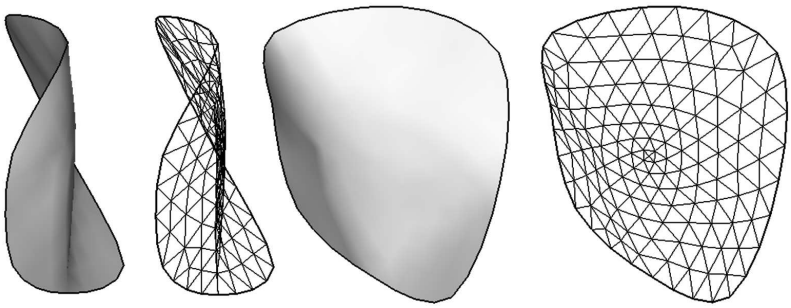

Figure 2: The surface with wrinkles that can be modelled by FL meshes but cannot be represented by a single developable ruled surfaces.

approach will also be developed in this paper. Different from previous approaches [12, 44] for the similar applications, this new local perturbation approach prevents the interference between the meshes under process and their nearby objects, where the computation is based on the constrained optimization with vertex positions as variables.

1.1. Related work

In this section, we review related work in the aspects of 1) developable parametric surfaces, 2) mesh processing for discrete developable surfaces, 3) mesh parameterization and flattening, and 4) Laplacian meshes.

Developable parametric surfaces

The study of flattenable mesh surfaces relates to the developable surface. From differential geometry [15], 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 which is the product of the maximum and minimum normal curvatures at a given point [15]. 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 [24, 33, 11] or approximating [10, 30, 32] a model with developable ruled surfaces (or ruled surfaces in other representations — e.g., B-spline or Bézier patches). However, it is difficult to use these approaches to model freeform surfaces (e.g., the surfaces with wrinkles as shown in Fig.2 and 7). 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 [46], the modelling ability for freeform objects by these approaches is still very limited.

Mesh processing for discrete developable surfaces

To compute flattenable mesh surface patches, Julius et al. developed an algorithm in [17] to separate a given model into quasi-conical proxies. Based on a similar idea, in [12] they processed a given mesh surface instead of segmenting it, where on every surface point a local conical surface is approximated to compute the expected normal vector at this point and then a deformation process is applied to make the surface follow the desired normal vectors — 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 [44], Wang and Tang adopted the definition of Gaussian curvature in discrete differential geometry [27] 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 [44] is akin to the approach in this paper, its converging speed is much slower than the computation of FL meshes. Besides, neither [12] nor [44] considers the interference between the surface and other objects nearby, which is usually the situation in the design of products worn by human bodies (e.g., clothes and shoes).

Recently, Liu et al. in [26] presented a novel PQ meshes — quad 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 [26] is still simple (i.e., with the shape similar to ruled surfaces). Our method presented in this paper computes flattenable meshes also by using the constrained optimization, however we can model the flattenable meshes with more complex shape (e.g., the surfaces in Fig.7).

Mesh parameterization and flattening

The work presented in this paper also relates to the research of mesh parameterization and mesh flattening. The parameterization of a given three-dimensional surface P computes its corresponding 2D parametric domain D, usually via surface flattening. An ideal surface flattening preserves the distances between any two points on P and D - mathematically named as *isometric mapping*. However, this property is not generally preserved. Therefore, a surface parameterization always introduces distortion in either angles or areas. All parameterization approaches in literature give strength on how to minimize the distortions in some sense, which has been mentioned in the detail review [16] by Floater and Hormann. In the literature of mesh parameterization, only a few parameterization schemes (e.g., [13, 19, 23, 25, 38, 37]) can generate a planar domain with a free boundary so that it can be used to determine the shape of 2D patterns. In the area of computer-aided design, the surface flattening for pattern design has been studied in various industries (cf. [1, 2, 3, 28, 43, 45]). Nevertheless, neither mesh parameterization nor mesh flattening approaches provides a tool for modelling flattenable freeform mesh surfaces in $\Re^3$.

Laplacian meshes

After being firstly applied in [40] to process mesh surfaces, Laplacian operators or Laplacian coordinates have been widely used in various geometric modelling applications for freeform objects (e.g., [14, 27]). Recently, Laplacian meshes have been systematically applied in the areas of mesh compression, mesh watermarking, mesh editing, shape interpolation, mesh metamorphosis and mesh editing (cf. the state-of-the-art report in [39] and its relevant technical papers [8, 35, 36, 34]). In our approach, the global Laplacian operator is applied to define the fairness term of surfaces, which works together with the flattenability term and the position term to model FL mesh surfaces.

1.2. Contributions and overview

This paper gives the following contributions.

- We introduce the Flattenable Laplacian (FL) mesh as a novel freeform surface representation which inherits the isometric mapping property;
- By subdividing FL meshes, a new modelling method for flattenable mesh surfaces has been developed under the variational subdivision scheme;
- Lastly, a new local flattenable perturbation approach is presented — the perturbation prevents the interferences between the surface under process and the nearby objects.

The rest of the paper is organized as follows. After presenting the concept of FL mesh and the computational method for its modelling in section 2, the FL mesh is utilized in the variational

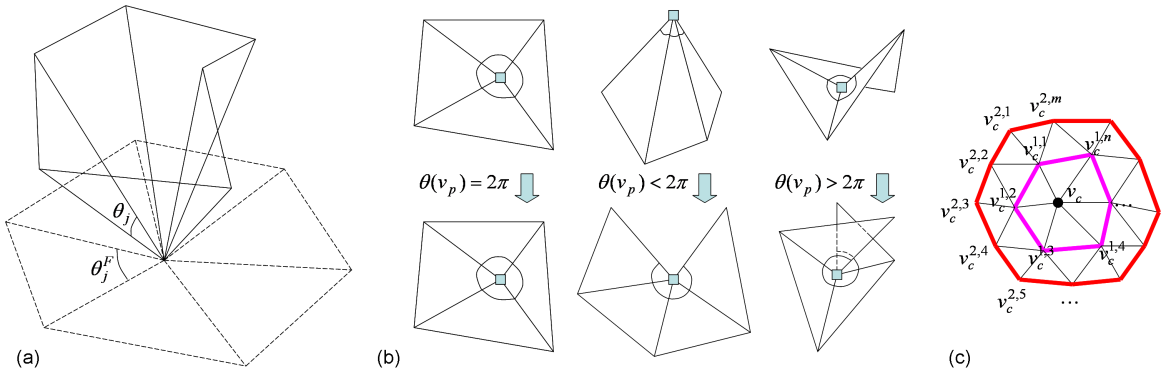

Figure 3: Illustration for flattenability: (a) the inner angles before and after flattening the triangles around a vertex, (b) the cases of $\theta(\mathbf{v}_p) = 2\pi$, $\theta(\mathbf{v}_p) < 2\pi$ and $\theta(\mathbf{v}_p) > 2\pi$, and (c) flattening on a disk-like patch can be given in a front advancing way.

subdivision scheme (in section 3) to model flattenable mesh surfaces with high quality. Section 4 describes the local flattenable perturbation approach which is developed to process the almost flattenable meshes. Computational statistics and some discussions are given in section 5. Finally, our paper ends with the conclusion section.

2. Flattenable Laplacian meshes

The concept of *Flattenable Laplacian meshes* (in short *FL meshes*) will be introduced in this section by inheriting advantages from both the flattenable meshes and the Laplacian meshes. The computational scheme for FL meshes will also be detailed.

2.1. Flattenable meshes

A flattenable mesh surface M is a polygonal mesh surface patch which can be flattened into a two-dimensional pattern D without stretching any polygon on it. More specifically, the flattening only involves the rotation and the transformation but no deformation. However, this is only a descriptive definition. Starting with flattenable vertices, we will give a more mathematical definition of flattenable mesh patches. Without loss of generality, only triangular mesh surfaces are referred in rest parts of the paper. Considering about an interior mesh vertex and its adjacent triangular faces as shown in Fig.3, we can easily conclude the following property.

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.

This property has been illustrated by Fig.3. If $\theta(\mathbf{v}_p) > 2\pi$, when flattening triangles around $\mathbf{v}_p$ without stretching, the triangles will generate overlap; while $\theta(\mathbf{v}_p) < 2\pi$, it yields gap. Therefore, we have the following definitions.

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

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

This is similar to the discrete form of Gaussian curvature [27]. The reason why we do not adopt the name *developable* (or *discrete developable*) as [17, 26, 44] 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.3, which is not differentiable, it is more appropriate to name it as flattenable/unflattenable rather than developable/undevelopable.

Property 2 A triangular *flattenable mesh patch* is with all its inner vertices flattenable.

Again, this is only a necessary condition but not sufficient. For example a triangular mesh surface like cylinder (with two boundary loops), even if all inner vertices on it are flattenable, we cannot flatten it into a two-dimensional region without stretching or inserting cuts. However, we observe the following geometry property.

Property 3 For a triangular mesh patch M in $\Re^3$ with the disk-like topology, if all its inner vertices are flattenable, it can be deformed into a patch D in $\Re^2$ without stretching any triangle.

Proof At first, if any vertex on M is not flattenable, by Fig.3(b) we know that the surface cannot be locally flattened into a plane without stretching.

Secondly, if all interior vertices on M are flattenable, can we find some case that M is not flattenable? As illustrated in Fig.3(c), starting from an interior point $\mathbf{v}_c$ (the dark one), we can flatten the triangles adjacent to $\mathbf{v}_c$ into $\Re^2$ by rotation and transformation only (without stretching) since $\theta(\mathbf{v}_c) = 2\pi$. Therefore, the shape of the pink front is formed. We sort the vertices on the pink front in the anti-clockwise order. For a vertex $\mathbf{v}_c^{1,i}$ located on the pink front, all triangles adjacent to it can be flattened into $\Re^2$ without stretching as that $\mathbf{v}_c^{1,i}$ is a flattenable vertex. The problem is that whether the location and orientation of planar triangles determined by the local flattening around $\mathbf{v}_c^{1,i}$ will conflict with the local flattening determined by other vertices. The answer is that if the topology of the front is disk-like, the triangles adjacent to $\mathbf{v}_c^{1,i}$ shall only be adjacent to its two neighboring vertices $\mathbf{v}_c^{1,i-1}$ and $\mathbf{v}_c^{1,i+1}$ but not any other vertex on the same front. Therefore, the triangles between the pink front and the red front can be flattened without stretching. Repeating this front advancing, as long as the front is in the disk-like topology, no stretch will be given during flattening. However, if the given mesh surface M is not in the disk-like topology, the disk-like topology cannot be always maintained on the fronts. For example, the cylinder or the cone, although every interior vertex on it is flattenable, some confliction will occur during the flattening of triangles in the front advancing.

Q.E.D.

2.2. Laplacian meshes

Let M be a given mesh patch with the graph $G = (V, E)$, where $V = 1, 2, ..., m$ is the set of vertices and E is the set of edges. $\mathbf{v}_i$ is used to denote the position of vertex i in $\Re^3$, and ∂V represents the set of vertices on the boundary of M. Similar to [39, 8, 35, 36, 34], the following equation defines the fairness or smoothness condition for $\mathbf{v}_i \in \Re^3$

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

where $N(\mathbf{v}_i)$ is the set of 1-ring neighboring vertices of $\mathbf{v}_i$ ($\mathbf{v}_i \in V \backslash \partial V$), and $|...|$ denotes the number of elements in a set. The linear system can be rewritten in the matrix form

$$L\mathbf{x} = 0, L\mathbf{y} = 0, L\mathbf{z} = 0, \tag{2}$$

where $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are the $n \times 1$ vectors containing the x, y and z coordinate of the vertices. The matrix L is known in [18, 40] as the Laplacian operator below

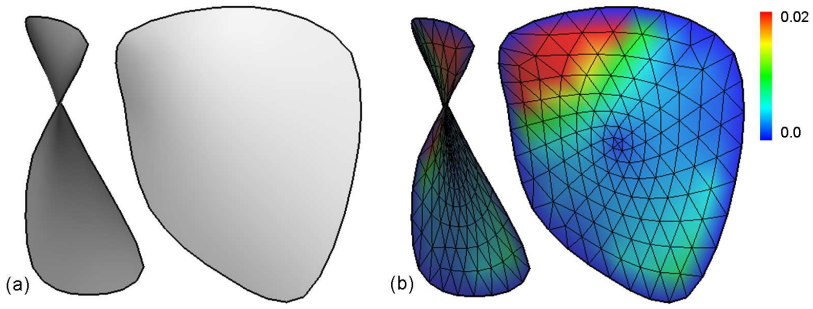

Figure 4: A Laplacian mesh with the same boundary and the same connectivity as the patch shown in Fig.2: (a) the surface and (b) the color map to illustrate the flattenability at vertices.

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

From [34], we know that the rank of L is $n-k$ where k is the number of connected components in the graph G. Therefore, for a mesh surface patch, the rank is $n-1$. When fixing vertices in ∂V, Eq.(2) gives the solution for the coordinates of inner vertices on a Laplacian mesh patch which minimize the linearized membrane energy $\int_\Omega \|\mathbf{v}_s\|^2 + \|\mathbf{v}_t\|^2 dsdt$ (ref.[14]) – the resultant mesh of Eq.(2) is a very smooth mesh surface which is uniquely defined by G and the position of vertices in ∂V.

A Laplacian mesh surface patch in general is not a flattenable mesh. For example, the Laplacian surface patch shown in Fig.4 is with the same boundary and the same connectivity as the surface in Fig.2; however, it is not flattenable — this can be easily found by the color map (Fig.4(b)) where colors represent the flattenability at vertices. Of course, surfaces like this cannot satisfy the applications in sheet manufacturing industries. Therefore, the FL meshes defined below are requested.

Definition 3 A *Flattenable Laplacian mesh* is a mesh surface patch which can be flattened into two-dimensional pieces without stretching, and at the meanwhile minimizes the fairness energy function defined by Laplacian operators.

2.3. On computing FL meshes

In this section, we develop the computational scheme for FL meshes which inherits advantages from both the flattenable mesh and the Laplacian mesh. In short, given a triangular mesh M, we need to recompute the positions of all inner vertices to generate a FL mesh M_{FL} which

- Being a flattenable mesh surface;
- Trying to be smooth (i.e., a Laplacian mesh);
- Trying to approximate the shape of M.

The three requirements can be formulated into a constrained optimization problem

$$\arg \min_{p \in V_{act}} \; w_1 J_{fair} + w_2 J_{pos} \quad subject\ to\ \ \theta(\mathbf{v}_p) \equiv 2\pi \tag{3}$$

where $V_{act} = V \setminus \partial V$, J_{fair} is the fairness term derived from the Laplacian mesh as

$$J_{fair} = \frac{1}{2} \sum_{p \in V_{act}} \phi(\mathbf{v}_p), \tag{4}$$

with $\phi(\mathbf{v}_p)$ a piecewise function $\phi(\mathbf{v}_p) = \|L\mathbf{v}_p\|^2$ only defined on the Voronoi area of $\mathbf{v}_p$, the gradient of J_{fair} with respect to $\mathbf{v}_p$ is $\frac{\partial J_{fair}}{\partial \mathbf{v}_p} = \mathbf{v}_p - \frac{1}{|N(\mathbf{v}_p)|}\sum_{k\in N(\mathbf{v}_p)} \mathbf{v}_k$, and J_{pos} is the position functional defined to minimize the difference between the new surface M_{FL} and the given surface M

$$J_{pos} = \frac{1}{2}\sum_{p\in V_{act}} \|\mathbf{v}_p - \mathbf{v}_p^0\|^2 \tag{5}$$

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 $\mathbf{v}_p$. Two coefficients w_1 and w_2 reflect the weights of functionals in fairness and position respectively. They are assigned by users. In all examples shown in this paper, we choose $w_1 = 1.0$ and $w_2 = 0.1$. In the fairness term, J_{fair}, we choose the uniform Laplacian but not the cotangent weighted Laplacian as [14, 27]. This is because that the uniform Laplacian also acts as a regularization term to re-distribute vertices while smoothing the given surface. Meanwhile, when cooperating with the variational subdivision scheme, most vertices are with regular valence. Therefore, it is appropriate to employ the uniform Laplacian here. In Eq.(3), we add the position constraints into the objective function (but not into the constraints set) — this is based on the reason that adding these constraints will disturb the constraints of flattenability during the numerical computation. The value of w_2 must be much smaller than w_1; otherwise, the numerical system may become instable.

With the Lagrange multiplier $\lambda = (\lambda_1, \lambda_2, ..., \lambda_n)$ ($n = |V_{act}|$), the constrained optimization problem defined in Eq.(3) can be converted into an augmented objective function

$$J(X) = J(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n, \lambda_1, \lambda_2, ..., \lambda_n);$$

in detail,

$$J(X) = w_1 J_{fair} + w_2 J_{pos} + \sum_{p\in V_{act}} \lambda_p(\theta(\mathbf{v}_p) - 2\pi). \tag{6}$$

This objective function $J(X)$ can be minimized by using the Newton's method [29] below, where we employ a damping factor $\tau = 0.25$ to increase the stability of computing.

Algorithm 1 Newton's Method

1: **while** $\|\delta\| > 10^{-5}$ **do**
2: Solve $\nabla^2 J(X)\delta = -\nabla J(X)$;
3: $X \leftarrow X + \tau\delta$;
4: **end while**

Using the sequential linearly constrained programming to minimize $J(X)$ by neglecting the terms coming from the second derivatives of the constraints in the Hessian matrix $\nabla^2 J(X)$, the equation $\nabla^2 J(X)\delta = -\nabla J(X)$ solved at each step is simplified into

$$\begin{pmatrix} H & & & \Lambda_x^T \\ & H & & \Lambda_y^T \\ & & H & \Lambda_z^T \\ \Lambda_x & \Lambda_y & \Lambda_z & 0 \end{pmatrix} \begin{pmatrix} \delta_{p_x} \\ \delta_{p_y} \\ \delta_{p_z} \\ \lambda \end{pmatrix} = \begin{pmatrix} B_{p_x} \\ B_{p_y} \\ B_{p_z} \\ B_\lambda \end{pmatrix} \tag{7}$$

where λ is the vector of multipliers. Note that since λ is solved in Eq.(7) instead of δ_λ, only the positions of inner vertices are updated by δ_p in the routine of Newton's method. Here B_{p_x}, B_{p_y} and B_{p_z} are the x-, y- and z-components of B_p respectively, and so as Λ_x, Λ_y and Λ_z. From Eq.(6), we can easily obtain that

$$B_p = \{-\frac{\partial}{\partial \mathbf{v}_p}(w_1 J_{fair} + w_2 J_{pos})\} = \{-w_1(\mathbf{v}_p - \frac{1}{|N(\mathbf{v}_p)|}\sum_{k\in N(\mathbf{v}_p)} \mathbf{v}_k) - w_2(\mathbf{v}_p - \mathbf{v}_p^0)\}, \tag{8}$$

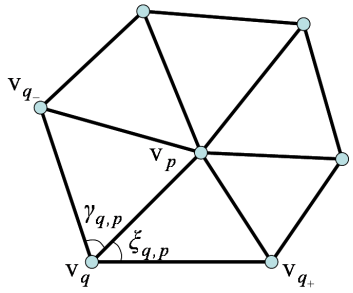

Figure 5: The gradient of summed angle at a vertex respect to the position of the vertex (or of its adjacent vertex) can be computed locally.

$$B_\lambda = -\{\frac{\partial}{\partial \lambda_p} \sum_{p \in V_{act}} \lambda_p(\theta(\mathbf{v}_p) - 2\pi)\} = \{2\pi - \theta(\mathbf{v}_p)\}, \tag{9}$$

$$H = \{h_{i,j}\}, \quad h_{i,j} = \begin{cases} w_1 + w_2 & (i = j) \\ -\frac{w_1}{|N(\mathbf{v}_i)|} & (\mathbf{v}_j \in N(\mathbf{v}_i)) \\ 0 & (otherwise) \end{cases}, \tag{10}$$

$$\Lambda = \{\frac{\partial^2}{\partial \lambda_i \partial \mathbf{v}_j} \sum_{p \in V_{act}} \lambda_p(\theta(\mathbf{v}_p) - 2\pi)\} = \{\frac{\partial \theta(\mathbf{v}_i)}{\partial \mathbf{v}_j}\}. \tag{11}$$

Proposition 1 The gradient of the summed inner angle $\theta(\mathbf{v}_p)$ at an inner vertex $\mathbf{v}_p \in V \setminus \partial V$ respect to the vertex $\mathbf{v}_p$ itself is

$$\frac{\partial \theta(\mathbf{v}_p)}{\partial \mathbf{v}_p} = \sum_{q \in N(\mathbf{v}_p)} \frac{\cot \gamma_{q,p} + \cot \xi_{q,p}}{\|\mathbf{v}_p \mathbf{v}_q\|} (\mathbf{v}_q - \mathbf{v}_p) \tag{12}$$

where $\xi_{q,p}$ and $\gamma_{q,p}$ are the left and right angles aside the edge $\mathbf{v}_p\mathbf{v}_q$ at $\mathbf{v}_q$ – see Fig.5.

Proposition 2 The gradient of the summed inner angle $\theta(\mathbf{v}_q)$ at an inner vertex $\mathbf{v}_q \in V \setminus \partial V$ respect to its adjacent vertex $\mathbf{v}_p$ is

$$\frac{\partial \theta(\mathbf{v}_q)}{\partial \mathbf{v}_p} = \frac{\cot \xi_{q,p} + \cot \gamma_{q,p}}{\|\mathbf{v}_p \mathbf{v}_q\|^2} (\mathbf{v}_p - \mathbf{v}_q) - \frac{(\mathbf{v}_{q_+} - \mathbf{v}_q)}{\|\mathbf{v}_p \mathbf{v}_q\| \|\mathbf{v}_{q_+} \mathbf{v}_q\| \sin \xi_{q,p}} - \frac{(\mathbf{v}_{q_-} - \mathbf{v}_q)}{\|\mathbf{v}_p \mathbf{v}_q\| \|\mathbf{v}_{q_-} \mathbf{v}_q\| \sin \gamma_{q,p}} \tag{13}$$

where $\mathbf{v}_{q_+}$ and $\mathbf{v}_{q_-}$ are the next and last vertices to $\mathbf{v}_q$ in $N(\mathbf{v}_p)$ anti-clockwise – see the illustration in Fig.5.

The proof of Proposition 1 has been given in [13]. The proof of the Proposition 2 is derived in Appendix A, which can also yield the proof of Proposition 1 (see Appendix B).

By Eq.(7-11), we can model the FL mesh patch M_{FL} from a given polygonal mesh M using the sequential linearly constrained programming. Figure 6 gives the progressive results about how the FL mesh in Fig.2 is obtained from a given mesh surface patch in Fig.6(a). Defining the maximal vertex flattenability $\varpi_{\max} = \max\{\varpi(\mathbf{v}_p),\ \forall \mathbf{v}_p \in V_{act}\}$, after 8 steps of iteration, the mesh patch is very similar to the final FL mesh and its $\varpi_{\max}$ has also been close to $\varpi_{\max}$ on the final FL mesh. In conclusion, the computation of FL meshes converges very fast. However, the computation may fail on some meshes with topological obstructions. To overcome the topological obstructions, if high value of $\varpi(\mathbf{v}_p)$ keeps showing on the vertex $\mathbf{v}_p$ during the computation of FL meshes, we locally refine the

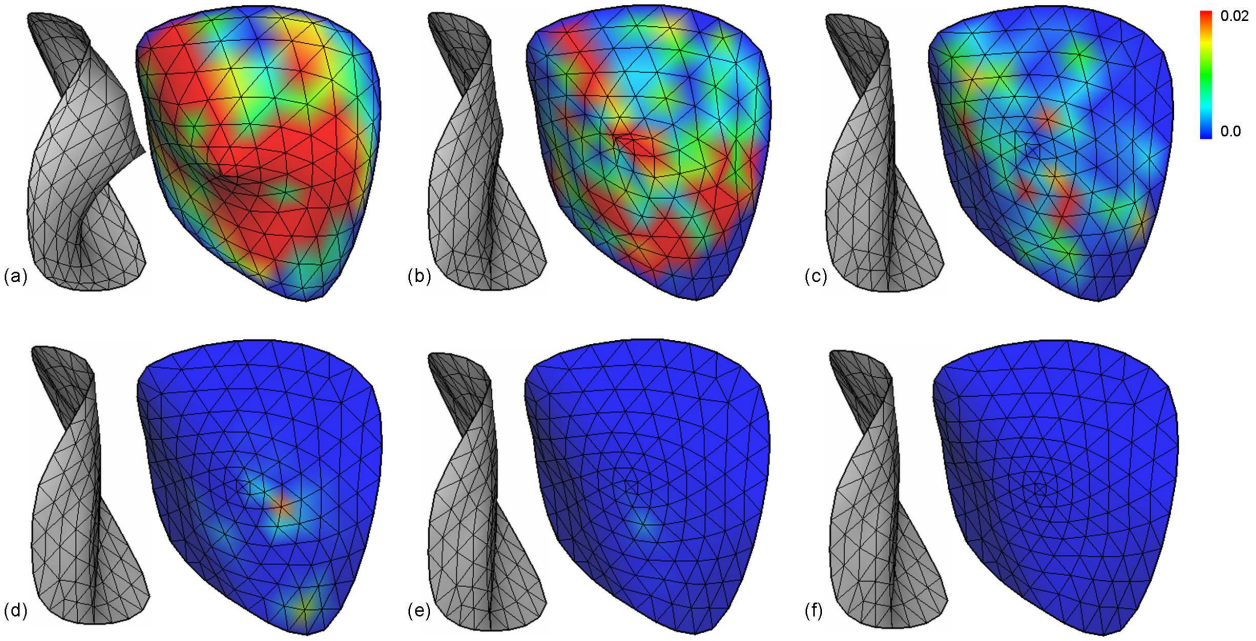

Figure 6: The progressive results of computing FL mesh: (a) the given mesh ($\varpi_{\max} = 1.92$), (b) after 2 steps ($\varpi_{\max} = 1.34 \times 10^{-1}$), (c) after 4 steps ($\varpi_{\max} = 5.46 \times 10^{-2}$), (d) after 6 steps ($\varpi_{\max} = 1.77 \times 10^{-2}$), (e) after 8 steps ($\varpi_{\max} = 2.86 \times 10^{-3}$), (f) the final FL mesh after 20 steps ($\varpi_{\max} = 3.20 \times 10^{-5}$).

triangles around $\mathbf{v}_p$ by the strategy similar to the $\sqrt{3}-$subdivision [20] to add more degree-of-freedom on the mesh under processing.

3. Variational Subdivision of FL meshes

FL mesh introduced in above section can be integrated with subdivision schemes to generate a high quality FL mesh patch from a coarse control mesh. Unlike [26], we do not conduct the subdivision schemes with fixed masks (e.g., the Loop, the Doo-Sabin, or the Catmull-Clark scheme) since the vertices will be adjusted later for making the surface flattenable — the fixed linear combination rules are in fact not followed. Here, a modified variational subdivision algorithm akin to [21] is adopted. Our algorithm consists of three steps: **step 1)** a topological splitting operator is conducted to introduce new vertices to increase the number of degree of freedom (i.e., M^{i+1} is obtained from M^i); **step 2)** discrete fairing operators are applied to move the newly created vertices to increase the overall smoothness and interpolate the user specified boundary curves; **step 3)** we employ the FL mesh processing method on M^{i+1} to compute a FL mesh surface M_{FL}^{i+1}. Iterating these three steps, a hierarchical sequence of FL meshes are generated (e.g., the sequence in Fig.7).

In the first step, instead of uniformly applying the 1-to-4 triangle subdivision, we split triangles based on the length of their edges. More specifically, we first compute the average edge length $\overline{L}$ on M^i — only the edges whose length is greater than $0.5\overline{L}$ are split. Therefore, each triangle on M^i is adaptively converted into one (with no edge split), two (with one edge split), three (with two edges split) or four triangles (with all edges split) on M^{i+1}.

The second step of our modified variational subdivision scheme moves the newly created vertices in the first step to 1) increase the overall smoothness or 2) interpolate the user specified boundary curve. As shown in Fig.7, our scheme allows users to specify some interpolation curves on the boundary

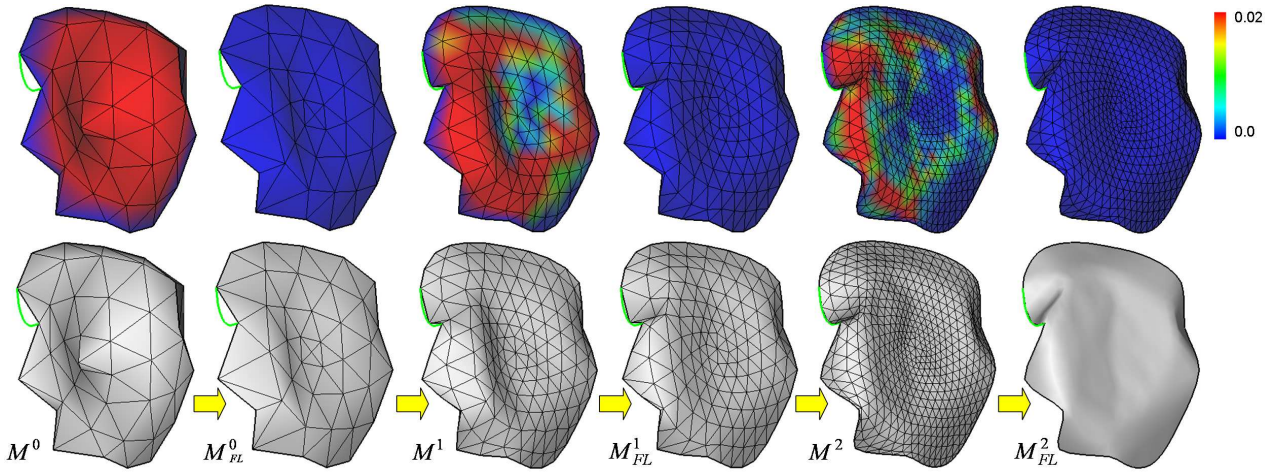

Figure 7: The variational subdivision with FL meshes.

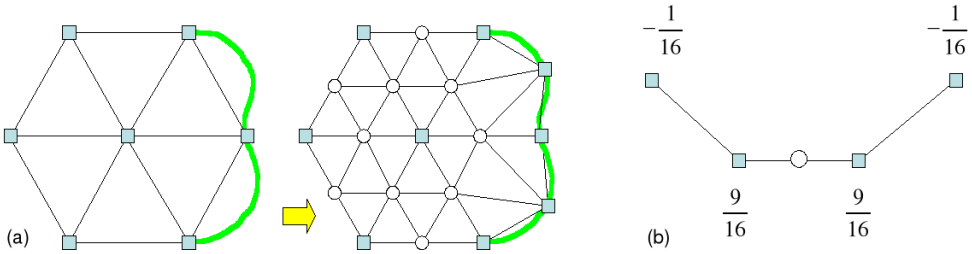

Figure 8: Determine the position of splitting vertices on boundary edges: (a) the boundary edge is with an interpolation curve so that the splitting vertex is positioned in the middle of the curve, and (b) without interpolation curve the splitting vertex is positioned by the modified Butterfly mask [49].

edges (the green curves on the control mesh M^0 in Fig.7). A refined mesh should interpolate these curves during the subdivision. Therefore, when a boundary edge with interpolation curve is split, the newly created vertex should be moved to the middle of the curve; meanwhile, the curve is split into two and attached to the newly created two edges as shown in Fig.8(a). Some of the boundary edges are with no interpolation curve specified; on these edges, the splitting vertex is expected to move to a place make the boundary curve smooth. We conduct the mask for boundary vertices in the modified Butterfly subdivision scheme [49] to determine their new positions. Figure 8(b) shows the linear combination mask: the rounded white node is the vertex position to be determined, and the rectangular gray nodes are the boundary vertices generated in last subdivision (or on M^0) whose positions are fixed. For the newly created inner vertices, their positions are determined by iteratively applying the 1st order umbrella operator and the 2nd order umbrella operator in succession. In detail, we first apply the 1st order umbrella operator to all vertices for 10 runs and then use the 2nd order umbrella operator for 100 runs. To be self-contained, we list the two operators from [21] below. Giving

$$\mathbf{u}(\mathbf{v}_i) = \frac{1}{|N(\mathbf{v}_i)|} \sum_{j \in N(\mathbf{v}_i)} \mathbf{v}_j - \mathbf{v}_i, \tag{14}$$

$$\mathbf{u}^2(\mathbf{v}_i) = \frac{1}{|N(\mathbf{v}_i)|} \sum_{j \in N(\mathbf{v}_i)} \mathbf{u}(\mathbf{v}_j) - \mathbf{u}(\mathbf{v}_i), \tag{15}$$

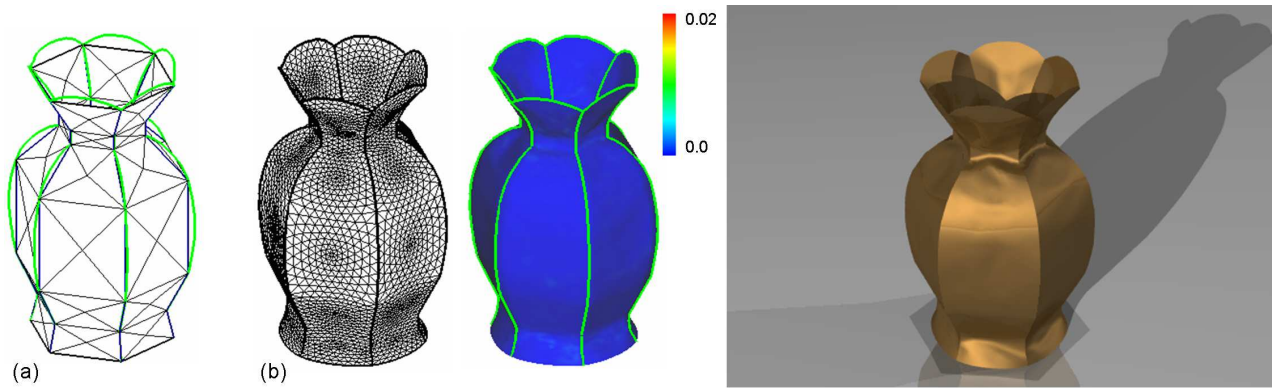

Figure 9: A metal-sheet vase modelled by the subdivision FL meshes: (a) the control mesh with interpolation curves specified and (b) the refined mesh surfaces.

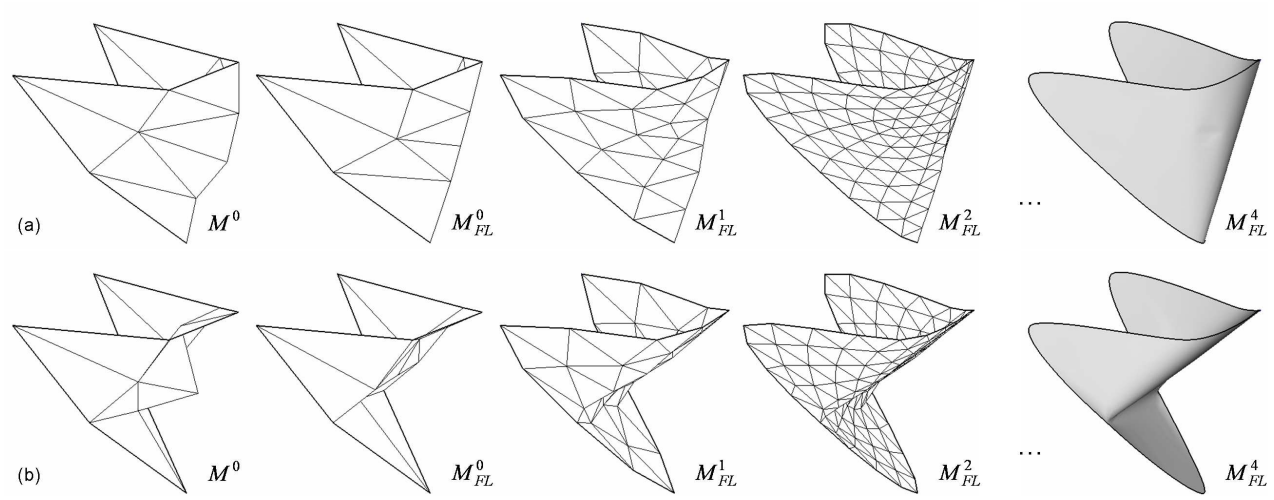

Figure 10: With the same boundary and the same connectivity on a given control mesh, different FL meshes are generated if different initial shapes are given.

thus the 1st order umbrella operator is defined as

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \mathbf{u}(\mathbf{v}_i), \tag{16}$$

and the 2nd order umbrella operator is

$$\mathbf{v}_i \leftarrow \mathbf{v}_i - \frac{1}{\nu}\mathbf{u}^2(\mathbf{v}_i) \tag{17}$$

with $\nu = 1 + |N(\mathbf{v}_i)|^{-1} \sum_{j \in N(\mathbf{v}_i)} |N(\mathbf{v}_j)|^{-1}$.

This subdivision algorithm introduces a very useful method for modelling flattenable mesh surfaces. We find that the surface generated by this method is not the same as the results from other developable meshes modelling approaches, which are derived from the theorems of continuous differential geometry. Our method can model both the smooth surfaces and the surface with the effect like a piece of crumpled leather (see Fig.7). The example shown in Fig.9 demonstrates the functionality of our approach for modelling a relative complex object — a metal-sheet vase by using the subdivision FL meshes. Figure 9(a) gives the control mesh with interpolation curves specified, where the refined

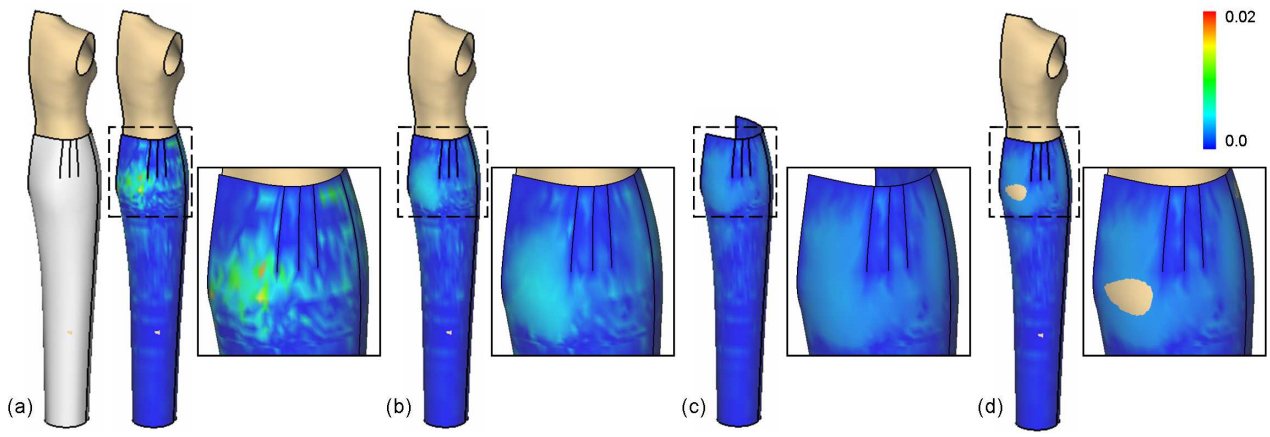

Figure 11: Local flattenable perturbation improves the flattenability of given mesh surfaces — an example from apparel industry: (a) the given pants and the reference human body, (b) the pants after local flattenable perturbation, (c) the local flattenable perturbation without the reference human body, and (d) the interference occurs.

FL meshes interpolate the curves as Fig.9(b). For the same control mesh, by using the variational subdivision scheme without the FL mesh processing step, the resultant mesh surfaces are as shown previously in Fig.1(a) which is non-flattenable. Two other examples of subdividing FL meshes are shown in Fig.10, which have the effect of paper craft.

4. Local flattenable perturbation

The mesh surface generated in some application may be almost flattenable (i.e., with only a few non-flattenable vertices). Figure 11(a) shows an example of such mesh surfaces generated from a 3D CAD system for garment design. For these mesh surfaces, we develop a new local perturbation approach in this section to increase their flattenability with slight position adjustment. The approach is local optimization based, so it can be finished in a relative short time comparing to the computation of FL meshes.

In the local perturbation, only the vertices falling in the disk region of a center vertex $\mathbf{v}_c$ are moved at each step. Suppose $N_r(\mathbf{v}_c)$ denotes the r-rings neighbors of $\mathbf{v}_c$ and $\partial N_r(\mathbf{v}_c)$ represents the vertices on the rth-ring of $\mathbf{v}_c$, we have

$$N_{r+1}(\mathbf{v}_c) = N_r(\mathbf{v}_c) + \partial N_{r+1}(\mathbf{v}_c).$$

The set of vertices that are moved in the local perturbation around $\mathbf{v}_c$ is defined as

$$V_{act} = N_r(\mathbf{v}_c) \bigcap (V \backslash \partial V) \tag{18}$$

In our tests, we usually choose $r = 3$. Choosing a larger support size will of course make the computation more robust to overcome a stick point – local optimum, but will have a longer computing time. When moving vertices in V_{act}, not only vertices in V_{act} but also the vertices adjacent to ∂V_{act} have their flattenability changed. The set of vertices whose flattenability will be effected by the movement of vertices in V_{act} is

$$V_{dev} = N_{r+1}(\mathbf{v}_c) \bigcap (V \backslash \partial V). \tag{19}$$

When moving vertices in V_{act}, we wish that:

- The surface shape after perturbation will still approximate the original surface around $\mathbf{v}_c$, which can be formulated as the same functional J_{pos} in Eq.(5);
- All vertices in V_{dev} are as flattenable as possible (i.e., $\forall \mathbf{v}_p \in V_{dev}$ letting $\varpi(\mathbf{v}_p) \simeq 0$);
- The interference between the given surface and some reference object H is prevented.

The last requirement is important for many industry applications (e.g., the design of garment and shoes) as the products presented by flattenable mesh patches are usually worn by a reference model (i.e., part of a human body). All these factors are formulated into a constrained optimization problem

$$\begin{array}{ll} \arg\min_{p\in V_{act}} & J_{pos} \\ \textit{subject to} & \theta(\mathbf{v}_q) \equiv 2\pi \qquad (q \in V_{dev}) \\ & (\mathbf{v}_r - \mathbf{h}_r)\cdot \mathbf{n}_{h_r} \geq \epsilon \quad (r \in V_{collid}) \end{array} \tag{20}$$

where $\mathbf{h}_r$ is the current closest point of $\mathbf{v}_r$ on the reference body H, $\mathbf{n}_{h_r}$ is the unit normal vector at $\mathbf{h}_r \in H$ pointing outwards, and $V_{collid} \subset V_{act}$ is the set of vertices whose distance to the reference body is less than the user specified collision tolerance ϵ.

Again, the optimization problem is solved by using the sequential linearly constrained programming, where the Lagrangian function below is adopted.

$$J_{local}(X) = \sum_{p\in V_{dev}} \frac{1}{2}\|\mathbf{v}_p - \mathbf{v}_p^0\|^2 + \sum_{q\in V_{dev}} \lambda_q(\theta(\mathbf{v}_q) - 2\pi) + \sum_{r\in V_{collid}} \lambda_r((\mathbf{v}_r - \mathbf{h}_r)\cdot \mathbf{n}_{h_r} - \epsilon) \tag{21}$$

The linear equation system solved at each iteration is

$$\begin{pmatrix} I & & & \Lambda_x^T \\ & I & & \Lambda_y^T \\ & & I & \Lambda_z^T \\ \Lambda_x & \Lambda_y & \Lambda_z & 0 \end{pmatrix} \begin{pmatrix} \delta_{p_x} \\ \delta_{p_y} \\ \delta_{p_z} \\ \lambda \end{pmatrix} = \begin{pmatrix} B_{p_x} \\ B_{p_y} \\ B_{p_z} \\ B_\lambda \end{pmatrix} \tag{22}$$

where

$$B_p = \{\mathbf{v}_p^0 - \mathbf{v}_p\},\ B_\lambda = \begin{pmatrix} 2\pi - \theta(\mathbf{v}_q) \\ -((\mathbf{v}_r - \mathbf{h}_r)\cdot \mathbf{n}_{h_r} - \epsilon) \end{pmatrix},\ \Lambda = \begin{pmatrix} \alpha_{q,p} \\ \beta_{r,p} \end{pmatrix} = \begin{pmatrix} \partial\theta(\mathbf{v}_q)/\partial\mathbf{v}_p \\ \delta_{r,p}\mathbf{n}_{h_r} \end{pmatrix}$$

with the Kronecker delta $\delta_{i,j} = \begin{cases} 1 & (i=j) \\ 0 & (i\neq j) \end{cases}$.

Being not surprised, we find that the problem defined in Eq.(20) is not as stable as the one in Eq.(3). This is because that fairness term defined in Eq.(3) take the role of regularization to the mesh system which prevents the singularity. Trying to avoid singular results, we employ *Singular Value Decomposition* (SVD) [31] to solve the linear equation system in Eq.(20). Firstly, λ in

$$(\Lambda_x\Lambda_x^T + \Lambda_y\Lambda_y^T + \Lambda_z\Lambda_z^T)\lambda = \Lambda_x B_{p_x} + \Lambda_y B_{p_y} + \Lambda_z B_{p_z} - B_\lambda \tag{23}$$

is solved by SVD. Then, δ_{p_x}, δ_{p_y} and δ_{p_z} are consequently computed by

$$\delta_{p_x} = B_{p_x} - \Lambda_x^T\lambda, \qquad \delta_{p_y} = B_{p_y} - \Lambda_y^T\lambda, \qquad \delta_{p_z} = B_{p_z} - \Lambda_z^T\lambda. \tag{24}$$

Based on these formulas (Eqs.(22-24)), we introduce the **Algorithm** *LocalFlattenablePerturbation* to improve the flattenability of a given mesh patch by local perturbation. The basic idea is that by storing all vertices into a maximum heap keyed with the flattenability on vertices, we iteratively

Algorithm 2 LocalFlattenablePerturbation

1: For a given mesh patch M, compute the flattenability on all vertices $\mathbf{v}_p \in V \backslash \partial V$ and record the maximal flattenability value $\varpi_{\max}$;
2: Save the current positions of all vertices as the current optimal record;
3: Search the closest tracking point of all vertices on the reference body H;
4: Insert all vertices into a maximum heap Υ;
5: **repeat**
6: **while** Υ is not empty **AND** the flattenability of its top node is greater than $0.5\varpi_{\max}$ **do**
7: Remove the top node $\mathbf{v}_t$ from Υ;
8: Determine V_{act}, V_{dev} and V_{collid} by a user specified ring number r;
9: **repeat**
10: Compute $(\delta_{p_x}, \delta_{p_y}, \delta_{p_z})$ for all $\mathbf{v}_p \in V_{act}$ using Eqs.(22-24);
11: $\mathbf{v}_p \leftarrow \mathbf{v}_p + \tau(\delta_{p_x}, \delta_{p_y}, \delta_{p_z})$;
12: **until** the process has been iterated for more than 5 times;
13: Update the closest tracking point of all vertices in V_{act};
14: **end while**
15: **if** the current maximal flattenability is less than $\varpi_{\max}$ **then**
16: Update $\varpi_{\max}$ and save the current positions of all vertices as an optimal result;
17: **end if**
18: Add all the removed nodes back into Υ, and update the position of all moved vertices in Υ by their new flattenability;
19: **until** the terminal condition is satisfied.

perturb the regions around k-most non-flattenable vertices while preventing the interference with the reference body H. Note that the value of k is controlled by the number of vertices whose flattenability is greater than $0.5\varpi_{\max}$. The pseudo-code has been listed.

A soft update strategy for Newton's method is employed here — we usually choose $\tau = 0.1$ to stabilize the computation. In **Algorithm** *LocalFlattenablePerturbation*, there are two time-consuming space searching steps — step 3 and 13. For step 3, we speed up the searching by uniformly decomposing the bounding box of H into several sub-regions and the closest point of a vertex $\mathbf{v}_p$ is searched in the sub-region holding $\mathbf{v}_p$ and its 26 neighboring sub-regions. In step 13, a local search approach from [48] is adopted: to update the closest tracking point of a vertex $\mathbf{v}_t \in V_{act}$, the polygonal faces $f \in H$ holding its newest tracking point are searched among the polygons adjacent to the face holding its current tracking point and the polygons that are tracked by the vertices $\mathbf{v}_j$ $(j \in N(\mathbf{v}_t))$.

There is no theoretical guarantee about the convergency of the computation conducted in **Algorithm** *LocalFlattenablePerturbation*. However, steps 15-17 ensures that this algorithm always output a result not worse than the input by keeping the optimal result during the computation. In our implementation, the terminal condition of the outer iteration loop (step 19) consists of three part — we stop the iteration if 1) $\varpi_{\max} < \varepsilon$ ($\varepsilon = 0.001$ is chosen by users in our application), or 2) the iteration has been repeated for 10 times without updating $\varpi_{\max}$, or 3) the iteration has been repeated for more than 50 times.

Figure 11 shows an example application of this local flattenable perturbation in apparel industry. The computation stops at $\varpi_{\max} = 4.75 \times 10^{-3}$ by the second type of terminal condition after running the outer iteration for 21 steps (see Fig.11(b)). Our experimental tests found that the collision constraint somewhat prevents the further optimization on given surfaces. After removing the reference object, the optimization runs for 36 steps and stops at $\varpi_{\max} = 3.14 \times 10^{-3}$ (see Fig.11(c)) — of course, it yields the interference between the pants and the reference object H as

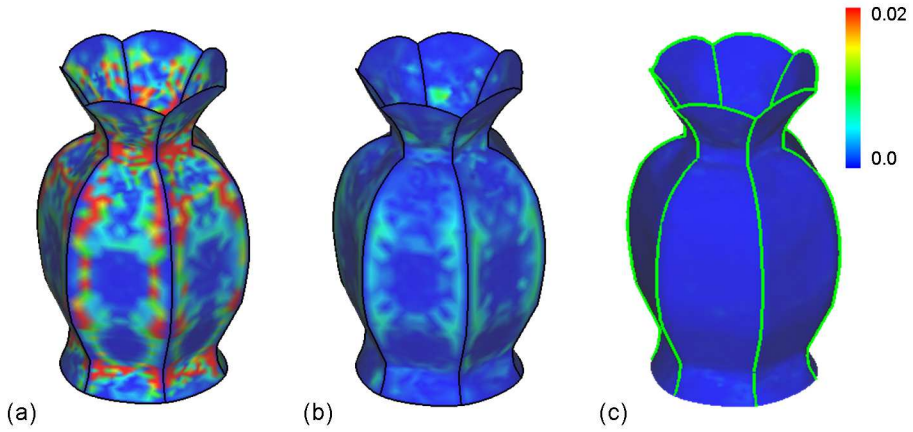

Figure 12: The comparison of results from [44] and the FL mesh computation: (a) the input mesh, (b) the output of [44] ($\varpi_{\max} = 1.18 \times 10^{-2}$), and (c) the computed FL mesh ($\varpi_{\max} = 1.05 \times 10^{-3}$).

shown in Fig.11(d).

5. Discussions

Similar to other numerical optimization based approaches, the computation of FL meshes relies on the initial input. Although the Laplacian term has improved the robustness of the computation, there is still no theoretical guarantee on the generation of FL meshes on arbitrary input mesh surface. In other words, there are input meshes where the FL-mesh computation and the local flattenable perturbation fail because of topological obstructions. However, during our tests, we seldom find the input that cannot generate a FL mesh surface after the subdivision or the local $\sqrt{3}$-refinement. This is because that the subdivision (or the refinement) increases the degree-of-freedom for optimization so that overcomes the topological obstructions. Different initial control meshes, even if they have the same boundary edges and the same connectivity, may lead to different resultant FL meshes. Models shown in Fig.10 exemplify this. The result shape cannot be predicted from the shape of input control mesh; however, the final shape can be predicted by the FL mesh surface computed from the input mesh (i.e., M^0_{FL}). Therefore, this tool is easy to use — once you create a control mesh and apply the FL mesh processing on it, you will see whether your mesh can be processed into a FL mesh close to your input. If your control mesh is very close to a flattenable one, the processed FL mesh will be similar to your input.

The computation of FL meshes and the local flattenable perturbation depend not only on the number of vertices in V_{act}, but also on the geometry of input mesh — whether it is close to a flattenable mesh. By preparing the examples on a PC with 3.0 GHz Pentium CPU, the computation times have been listed in Table 1. Note that for the mesh with 7.8k faces, the FL mesh computation takes less than 1 second for each iteration step, which is much faster than the time reported in [12] running on a PC with the same configuration but less triangles (they take 29 seconds for each step on a mesh with only 7k faces). The approach in [44] needs about 309 seconds to process the mesh shown in Fig.12(a) (which is M^3 in Fig.9) resulting in patches not as good as FL meshes. Figure 12 shows the result comparison of [44] and M^3_{FL}. Solving linear equation system Eq.(7) is the most time-consuming step in the computation of FL meshes. In our first implementation, the sparse linear systems are solved by PBCG in [31] with the 1st-type terminal condition (more specifically, the conjugate gradient iteration stops when $|\mathbf{A} \cdot \mathbf{x} - \mathbf{b}|/|\mathbf{b}| < 10^{-3}$), where the times used are listed in the bracket of Table 1. To speed up, we move to the LU-decomposition solver [22] so that the

Table 1: Computational Statistics

Example	Method	Vertices	Faces	Time* (sec)	Iteration Steps
Fig.6	FL meshes	142	250	0.81 (1.37)	56
Fig.7 - M^0	FL meshes	41	64	0.02 (0.05)	68
Fig.7 - M^1	FL meshes	141	248	1.03 (1.67)	77
Fig.7 - M^2	FL meshes	527	988	6.55 (57.5)	58
Fig.9 - M^1	FL meshes	270	516	0.67 (0.28)	69
Fig.9 - M^2	FL meshes	1,028	2,008	3.09 (8.52)	39
Fig.9 - M^3	FL meshes	3,952	7,808	21.5 (252.6)	47
Fig.10 - with H^*	Local Perturbation	3,539	3,458	21.5	21
Fig.10 - without H	Local Perturbation	3,539	3,458	48.8	36

* 1) The times listed in the brackets are by the PBCG solver [31], and the times outside the brackets are by the LU-decomposition solver [22]; 2) The reference human body is a triangular mesh with 5,659 vertices and 5,544 triangles.

computational time has been greatly reduced (especially for the surface with large mesh size – see Table 1).

Our approach for computing flattenable mesh surfaces has the following limitations:

- The current method only preserves G^0 continuity on the boundary of a FL mesh surface, which is acceptable for the sheet-manufacturing of soft materials (e.g., the apparel industry). For those industries considering more stiff materials (e.g., the ship industry), G^1 continuity may be expected. We will address this issue in our near future work.

- The FL mesh is not a connectivity invariant representation (i.e., for an input with the same geometry shape but with different mesh connectivity, the resultant FL could be different). This limitation can be somewhat overcome if we apply the remeshing procedure like [5] to remesh the input surface into a semi-regular mesh with almost uniform triangle size. We have tested this with our implementation of [5], the result is satisfactory.

- The collision response method, which is currently implemented in the local flattenable perturbation, is relative primitive. Ideally, the collision handling algorithm should provide a response that changes the direction of colliding vertices in order to simulate the slippage of vertices on the reference bodies. A geometric distribution correction method [42] will be considered in our future work.

6. Conclusion

This paper presents two approaches for the modelling of flattenable mesh surfaces which can be flattened into a two-dimensional pattern without stretching. Firstly, a new flattenable mesh surface — Flattenable Laplacian (FL) mesh is introduced and its relevant modelling tool is developed by the constrained numerical optimization. In the following, the technique of FL meshes is integrated with the variational subdivision scheme to model more complex objects. Furthermore, a local flattenable perturbation approach has been developed to increase the flattenability of a given mesh surface if it is almost flattenable. The constraint for preventing interference between the surface under process and their nearby reference objects has been incorporated. The computation of local permutation is

also solved by the constrained optimization. The experimental results show that our method can successfully model flattenable meshes in a reasonable time on a PC with standard configuration. In summary, the approach developed in this paper provides a very useful geometric modelling tool for designing products built from sheet materials in various applications.

Acknowledgments

The author would like to thank the support of the CUHK project CUHK/2050341 and Shengjun Liu who helped to render some figures.

References

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

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

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

[4] D.E. Breen, D.H. House and M.J. Wozny, "Predicting the drape of woven cloth using interacting particles", *Proceedings of SIGGRAPH 94*, pp.365-372, 1994.

[5] M. Botsch, and L. Kobbelt, "A remeshing approach to multiresolution modeling", *Proceedings of Eurographics Symposium on Geometry Processing*, pp.185-192, 2004.

[6] R. Bridson, and S. Marino, and R. Fedkiw, "Simulation of clothing with folds and wrinkles", *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2003*, pp.28-36, 2003.

[7] D. Baraff, and A. Witkin, "Large steps in cloth simulation", *Proceedings of SIGGRAPH 98*, pp.43-54, 1998.

[8] D. Chen, D. Cohen-Or, O. Sorkine, and S. Toledo, "Algebraic analysis of high-pass quantization", *ACM Transactions on Graphics*, vol.24, no.4, pp.1259-1282, October 2005.

[9] K.J. Choi, and H.S. Ko, "Stable but responsive cloth", *ACM Transactions on Graphics*, vol.21, no.3, pp.604-611, 2002.

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

[11] C.H. Chu and C.H. Séquin, "Developable Bézier patches: properties and design", *Computer-Aided Design*, vol.34, no.7, pp.511-527, 2002.

[12] 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* (Eurographics'06 Proceedings), vol.25, no.3, pp.625-634, 2006.

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

[14] M. Desbrun, M. Meyer, P. Schröder, and A.H. Barr, "Implicit fairing of irregular meshes using diffusion and curvature flow", *Proceedings of SIGGRAPH 99*, pp.317-324, 1999.

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

[16] M.S. Floater and K.Hormann, "Surface parameterization: a tutorial and survey", *Advances in Multiresolution for Geometric Modelling*, N.A. Dodgson, M.S. Floater, and M.A. Sabin (eds.), Springer-Verlag, Heidelberg, pp.157-186, 2005.

[17] D. Julius, V. Kraevoy, and A. Sheffer, "D-Charts: quasi-developable mesh segmentation", *Computer Graphics Forum*, *Proceedings of Eurographics 2005*, vol.24, no.3, pp.581-590, 2005.

[18] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry", *Proceedings of ACM SIGGRAPH 2000*, pp.279-286, 2000.

[19] Z. Karni, C. Gotsman, and S.J. Gortler, "Free-boundary linear parameterization of 3D Meshes in the presence of constraints", *Proceedings of Shape Modeling International*, pp.266-275, June, 2005.

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

[21] L. Kobbelt and P. Schröder, "A multiresolution framework for variational subdivision", *ACM Transactions on Graphics*, vol.17, no.4, pp.209-237, 1998.

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

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

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

[25] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, "Least squares conformal maps for automatic texture atlas generation", *ACM Transactions on Graphics*, *SIGGRAPH 2002*, vol.21, pp.362-371, 2002.

[26] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, "Geometric modeling with conical meshes and developable surfaces", ACM Transactions on Graphics, vol.25, no.3, pp.681-689, 2006.

[27] M. Meyer, M. Desbrun, P. Schröder, and A.H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds", *Visualization and Mathematics III*, pp.35-58, 2003, Springer.

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

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

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

[31] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge: Cambridge University Press, pp.71-89, 1995.

[32] M. Peternell and T. Steiner, "Reconstruction of Piecewise Planar Objects from Point Clouds", *Computer-Aided Design*, vol. 36, pp. 333-342, 2004.

[33] H. Pottmann and J. Wallner, "Approximation Algorithms for Developable Surfaces", *Computer Aided Geometric Design*, vol. 16, pp.539-556, 1999.

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

[35] O. Sorkine, D. Cohen-Or, D. Irony, and S. Toledo, "Geometry-aware bases for shape approximation", *IEEE Transactions on Visualization and Computer Graphics*, vol.11, no.2, pp.171-180, March/April 2005.

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

[37] A. Sheffer, B. Lévy, M. Mogilnitsky, and A. Bogomjakov, "ABF++: fast and robust angle based flattening", *ACM Transactions on Graphics*, vol.24, no.2, pp.311-330, 2005.

[38] A. Sheffer, and E. de Sturler, "Parameterization of faceted surfaces for meshing using angle based flattening", *Engineering with Computers*, vol.17, no.3, pp.326-337, 2001.

[39] O. Sorkine, "Differential representations for mesh processing", *Computer Graphics Forum*, vol.25, no.4, pp.789-807, 2006.

[40] G. Taubin, "A signal processing approach to fair surface design", *Proceedings of ACM SIGGRAPH 95*, pp.351-358.

[41] P. Volino, M. Courchesne, and N. Magnenat-Thalmann, "Versatile and efficient techniques for simulating cloth and other deformable objects", *Proceedings of SIGGRAPH 95*, pp.137-144, 1995.

[42] P. Volino, and N. Magnenat-Thalmann, "Accurate collision response on polygon mehses", *Proceedings of Computer Animation 2000*, pp.137-144, 2000.

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

[44] C.C.L. Wang, and K. Tang, "Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches", *The Visual Computer*, vol.20, no.8-9, pp.521-539, 2004.

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

[46] C.C.L. Wang, Y. Wang, and M.M.F. Yuen, "On increasing the developability of a trimmed NURBS surface", *Engineering with Computers*, vol.20, no.1, pp.54-64, 2004.

[47] C.C.L. Wang, Y. Wang, and M.M.F. Yuen, "Design automation for customized apparel products", *Computer-Aided Design*, vol.37, no.7, pp.675-691, 2005.

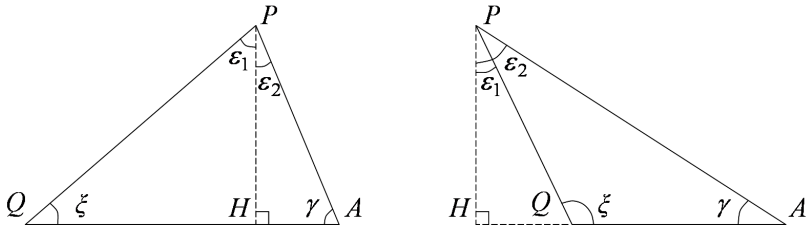

Figure 13: For the proof of Proposition 1 and 2: left) acute angle, and right) obtuse angle.

[48] Y. Wang, C.C.L. Wang, and M.M.F. Yuen, "Duplicate-skins for compatible mesh modelling", *Proceedings of the 2006 ACM symposium on Solid and Physical Modeling*, pp.207-217, Cardiff University, Wales, UK, June 6-8, 2006.

[49] D. Zorin, P. Schröder, and W. Sweldens, "Interpolating subdivision for meshes with arbitrary topology", *Proceedings of SIGGRAPH 96*, pp.189-192, 1996.

Appendix A

Proposition 2 can be proved as below. In the summed angle $\theta(\mathbf{v}_q)$ at $\mathbf{v}_q$, only the components $\xi_{q,p}$ and $\gamma_{q,p}$ are effected by the position of $\mathbf{v}_p$. Let us consider $\xi_{q,p}$ first (see Fig.13). When ξ is an acute angle, $\xi = \arccos \frac{\|QH\|}{\|PQ\|}$ (with $\|QH\| = (P-Q) \cdot \frac{QA}{\|QA\|}$)) yields

$$\begin{aligned}
\frac{d\xi}{dP} &= -\frac{1}{\sqrt{1-\frac{\|QH\|^2}{\|PQ\|^2}}} \frac{d}{dP}(\frac{\|QH\|}{\|PQ\|}) \\
&= -\frac{\|PQ\|}{\|PH\|\|PQ\|^2}(\|PQ\| \frac{d\|QH\|}{dP} - \|QH\| \frac{d\|PQ\|}{dP}) \\
&= -\frac{\|PQ\|}{\|PH\|\|PQ\|^2}(\frac{\|PQ\|}{\|QA\|}(A-Q) - \frac{\|QH\|}{\|PQ\|}(P-Q)) \\
&= \frac{\cot\xi}{\|PQ\|^2}(P-Q) - \frac{1}{\|QA\|\|PQ\|\sin\xi}(A-Q)
\end{aligned}$$

When ξ is an obtuse angle, $\xi = \arccos(-\frac{\|QH\|}{\|PQ\|})$ (with $\|QH\| = (Q-P) \cdot \frac{QA}{\|QA\|}$) leads to

$$\begin{aligned}
\frac{d\xi}{dP} &= -\frac{1}{\sqrt{1-\frac{\|QH\|^2}{\|PQ\|^2}}} \frac{d}{dP}(-\frac{\|QH\|}{\|PQ\|}) \\
&= \frac{\|PQ\|}{\|PH\|\|PQ\|^2}(\|PQ\| \frac{d\|QH\|}{dP} - \|QH\| \frac{d\|PQ\|}{dP}) \\
&= \frac{\|PQ\|}{\|PH\|\|PQ\|^2}(\frac{\|PQ\|}{\|QA\|}(Q-A) - \frac{\|QH\|}{\|PQ\|}(P-Q)) \\
&= \frac{\cot\xi}{\|PQ\|^2}(P-Q) - \frac{1}{\|QA\|\|PQ\|\sin\xi}(A-Q)
\end{aligned}$$

The part corresponding to $\gamma_{q,p}$ can be derived in a similar way.

Q.E.D.

Appendix B

Based on the formulas in Appendix A, we can also proof Proposition 1 as follows.

$$\frac{d\varepsilon_1}{dP} = \frac{d}{dP}(\frac{\pi}{2} - \xi) = -\frac{d\xi}{dP} = \frac{1}{\|QA\|\|PQ\|\sin\xi}(A-Q) + \frac{\cot\xi}{\|PQ\|^2}(Q-P)$$

and

$$\frac{d\varepsilon_2}{dP} = \frac{d}{dP}(\frac{\pi}{2} - \gamma) = -\frac{d\gamma}{dP} = \frac{1}{\|QA\|\|PA\|\sin\gamma}(Q - A) + \frac{\cot\gamma}{\|PQ\|^2}(A - P)$$

As $\|PQ\|\sin\xi = \|PA\|\sin\gamma = \|PH\|$, the first terms will be eliminated when summing all apex angles around P.

Q.E.D.