Quasi-Developable Mesh Surface Interpolation via Mesh Deformation

Kai Tang, *Member*, *IEEE*, and Ming Chen

Abstract—We present a new algorithm for finding a most "developable" smooth mesh surface to interpolate a given set of arbitrary points or space curves. Inspired by the recent progress in mesh editing that employs the concepts of preserving the Laplacian coordinates and handle-based shape editing, we formulate the interpolation problem as a mesh deformation process that transforms an initial developable mesh surface, such as a planar figure, to a final mesh surface that interpolates the given points and/or curves. During the deformation, the developability of the intermediate mesh is maintained by means of preserving the zero-valued Gaussian curvature on the mesh. To treat the high nonlinearity of the geometric constrains owing to the preservation of Gaussian curvature, we linearize those nonlinear constraints using Taylor expansion and eventually construct a sparse and overdetermined linear system that is subsequently solved by a robust least squares solution. By iteratively performing this procedure, the initial mesh is gradually and smoothly "dragged" to the given points and/or curves. The initial experimental tests have shown some promising aspects of the proposed algorithm as a general quasi-developable surface interpolation tool.

Index Terms—Developable surfaces, mesh deformation, garment design, flat mesh, cloth simulation.

1 Introduction

Developable surfaces, namely, those that are isometric to their counterparts in the plane, play an extremely important role in our daily life. Examples in this are numerous. In garment industry, virtually all the clothing products must be developable as they are made of planar fabrics. In architectural design, most structures eventually undergo a so-called "tiling" process—the surface of the structure, e.g., a roof, needs to be covered by small tiles of some similar shape, which again demands that the surface be as developable as possible, so to avoid overlapping of the tiles. In deep draw die development, although the final metal sheet usually undergoes certain distortion due to plastic deformation, the binder wrap surface is always preferred to be developable (see Fig. 1). Also, not necessarily from the manufacturing consideration, owing to their atheistic appeal, developable surfaces are frequently used in home artifacts, modern art [1], [2], etc.

In a typical shape modeling process when developability is required (so called *developable surface interpolation*), the designer is given some points and/or curves in the 3D space (called *boundary constraints*), and the goal is to define a smooth and developable surface to interpolate them. Notwithstanding the progress over the last 15 years or so, existing solutions suffer from one or more of the following drawbacks:

1. very rigid requirements on the geometry of the boundary constraints,
2. nonsmoothness of the surface—the final surface is comprised by many independent developable patches,
3. high computing cost, in terms of both runtime and memory cost, or
4. no guarantee of the developability on the final surface.

In this paper, we introduce a robust and simple-to-implement algorithm for the *quasi-developable mesh surface interpolation problem*—find a mesh surface that not only meets the given interpolation requirement but also seeks maximum developability on the surface. The algorithm is in principle an iterative process based on energy minimization. However, different from most existing energy-minimization type algorithms that depend on traditional heuristic search methods such as the Newton's method or Genetic Algorithm, our algorithm formulates the minimization as a continuous shape deformation process that starts from some developable mesh surface (e.g., a planar figure) and ends on the final mesh surface that satisfies both the interpolation and developability requirements. Our algorithm is motivated by recent progress in mesh editing that employs the Laplacian operator [3], [4]. In a mesh editing environment that relies on the Laplacian operation, a mesh surface is continuously deformed, while the Laplacian coordinates at all the vertices are being preserved as much as possible (see Fig. 2). In a sense, our algorithm for the quasi-developable surface interpolation works in a similar fashion: an initial developable mesh surface is continuously deformed until certain designated vertices on the surface (to be referred to as *anchor points*) coincide with their target positions. Different from the Laplacian coordinates, though, the local geometric property that is kept preserved in the deformation is the (zero-valued) Gaussian curvature that ensures the developability. While Laplacian coordinates are linear, the (discrete) Gaussian curvature at a vertex is a highly nonlinear function of the vertex and its neighboring

• *The authors are with the Mechanical Engineering Department, Hong Kong University of Science and Technology, Clearwater Bay, Hong Kong. E-mail: {mektang, mecm}@ust.hk.*

Manuscript received 23 May 2008; revised 9 Aug. 2008; accepted 21 Oct. 2008; published online 29 Oct. 2008.
Recommended for acceptance by G. Taubin.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2008-05-0067.
Digital Object Identifier no. 10.1109/TVCG.2008.192.

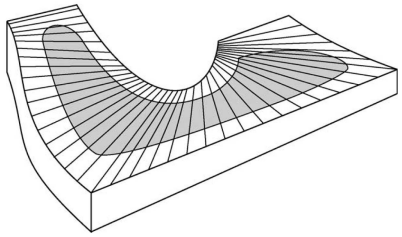

Fig. 1. Binder and binder wrap surfaces and die cavity for a fender die. A developable binder wrap surface reduces potential cracks and buckling on the final stamped sheet. (Frey [18].)

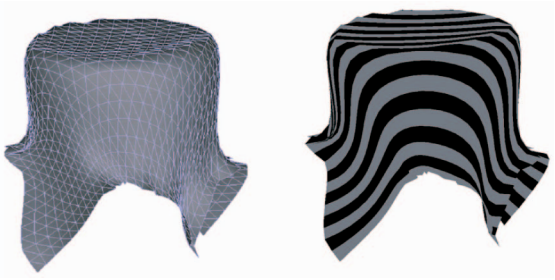

Fig. 2. Laplacian deformation. An initially flat mesh is continuously deformed, while its Laplacian coordinates are kept preserved (as much as possible). Neither length nor surface area is preserved and the final shape is highly nondevelopable.

vertices. To counter this nonlinearity, we use Taylor expansion to linearize the geometric constraint of zero Gaussian curvature. In the same light, the nonlinear boundary constraints are also linearized. Furthermore, to help ensure a smooth deformation, a constraint for length preservation is added to the system and linearized as well. These three types of linearized constraints result in a large and overdetermined linear system, which is subsequently solved by the least squares method. By iteratively undergoing this procedure, the original mesh surface gradually and smoothly deforms to its target while in all the time maintaining the developability. Our initial experiments have demonstrated the high efficacy of the proposed approach in achieving maximum developability on the final interpolation surface.

2 Background and Previous Work

Intuitively, developable surfaces are those that can be unfolded into the plane with no length or area distortion. Developable surfaces are closely related to ruled surfaces, where a ruled surface is defined as the trajectory of a line (called a *ruling*) when it moves in space along a prescribed directrix curve [5], [6]. It is well known that a G^2 surface is developable if and only if it is a ruled surface whose normals are constant along each ruling [6]. Such a surface has a distinct characteristic: its normals on the Gaussian sphere form a continuous curve and is sometimes called a *torsal developable* surface [7]. A G^1 surface is sometimes called a *composite developable surface* if it is a union of some torsal developable surfaces. From differential geometry, a ruled surface is developable if and only if its Gaussian curvature is zero everywhere on the surface.

Strictly speaking, a G^1 surface is either developable or nondevelopable but not in between. Nevertheless, in our case, any smooth surface is associated with a degree of *developability*, which is measured by the integral of the absolute Gaussian curvature over the entire surface—the less the integral is, the more "developable" the surface is. This relaxation is by no means unreasonable. In contrast, in real life, almost all the "soft" products (e.g., garment, paper, thin metal sheet, etc.) undergo certain distortion such as stretch or compression. Therefore, relating to our quasi-developable mesh surface interpolation problem, our objective is to find one "most" developable mesh surface to satisfy the given boundary constraints. This is different from the problem of fitting a point cloud with a single "100 percent" developable torsal surface (cf., [8] and [9]), wherein the objective is to minimize the interpolation error.

Exact surface interpolation. Almost all the works in exact surface interpolation deal with only torsal developable surfaces and are restricted to four-sides patches. Basically, an exact surface—parametric or algebraic—is defined to interpolate the given points or curves. Bezier or B-spline surfaces are the most used ones, and the developability is enforced by nonlinear constraints [10], [11], [12], [13], [14], [15]. To facilitate the modeling task, some novel algebraic tools were proposed. Pottmann and Wallner [6] used a dual space approach to define a plane-based control interface for modeling developable patches. Bo and Wang [16] presented an interactive modeling scheme that can be used to simulate paper bending in computer animation. Notwithstanding its mathematical rigor and elegance, exact surface interpolation is considered ineffective and impractical for general interpolation due to the extremely stringent and nonlinear constraints required.

Mesh surface interpolation. Frey [17], [18] introduced the concept of boundary triangulation—where all the triangles have their vertices on two 3D polylines—and presented a simple modeling method for generating a discrete (nearly) developable surface interpolating a given closed polyline. However, the method is restricted to height-field surfaces, i.e., surfaces must be monotone with respect to the XY. The resulting surface depends on the choice of projection direction. Moreover, only a single polyline can be interpolated. Inspired by the Frey's work, Tang and Wang [19] presented an approach that is capable of finding the most developable boundary triangulation interpolating two arbitrary polylines. Their method formulates the problem as a deterministic search problem and uses Dijkstra algorithm to perform the optimization, which is fast and robust. Nonetheless, their method can only generate a (discrete) torsal surface. Recently, based on boundary triangulation and convex hull principle, Rose et al. [7] gave an algorithm that is able to obtain a discrete (nearly) developable surface interpolating an arbitrary closed polyline. Though the algorithm can be expanded to more than one polyline, the resulting mesh usually is not smooth, that is, where two torsal surfaces meet incurs discontinuity in normals.

Developable surface approximation. There exist a number of algorithms that aim at approximating a given mesh surface by one or more developable surfaces, continuous, or discrete. Wang and Tang [20] suggested deforming a given mesh to minimize its total Gaussian curvature, where a gradient-based search method is

adopted for the minimization. Their algorithm however is not suitable for interpolation, since interpolation constraints were not considered in the deformation process. Mitani and Suzuki [21] introduced a method that approximates an arbitrary mesh surface by triangular strips, and the latter can be unfolded into their planar counterparts. Despite its usefulness in some particular applications, e.g., modeling paper craft toys, this method is not suitable for our interpolation problem, as the resulting mesh surface is only G^0—the places where triangular strips meet do not have continuous normals. Due to the same reason, the scheme introduced by Shatz et al. [22] is not suitable for us either, since their method approximates a mesh surface by conics that again cause discontinuity in normals. The algorithm proposed by Julius et al. [23], which is based on the Lloyd's segmentation scheme, partitions a mesh surface into (nearly) developable charts. The original mesh surface though is required to already interpolate the given boundary constraints. In their work primarily applicable to architectural design, Liu et al. presented a development algorithm based on the use of planar quad strips [24]. Despite its promise as a modeling and design tool, this algorithm nevertheless is not capable of dealing with general developable surface interpolation problem.

In light of the above review, we state the following unique characteristics of our algorithm:

1. It does not impose any particular limitations on the boundary constraints.
2. The interpolation mesh surface has G^1 continuity.
3. Unlike most other energy-minimization approaches, ours relies on solving a sparse and overdetermined linear system whose numerical stability and also the overall convergence of the iterations are considered high.
4. With the boundary constraints strictly met, the interpolation mesh surface usually attains high developability, as exemplified by our experimental results.

3 Preliminaries

The input to our quasi-developable mesh surface interpolation algorithm consists of two items: the initial polygonal mesh model M, and a set Φ of points in space that a subset of the vertices of M, called *anchor vertices*, must eventually coincide with. Mesh M is defined by its vertices set $V = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$, the edges set E, and the faces set F (in other words, the topological graph of M). Each vertex $\boldsymbol{v}_i$ in V is represented by its x, y, and z coordinates, i.e., $\boldsymbol{v}_i = [x_i, y_i, z_i]^T$. Set $\Phi = \{\boldsymbol{P}_1, \boldsymbol{P}_2, \ldots, \boldsymbol{P}_m\}$ specifies m points in space that the anchor vertices on the final deformed mesh are required to coincide respectively with $m << n$. Hereinafter, without loss of generality, it will be assumed that the coincidence correspondence specifies that, on the final deformed mesh, vertex $\boldsymbol{v}_i$ must coincide with $\boldsymbol{P}_i$, for $i = 1, 2, \ldots, m$. The final deformed mesh will be denoted as M', and the intermediate meshes in the deformation process from M to M' are M^i for $i = 0, 1, 2, \ldots, N$ for some N, with $M^0 = M$ and $M^N = M'$. The following entities/terminologies are defined.

Discrete Gaussian curvature on a polygonal mesh. From elementary differential geometry [5], the developability of a C^1 surface is easily determined by its Gaussian curvature, i.e., a C^1 surface is developable if and only if its Gaussian curvature is zero everywhere on the surface. In our problem, since the surface is a polygonal mesh, an approximation is needed for measuring the Gaussian curvature $K(\boldsymbol{v})$ at a vertex $\boldsymbol{v}$. We adopted the following simple formula [25] that has been proven to be accurate enough in practice:

$$K(\boldsymbol{v}) = \frac{1}{A_{\boldsymbol{v}}}\left(2\pi - \sum_j \theta_j\right),$$

where A_v is the "Voronoi area" surrounding $\boldsymbol{v}$, the shaded region in Fig. 3, and θ_j are the incident internal angles at $\boldsymbol{v}$. Thus, $|K(\boldsymbol{v})|$ indicates the degree of developability—the smaller it is, the more developable the mesh surface will be, with 0 indicating 100 percent developability. For vertices on the boundary of the mesh, they are considered to be developable, i.e., their K are 0.

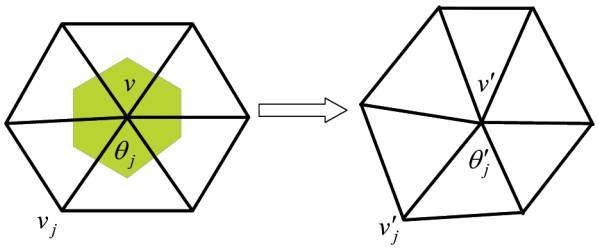

Fig. 3. The inner angle before and after the deformation around a vertex v.

Measure of developability. In the deformation process from M to M', the developability of an intermediate mesh M^k is represented by the summation $E_D = \sum_{v \in V_I} K^2(v)$ over all the internal vertices of M^k, where V_I denotes the set of the internal vertices of M. The deformation gradually, though not monotonely, reduces E_D such that at termination the final mesh M' has a minimum E_D.

Measure of interpolation error. The final interpolation constraint requires that vertex $\boldsymbol{v}_i$ of M' identifies with $\boldsymbol{P}_i$, that is, $\boldsymbol{v}_i = \boldsymbol{P}_i$, for $i = 1, 2, \ldots, m$. The total interpolation error $E_I = \sum_{i=1}^{m} \|\boldsymbol{v}_i - \boldsymbol{P}_i\|^2$ of an intermediate M^k measures as a whole how close the vertices $\{\boldsymbol{v}_i : i = 1, 2, \ldots, m\}$ are to their target positions $\{\boldsymbol{P}_i : i = 1, 2, \ldots, m\}$. Similar to the measure of developability, the deformation process gradually decreases the total interpolation error E_I until it becomes zero or the termination conditions of the iteration are met.

Measure of length preservation. In addition to the above two measures, we take into account another measure, $E_L = \sum_{e \in E} \|\ell(e) - \ell_0(e)\|^2$, where $\ell(e)$ and $\ell_0(e)$ are the lengths of edge e on the current mesh M^i and the original M, respectively. The measure E_L serves several important purposes. First, since any developable surface is an isometry of its unfolded counterpart in the plane, the final mesh surface M' should be isometric to the original M if M is developable (e.g., a planar figure), i.e., $E_L = 0$. Second, even in the case that M is nondevelopable or the interpolation constraints forbid the isometry between M and M', for instance, when the two end points of edge e are anchor points and the length of e on M is different from that in set Φ, the inclusion of the minimization of E_L in the

solution process reflects the intention of trying to maintain the "similarity" between the final developable M' and the original M. Finally, as to be described in Section 4, our least squares method eventually results in a system of linear equations whose solution requires that the system be overdeterminant, and the inclusion of E_L helps meet this requirement.

The final weighted form. Combining all the three measures together, the final weighted total energy is $E_W = w_1E_D + w_2E_L + w_3E_I$, where the three weights w_1, w_2, and w_3 are greater than 0. The deformation process, starting from the original M, gradually moves the positions of the vertices in V to reduce E_W and eventually reaches the state when both E_D and E_I are minima. Therefore, we aim at solving the following minimization problem:

$$\underset{V}{\boldsymbol{arg\ min}}(w_1E_D + w_2E_L + w_3E_I). \tag{1}$$

4 Linearization and Least Square Solution

There is a total of $3n$ variables in (1), i.e., the x, y, and z coordinates of the n vertices in V. Let $\boldsymbol{X} = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n]^{\mathrm{T}}$ be a column vector representing them, with $\boldsymbol{v}_i = [x_i, y_i, z_i]^{\mathrm{T}}$. The three criteria—developability, interpolation, and length preservation—impose the following geometric constraints on $\boldsymbol{X}$:

$$\begin{aligned} K(\boldsymbol{v}_i) &= 0, \boldsymbol{v}_i \in V_I, \\ \delta(e) &= \ell(e) - \ell_0(e) = 0, e \in E, \\ \lambda(\boldsymbol{v}_i) &= \boldsymbol{v}_i - \boldsymbol{P}_i = \boldsymbol{0}, 1 \le i \le m. \end{aligned} \tag{2}$$

The left-hand sides of all the equations in (2) are functions of $\boldsymbol{X}$. These are highly nonlinear and overdetermined equations, and thus, trying to solve them directly is doomed to fail. Our approach is to iteratively linearize (2) and then solve the resultant linear system based on the powerful least squares method. In a nutshell, let $\boldsymbol{X}_0$ be the solution to (2) at the current iteration. We amend $\boldsymbol{X}$ by a small difference $\delta\boldsymbol{X}$, that is, $\boldsymbol{X}' = \boldsymbol{X}_0 + \delta\boldsymbol{X}$, such that $\boldsymbol{X}'$ is better than $\boldsymbol{X}_0$ in (globally) reducing E_W. This process is then repeated with $\boldsymbol{X}'$ as the new $\boldsymbol{X}_0$ until a satisfactory solution is obtained for (2).

In order for our iterative process to converge, the change $\delta\boldsymbol{X}$ between iterations must be extremely small. This would be quite inefficient since the anchor points on the initial mesh surface M are usually far away from their target positions in Φ. To speed up the deformation, we introduce a scale vector $\boldsymbol{s}_i = [\boldsymbol{s}_i^x, \boldsymbol{s}_i^{\mathrm{y}}, \boldsymbol{s}_i^z]^T$ for each vertex. At the current iteration with $\boldsymbol{X} = \boldsymbol{X}_0$, rather than $\boldsymbol{X}$ itself, all the left-hand sides in (2) are expressed in terms of $\boldsymbol{s}_i$ in the form of $\boldsymbol{X} = \boldsymbol{X}_0 + \boldsymbol{S} \otimes \boldsymbol{X}_0$, where $\boldsymbol{S} = [s_1^x, s_1^y, s_1^z, \ldots, s_n^x, s_n^y, s_n^z]^T$ and "$\otimes$" stands for the elementwise multiplication between two arrays or vectors of same dimensions or size. Before the deformation process begins, a large constant is added to the coordinates of all the vertices in V (and the set Φ as well). As a result, a small variation in $\boldsymbol{S}$ can still result in a large change in $\boldsymbol{X}$, so as to converge more quickly to the target positions.

4.1 Linearization

The left-hand sides in (2) need to be linearized around the current $\boldsymbol{X}$ in the iteration process, as described next.

Linearization of $K(\boldsymbol{v}_i)$. Each $K(\boldsymbol{v}_i)$ is a nonlinear function of the coordinates of vertex $\boldsymbol{v}_i$ and all of its one-ring neighboring vertices, as shown in Fig. 3. However, after introducing the scale vectors $\boldsymbol{S}$, we instead rewrite $K(\boldsymbol{v}_i)$ as a function in terms of all the involved vectors $\boldsymbol{s}_i$, i.e., $K(\boldsymbol{v}_i) = K(\boldsymbol{s}_i^T, \boldsymbol{s}_{i1}^T, \boldsymbol{s}_{i2}^T, \cdots)$, where $\boldsymbol{s}_{i1}, \boldsymbol{s}_{i2}, \ldots,$ are the scale vectors of the neighboring one-ring vertices of $\boldsymbol{v}_i$. (Note that the current $\boldsymbol{X}$, $\boldsymbol{X}_0$, is taken as a constant in the linearization.) Let $\boldsymbol{\rho}_i = [\boldsymbol{s}_i^T, \boldsymbol{s}_{i1}^T, \boldsymbol{s}_{i2}^T, \cdots]^T$ denote the vector of all the involved scale vectors of vertex $\boldsymbol{v}_i$. Thus, $K(\boldsymbol{v}_i) = K(\boldsymbol{\rho}_i)$, and, in particular, $\boldsymbol{\rho}_i = 0$ represents the current solution $\boldsymbol{X}_0$. Assuming the delta $\Delta\boldsymbol{\rho}_i$ to be small, we then use the Taylor expansion to linearize $K(\boldsymbol{\rho}_i)$ around $\boldsymbol{\rho}_i = 0$, i.e.,

$$K(\boldsymbol{\rho}_i) = K_i(\boldsymbol{0}) + \nabla K_i(\boldsymbol{0})^{\mathrm{T}}\boldsymbol{\rho}_i + O\Big(\|\boldsymbol{\rho}_i\|^2\Big), \tag{3}$$

where $K_i(\boldsymbol{0})$ denotes the K of $\boldsymbol{v}_i$ at the current $\boldsymbol{X}_0$. Note that in (3), we used $\boldsymbol{\rho}_i$ instead of $\Delta\boldsymbol{\rho}_i$ since the expansion is carried out around $\boldsymbol{\rho}_i = 0$. Regarding the calculation of the gradient $\nabla K(\boldsymbol{v}_i)$ at $\boldsymbol{X} = \boldsymbol{X}_0$, since it is quite complex, it is calculated numerically using the standard central difference scheme.

Linearization of $\delta(\boldsymbol{e})$. For an edge e, its length is determined by its two vertices $\boldsymbol{v}_{i1}$ and $\boldsymbol{v}_{i2}$, i.e., $\ell(e) = \|\boldsymbol{v}_{i1} - \boldsymbol{v}_{i2}\|$. Similar to $K(\boldsymbol{\rho}_i)$, the $\delta(e)$ is expressed as $\delta(\boldsymbol{\rho}_e)$, with $\boldsymbol{\rho}_e = [\boldsymbol{s}_{i1}^{\mathrm{T}}, \boldsymbol{s}_{i2}^{\mathrm{T}}]^{\mathrm{T}}$, which is linearized via Taylor expansion as

$$\delta(\boldsymbol{\rho}_e) = \delta_e(\boldsymbol{0}) + \nabla\delta_e(\boldsymbol{0})^{\mathrm{T}}\boldsymbol{\rho}_e + O\Big(\|\boldsymbol{\rho}_e\|^2\Big), \tag{4}$$

where the gradient $\nabla\delta$ can be analytically calculated as

$$\nabla\delta_e(\boldsymbol{0}) = \frac{(\boldsymbol{v}_{i1}^{\mathrm{T}}, -\boldsymbol{v}_{i2}^{\mathrm{T}})}{\|\boldsymbol{v}_{i1} - \boldsymbol{v}_{i2}\|}.$$

Linearization of $\lambda(\boldsymbol{v}_i)$. Compared to the other two, the linearization of $\lambda(\boldsymbol{v}_i)$ is the simplest, since it only depends on the scale vector $\boldsymbol{s}_i$ (the anchor point $\boldsymbol{P}_i$ is a constant), and this condition is originally linear as

$$\lambda(\boldsymbol{s}_i) = \lambda_i(\boldsymbol{0}) + \boldsymbol{v}_i^{\mathrm{T}}\boldsymbol{s}_i. \tag{5}$$

4.2 Least Square Solution

With the second-order terms omitted and the current solution $\boldsymbol{X}_0$ taken as a constant, all the equations in (3), (4), and (5) are linear equations of the *scale vector* $\boldsymbol{S}$. Setting these three equations to be zero and moving those constant terms to the right-hand side, we construct a large sparse linear equation system as

$$\begin{aligned} L\boldsymbol{S} &= \begin{pmatrix} A \\ B \\ C \end{pmatrix}\boldsymbol{S} = \begin{pmatrix} w_1\Big[\nabla K_i(\boldsymbol{\rho}_i)^{\mathrm{T}}\Big] \\ w_2\Big[\nabla\delta_e(\boldsymbol{0})^{\mathrm{T}}\Big] \\ \big[\boldsymbol{v}_i^{\mathrm{T}}\big] \end{pmatrix}\boldsymbol{S} = \boldsymbol{b} \\ &= \begin{pmatrix} -w_1[K_i(\boldsymbol{0})] \\ -w_2[\delta_e(0)] \\ -\lambda_i(\boldsymbol{0}) \end{pmatrix}, \end{aligned} \tag{6}$$

where A is an $n_1 \times 3n$ matrix (n_1 is the number of nonboundary vertices in V), B is an $n_2 \times 3n$ matrix (n_2 is the number of edges in E), C is a $3m \times 3n$ matrix, and $\boldsymbol{b}$ is an $(n_1 + n_2 + 3m) \times 1$ column vector. The two control coefficients w_1 and w_2 are specified by the user or automatically determined by the program, which balance

the weight among the developability, interpolation, and length preservation. (Note that the original formulation (1) requires a third coefficient, w_3, which is set to a constant 1 by us, since the effect of the three are all relative to each other.) Their values play an important role in program efficiency as well as the final result, and the strategy for their choices will be explained in Section 5.

As $n_1 + n_2 + 3m > 3n$, the linear system (6) is overdetermined and, hence, any attempt at finding the exact solution would be futile. We solve it in the least square sense:

$$\boldsymbol{S} = (L^T L)^{-1} L^T \boldsymbol{b}. \tag{7}$$

It is noted that the matrix $L^T L$ is sparse, positive definite, and symmetric. Therefore, it is possible to compute a Cholesky factorization of $L^T L$, which helps solve (6) more efficiently, that is, $L^T L = R^T R$, where R is an upper triangular sparse matrix. After the factorization, $\boldsymbol{S}$ is obtained by back substitutions:

$$\begin{aligned} R^T \mathrm{Y} &= L^T \boldsymbol{b}, \\ R\boldsymbol{S} &= \mathrm{Y}. \end{aligned} \tag{8}$$

The above least-squares-based approach crucially depends on the overdeterminacy of (6). In most cases, the condition "$(n_1 + n_2 + 3m) > 3n$" is met. However, in case (6) is underdetermined—this is rare, but it could happen, e.g., when the number m of anchor points is very small—we make it overdetermined by means of reducing the degree of (6). Explicitly, the linear system (6) this time is solved independently and separately for the $\{\boldsymbol{s}_i^x\}$, $\{\boldsymbol{s}_i^y\}$, and $\{\boldsymbol{s}_i^z\}$, each with n unknowns. Accordingly, after this adjustment, A will be an $n_1 \times n$ matrix, B an $n_2 \times n$ matrix, C an $m \times n$ matrix, and $\boldsymbol{b}$ an $(n_1 + n_2 + m) \times 1$ column vector. More specifically, this time in (6), L will be an $(n_1 + n_2 + m) \times n$ matrix, and the number of unknowns is n. Apparently, $(n_1 + n_2 + m) > n$. On the flip side, however, since this time the x, y, and z coordinates of V are treated independently of each other, the convergence speed may become faster, but the final optimization result might be affected. Fortunately, as already alluded, the condition "$(n_1 + n_2 + 3m) > 3n$" is usually met in practice.

5 The Final Algorithm

The final algorithm, **UpdateMesh**, for the optimization problem (1), is of recursive type: it takes as input the x, y, and z coordinates, $\boldsymbol{X}_0$, of the current mesh M^k, makes an amendment to $\boldsymbol{X}_0$, and calls itself again with the new $\boldsymbol{X}_0$, until the termination criteria are met.

UpdateMesh. $(\boldsymbol{X}_0)$.
Begin
- Step 1: If (Termination) then Exit
- Step 2: Determine weights w_1 and w_2
- Step 3: Linearize (2) as specified by (3), (4), and (5)
- Step 4: Construct the linear system (6)
- Step 5: Use the least squares algorithm as specified in (7) and (8) to solve the overdetermined linear system from Step 4 (the solution is $\boldsymbol{S}$)
- Step 6: $\boldsymbol{X}_0 \leftarrow \boldsymbol{X}_0 + \boldsymbol{X}_0 \otimes \boldsymbol{S}$
- Step 7: Call **UpdateMesh** $(\boldsymbol{X}_0)$

End

As seen in the algorithm, the weights w_1 and w_2 are not static—they are decided dynamically in every iteration. Conceivably, too large w_1 and w_2 (with respect to the constant $w_3 = 1$) may diminish the influence of E_I in the total energy E_W and consequently prolong the process of moving the anchor vertices to their target positions. On the other hand, if w_1 and w_2 are too small, the anchor vertices might quickly move to the target positions, while E_D and E_L still remain relatively large, which again protracts the deformation process since the iteration has to continue in order to bring E_D and E_L down.

Ideally, as the initial mesh M is usually a flat figure, we hope that the developability and length preservation can be maintained throughout the deformation, while the anchor vertices gradually move toward their target positions. The algorithm **AdjustParameter** given next realizes this dynamic adjustment of w_1 and w_2, which is called in Step 2 of **UpdateMesh**. The w_1 and w_2 for the very first execution of **UpdateMesh** are assigned with a relatively large value ηE_W, where $\eta = 50$ in our implementation. Because w_1 and w_2 are determined based on the progress status of the deformation, in addition to the current $\boldsymbol{X}_0$, E_D, E_L, and E_I, we also need their counterparts in the previous iteration, i.e., X_0^{-1}, E_D^{-1}, E_L^{-1}, and E_I^{-1}.

AdjustParameter. $(\boldsymbol{X}_0, E_D, E_L, E_I, X_0^{-1}, E_D^{-1}, E_L^{-1}, E_I^{-1})$.
Begin
Step 1.1: If $((E_I - E_I^{-1}) < \delta_I$ and $E_I > \varepsilon_I)$ do {
$w_1 \leftarrow w_1/2$, $w_2 \leftarrow w_2/2$
}
Step 1.2: Else, if $((|E_L - E_L^{-1}|/E_L^{-1}) > \delta_{L+})$ do {
$w_2 \leftarrow w_2/2$, $\boldsymbol{X}_0 \leftarrow X_0^{-1}$
}
Step 1.3 Else, if $((|E_L - E_L^{-1}|/E_L^{-1}) < \delta_{L-}$ and $E_L > \varepsilon_L)$ do {
$w_2 \leftarrow 2w_2$
}
Step 1.4: Else, if $((E_D - E_D^{-1}) < \delta_D$ and $E_D > \varepsilon_D)$ do {
$w_1 \leftarrow 2w_1$
}
End

In **AdjustParameter**, all the δ_I, δ_{L+}, δ_{L-}, δ_D, ε_I, ε_L, and ε_D are preset system constants. Since our first priority is interpolation, we first check E_I, and if it remains high and its reduction rate is low, increase its weight w_3 (by reducing w_1 and w_2) when necessary (Step 1.1). Provided that E_I has decreased satisfactorily, we next (Step 1.2) examine the change of E_L. If E_L varies too quickly, it may cause oscillation and even affect adversely the shape quality of the final mesh; in such case, w_2 is reduced, the current solution $\boldsymbol{X}_0$ discarded, and the deformation restarts at the previous $\boldsymbol{X}$. On the other hand, if E_L changes too slowly and is still large (Step 1.3), its weight is increased. In our current implementation, the ratio δ_{L+}/δ_{L-} is set to 15. Finally (Step 1.4), the weight for E_D is increased if needed. It is noted that all the four steps Step 1.1 through Step 1.4 are mutually exclusive of each other, which ensures their intended individual effect.

The terminal condition (Step 1 in **UpdateMesh**) is relatively simple in our current implementation—the deformation will stop if either (condition A) all the three energy components are below their thresholds, that is, $E_I < \varepsilon_I$, $E_L < \varepsilon_L$, and $E_D < \varepsilon_D$, or (condition B) the number of cumulative executions of **UpdateMesh** has reached a maximum.

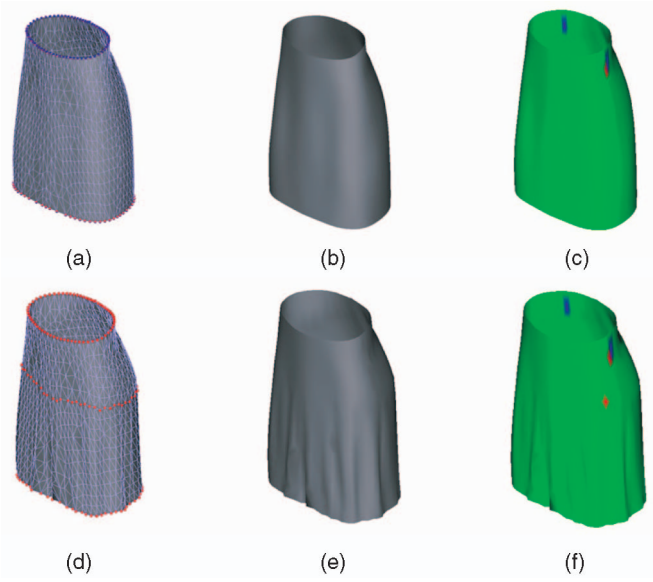

Fig. 4. Example 1 (skirt design). (a) The original mesh. (b) Its shaded image. (c) The Gaussian curvature K distribution. (d), (e), and (f) The deformed mesh and its Gaussian curvature distribution. The red points are anchor points.

6 Experimental Results and Discussion

We have implemented a prototype of the proposed quasi-developable mesh interpolation algorithm and applied it to a batch of test examples. All the tests were performed on a P4, 3.0G, and 1 Gbyte of RAM PC. In this section, several test results are given. To gauge the efficacy of the proposed algorithm, aside from E_I and E_L, two new measures—$E_D^{mean} = \frac{1}{n_1}\sum_{v\in V_I}|K(v)|$ and $E_D^{\boldsymbol{max}} = \boldsymbol{max}\{|K(v)| : v \in V\}$—are adopted in place of the original E_D, as they are believed to be better estimates of the overall developability of a mesh surface. In addition, since developability preserves surface area, we also included the area change into the test data, i.e., $E_A = |\mathrm{Area}(M') - \mathrm{Area}(M)|/\mathrm{Area}(M)$.

In the first test, which is a clothing design example, certain points on the original skirt (Fig. 4) are designed to move to some new locations (red colored), so to realize certain atheistic effect (e.g., wrinkles). As this concerns garment, developability is required. Figs. 4d, 4e, and 4f show the result of our algorithm, and the corresponding statistics data are given in Table 1 under Example 1. From the data, one sees that the interpolation requirement (E_I) is met wonderfully and all the other measures are also satisfactory within tolerance.

The second example, which is also pertinent to garment design, is used to illustrate how our algorithm can be utilized to improve the developability of a mesh surface. The original design (Fig. 5a) incurs high Gaussian curvature at various points (Fig. 5b). With certain key design points fixed as anchor points (the red colored), the mesh underwent deformation with an application of our developable interpolation program and the resultant mesh (Fig. 5d) is seen to have eliminated high Gaussian curvature in most places (Fig. 5e). Referring to Table 1, under Example 2, the average Gaussian curvature is reduced more than 30 times and the maximum Gaussian curvature 3 times as well. The length and area this time are not preserved, which though is

TABLE 1
Computing Statistics of the Test Examples

Example	Number of vertices	Running time	Result Fig.	E_D^{mean}	$E_D^{\boldsymbol{max}}$	E_I	E_L	E_A	Terminal conditions
I (Fig. 4)	1261	16.2s	Top Bottom	6.35*10-6 4.72*10-6	7.85*10-3 7.81*10-3	N/A 6.41*10-3	N/A 5.13*10-3	N/A 7.12*10-3	(A)
II (Fig. 5)	1708	179.5s	Fig. 5a Fig. 5d	3.25*10-5 1.02*10-6	7.41*10-2 2.23*10-2	N/A 1.23*10-4	N/A 0.043	N/A 0.013	(A)
III (Fig. 6)	841	37.2s 47.6s 49.2s 52.9s 78.3s 90.2s	Fig. 6a Fig. 6d Fig. 6e Fig. 6f Fig. 6g Fig. 6h Fig. 6i	0 2.21*10-7 3.31*10-7 4.12*10-7 5.20*10-7 7.11*10-7 6.73*10-7	0 3.32*10-3 3.12*10-3 5.81*10-3 5.72*10-3 6.28*10-3 6.18*10-3	N/A 0.070 0.021 0.033 0.015 0.067 0.072	N/A 1.76*10-5 3.13*10-5 4.27*10-5 5.12*10-5 5.32*10-5 5.77*10-5	N/A 1.16*10-4 2.99*10-4 3.82*10-4 4.12*10-4 4.25*10-5 4.77*10-5	(A) (A) (A) (A) (A) (A)
IV (Fig. 7)	961	10.6s 36.3s 58.7s 72.5s 108.9s 128.6s 174.5s 208.3s 230.5s	Fig. 7a Fig .7c Fig. 7d Fig. 7e Fig. 7f Fig. 7g Fig. 7h Fig. 7i Fig. 7j Fig. 7k	0 7.11*10-7 3.86*10-6 4.68*10-6 1.57*10-6 1.69*10-6 1.68*10-6 1.22*10-6 2.01*10-6 3.10*10-6	0 9.23*10-4 6.63*10-3 8.07*10-3 2.98*10-3 3.33*10-3 2.11*10-3 1.03*10-3 4.23*10-3 7.10*10-3	5.69*103 2.73*103 878.36 322.87 168.93 73.21 42.90 28.87 13.23 0.028	N/A 1.21*10-3 2.72*10-3 1.33*10-3 5.01*10-3 3.11*10-3 2.07*10-3 6.63*10-3 5.12*10-3 8.33*10-3	N/A 9.21*10-3 5.21*10-3 4.54*10-3 0.010 6.22*10-3 3.15*10-3 9.01*10-3 8.82*10-3 9.3*10-3	N/A N/A N/A N/A N/A N/A N/A N/A N/A (A)
V (Fig. 8)	961	23.2s 25.1s 28.6s 28.9s	Fig. 8a Fig. 8b Fig. 8c Fig. 8d	$1.25*10^{-7}$ $1.43*10^{-7}$ $3.12*10^{-7}$ $2.73*10^{-7}$	$3.15*10^{-3}$ $4.11*10^{-3}$ $5.17*10^{-3}$ $4.91*10^{-3}$	0.014 0.017 0.021 0.023	$2.73*10^{-3}$ $3.91*10^{-3}$ $3.22*10^{-3}$ $3.12*10^{-3}$	$2.25*10^{-3}$ $5.12*10^{-3}$ $6.03*10^{-3}$ $6.02*10^{-3}$	(A) (A) (A) (A)

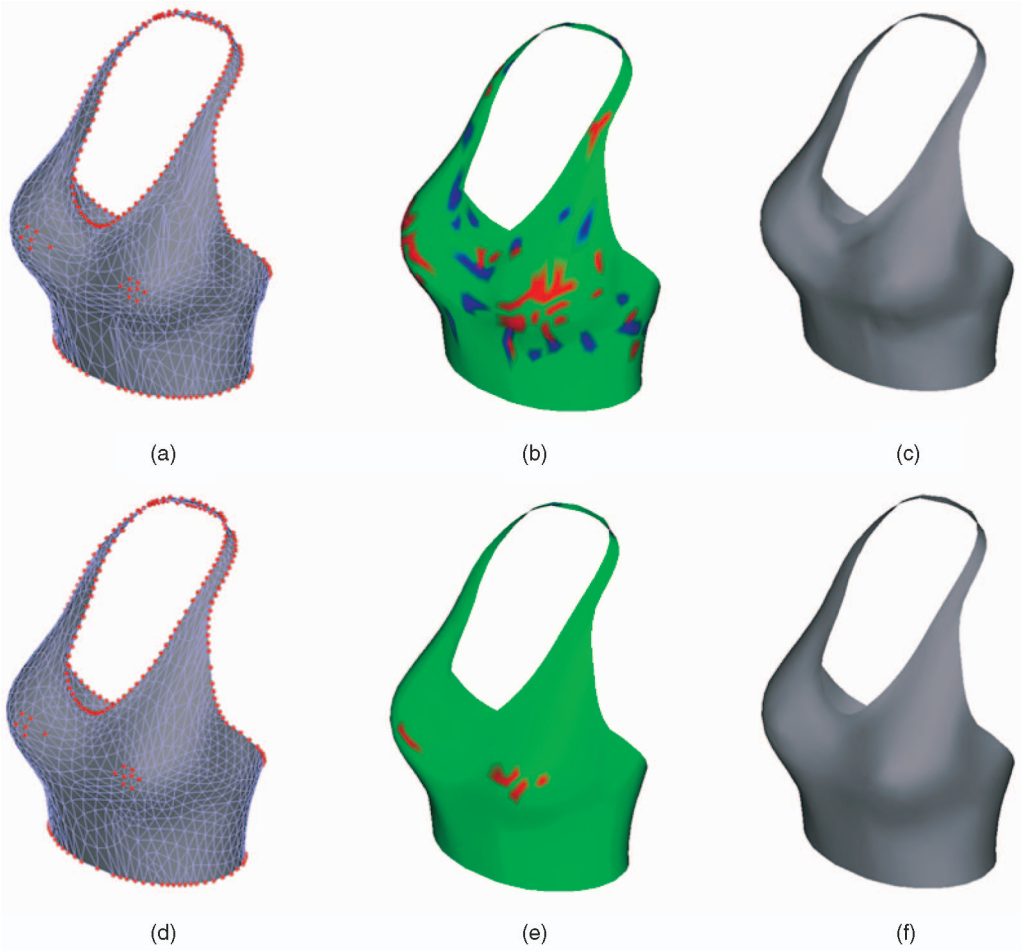

Fig. 5. Example 2 (swimsuit design). (a), (b), and (c) The initial design and its Gaussian curvature. (d), (e), and (f) The new design and its Gaussian curvature, obtained by applying our developable interpolation algorithm to the initial design, with the red colored vertices fixed as anchor points.

of no concern in this particular case, as the goal of the design is developable interpolation, while the original mesh (Fig. 5a) acts only as an initial concept design.

Example 3 (Fig. 6) simulates the deformation of a square piece of paper under a classical boundary condition: two adjacent sides of the paper (red colored) are stapled on a table, while their opposite corner is raised. Here, we give six different target positions for the corner point and show the corresponding shapes of the paper after the application of the proposed algorithm. As the original model (Fig. 6a) is flat and, hence, 100 percent developable, any *developable* deformation of it must be an isometry of itself. The data in Table 1 clearly ratify our result in this regard: in all the six cases, all the measures E_D^{mean}, $E_D^{\max}$, E_I, E_L, and E_A can be considered satisfactory within tolerance.

The next example, Example 4, demonstrates the application of the proposed algorithm in cloth simulation: a nonstretchable piece of cloth is placed on a round table, and it will drape under the gravity. Most existing methods for this kind of simulations are physically based (cf., [26], [27], [28], [29], [30], [31], and [32]), which usually require extremely long computing time and in most cases are not capable of maintaining developability on the cloth since the cloth is always assumed to be elastic (extensible). In their recent impressive work [33], Goldenthal et al. improved it by forbidding the extensibility in two mutually orthogonal directions, warp and weft, on the cloth; nevertheless, the developability is neither explicitly addressed nor analyzed in [33]. As an alternative, our algorithm offers a pure geometric solution to this task. Referring to Fig. 7a, the cloth is initially flat and placed on the table, on which the vertices in the center (which overlap with the table) and certain key points on the edge are taken as anchor points (red colored); the blue points are the target positions for those anchor points. Fig. 7b shows the final deformed cloth that satisfies the constraint of these anchor points. To help better glimpse the progressive nature of the algorithm, in Figs. 7c, 7d, 7e, 7f, 7g, 7h, 7i, 7j, and 7k, we also show some intermediate meshes generated during the execution of our program for the result in Fig. 7b, and the corresponding computing statistics data are given in Table 1. The data convincingly demonstrate the plausible characteristic of the proposed deformation algorithm for an already developable initial mesh: the developability—measured by E_D^{mean}, $E_D^{\max}$, E_L, and E_A—is maintained throughout the deformation process, while the anchor points are gradually dragged to their target positions (i.e., E_I, continuously decreases).

As a comparison, in the Laplacian deformation example shown in Fig. 2 in which the same initial mesh and the same interpolation conditions as in Example 4 were used, the

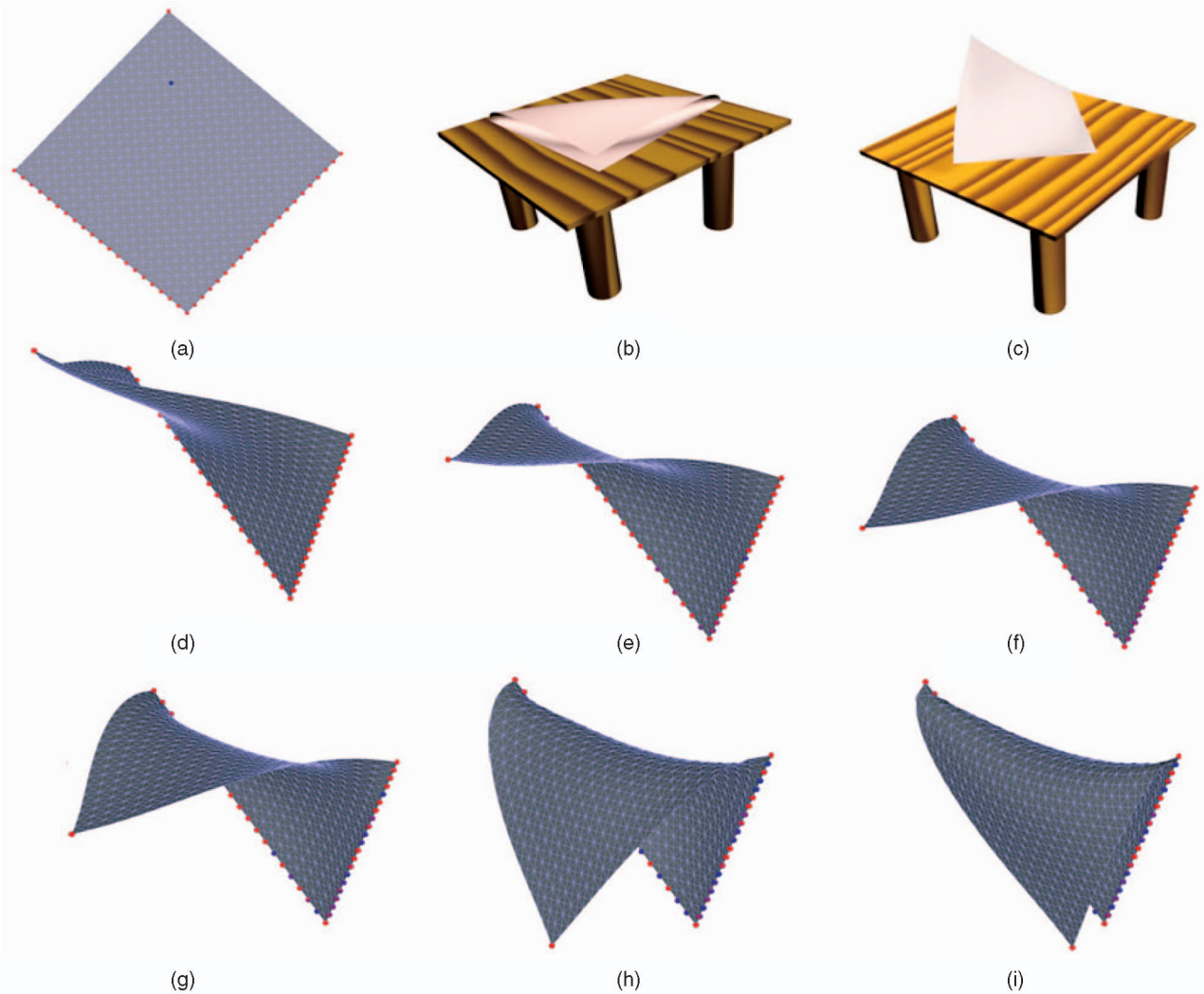

Fig. 6. Example 3 (simulation of paper bending). (a), (b), and (c) A square piece of paper whose two sides are stapled on a table, while their opposite corner is raised. (d)-(i) The deformed shapes corresponding to various positions of the moved corner.

final deformed mesh has a 25 percent reduction in total surface area, and at some vertices, the internal angle is smaller than 250 degree (for a developable surface, it must be 360 degree).

The final example, shown in Fig. 8, which is also about cloth simulation, serves to show the usage of the proposed method for modeling wrinkles. On an initially flat square piece of cloth, with two of its adjacent sides fixed, a few points on the other two sides are marked as anchor points and are moved to some different target positions, thus generating different wrinkle patterns on the cloth. In all the four resultant patterns, Figs. 8a, 8b, 8c, and 8d, as seen from the data in Table 1, all the measures E_D^{mean}, $E_D^{\max}$, E_L, and E_A are satisfactory within tolerance.

7 Summary and Future Work

The primary objective of the paper is to introduce a new algorithm for the general problem of quasi-developable mesh surface interpolation—how to find a most "developable" G^1 mesh surface to interpolate an arbitrary set of points and space curves—and report our initial experimental results to validate the promise of the introduced algorithm. Because of the arbitrariness allowed on the points and curves to be interpolated and the strictness on the interpolation (i.e., the interpolation is required to be exact and the surface be G^1), most existing methods are incapable of solving the problem. We formulate the problem as a mesh deformation problem by transforming an initially developable mesh surface gradually toward the given points and polylines until certain designated vertices (called anchor vertices) reach their targets. The developability of the final surface is achieved by maintaining the zero-valued Gaussian curvature on the intermediate surface throughout the deformation process. Though in principle still an energy minimization, our algorithm does not explicitly minimize the total energy. Instead, the highly nonlinear constraints due to the developability and interpolation requirements are linearized and then solved by a robust least squares approach. Our initial experiments show that the proposed algorithm is in general able to fulfill the exact interpolation requirement and, at the same time, attains high developability on the final mesh surface. In addition, the final surface usually exhibits good shape quality.

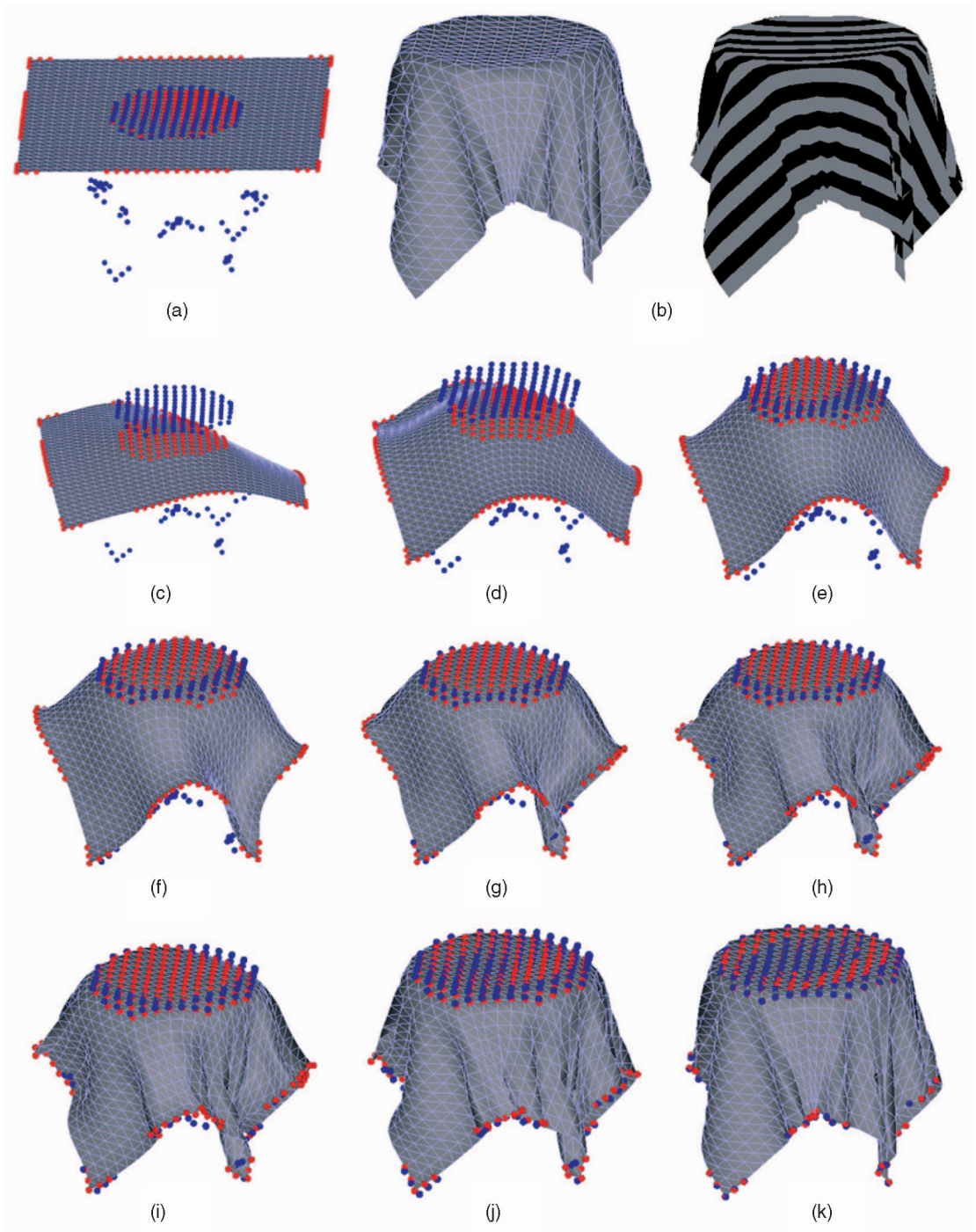

Fig. 7. Example 4 (simulation of draping). (a) and (b) The initial and final shape of the cloth; red vertices are the anchor points, and blue points are their target positions. (c)-(k) Some intermediate shapes of the mesh during the deformation toward Fig. 7b.

Currently, the correspondence between the points and polylines to be interpolated and their designated anchor vertices are specified by the user. Different correspondences would produce different results, for a same set of points and polylines. Although in some applications this correspondence is obvious and easy to decide (e.g., Examples 1-3 in Figs. 4, 5, and 6, or for instance if the correspondence is already known, and we want to try different target positions for design evaluation purpose), determining a "good" correspondence is not that intuitive. A further study on this subject is needed. Another interesting topic is the effect of the surface area of the initial surface on the final surface. Conceivably, with respect to the same correspondence, similar initial meshes but of different surface areas will generate different results under our algorithm, which though is perfectly normal. This however suggests a plausible idea: the surface area of the initial surface can act as the stiffness of the final shape—the less the surface area, the stiffer the final shape. We have done some tests along this line. Nevertheless, more investigation is required, in particular for extreme situations when the surface area is very small or very large.

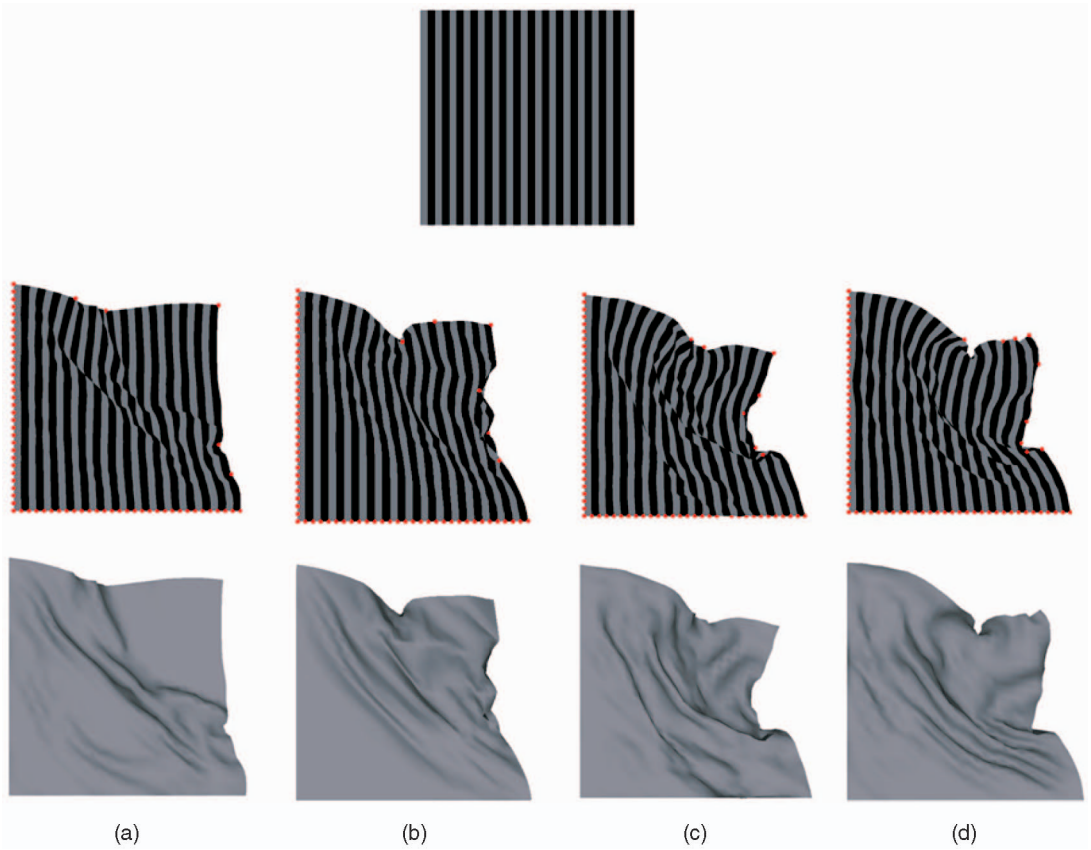

Fig. 8. Example 5 (wrinkle modeling). (top) The original flat cloth. (a)-(d) The four wrinkle patterns. The red colored points are anchor points.

ACKNOWLEDGMENTS

The partial support for this work from Hong Kong CERG RGC07/08 EG.620307 is appreciated. The authors also thank the reviewers for their helpful comments.

REFERENCES

[1] A. Hill, "Constructivism—The European Phenomenon," *Studio Int'l,* vol. 171, pp. 140-147, 1966.
[2] T. Akgun, A. Koman, and E. Akleman, "Developable Sculptures of Ilhan Koman," *Proc. Bridges,* 2006.
[3] O.K.-C. Au, "Differential Techniques for Scalable and Interactive Mesh Editing," PhD dissertation, Computer Science, HKUST, 2007.
[4] O.K.-C. Au, H. Fu, C.-L. Tai, and D. Cohen-Or, "Handle-Aware Isolines for Scalable Shape Editing," *ACM Trans. Graphics,* vol. 26, no. 3, pp. 83:1-83:10, 2007.
[5] M. doCarmo, *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.
[6] H. Pottmann and J. Wallner, *Computational Line Geometry.* Springer, 2001.
[7] K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert, "Developable Surfaces from Arbitrary Sketched Boundaries," *Proc. Fifth Eurographics Symp. Geometry Processing (SGP '07),* pp. 163-172, 2007.
[8] M. Peternell, "Developable Surface Fitting to Point Clouds," *Computer Aided Geometric Design,* vol. 21, no. 8, pp. 785-803, 2004.
[9] 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, no. 2, pp. 110-124, 1999.
[10] G. Aumann, "Interpolation with Developable Bézier Patches," *Computer Aided Geometric Design,* vol. 8, no. 5, pp. 409-420, 1991.
[11] G. Aumann, "A Simple Algorithm for Designing Developable Bézier Surfaces," *Computer Aided Geometric Design,* vol. 20, nos. 8/9, pp. 601-619, 2003.
[12] G. Aumann, "Degree Elevation and Developable Bézier Surfaces," *Computer Aided Geometric Design,* vol. 21, no. 7, pp. 661-670, 2004.
[13] C.H. Chu and C.H. Séquin, "Developable Bézier Patches: Properties and Design," *Computer-Aided Design,* vol. 34, no. 7, pp. 511-527, 2002.
[14] J. Lang and O. Röschel, "Developable (1, n)-Bézier Surfaces," *Computer Aided Geometric Design,* vol. 9, no. 4, pp. 291-298, 1992.
[15] H. Pottmann and G.E. Farin, "Developable Rational Bézier and B-spline Surfaces," *Computer Aided Geometric Design,* vol. 12, no. 5, pp. 513-531, 1995.
[16] P. Bo and W. Wang, "Geodesic-Controlled Developable Surfaces for Modeling Paper Bending," *Computer Graphics Forum/Proc. Ann. Conf. European Assoc. for Computer Graphics (Eurographics '07),* vol. 26, no. 3, pp. 329-338, 2007.
[17] W. Frey, "Boundary Triangulations Approximating Developable Surfaces That Interpolate a Closed Space Curve," *Int'l J. Foundations of Computer Science,* vol. 13, pp. 285-302, 2002.
[18] W. Frey, "Modeling Buckled Developable Surface by Triangulation," *Computer-Aided Design,* vol. 36, no. 4, pp. 299-313, 2004.
[19] C. Wang and K. Tang, "Optimal Boundary Triangulations of an Interpolating Ruled Surface," *ASME J. Computing and Information Science in Eng.,* vol. 5, no. 4, pp. 291-301, 2005.
[20] 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, nos. 8-9, pp. 521-539, 2004.
[21] J. Mitani and H. Suzuki, "Making Papercraft Toys from Meshes Using Strip-Based Approximate Unfolding," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 259-263, 2005.
[22] I. Shatz, A. Tal, and G. Leifman, "Paper Craft Models from Meshes," *The Visual Computer,* vol. 22, no. 9, pp. 825-834, 2006.
[23] D. Julius, V. Kraevoy, and A. Sheffer, "D-Charts: Quasi-Developable Mesh Segmentation," *Computer Graphics Forum,* vol. 24, no. 3, pp. 581-590, 2005.
[24] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang, "Geometric Modeling with Conical Meshes and Developable Surfaces," *ACM Trans. Graphics (Proc. ACM SIGGRAPH '06),* vol. 25, no. 3, pp. 681-689, 2006.

[25] E. Magid, O. Soldea, and E. Rivlin, "A Comparison of Gaussian and Mean Curvature Estimation Methods on Triangular Meshes of Range Image Data," *Computer Vision and Image Understanding,* vol. 107, no. 3, pp. 139-159, 2007.
[26] D. Terzopoulost, J. Platt, A. Barr, and K. Fleischert, "Elastically Deformable Models," *Proc. ACM SIGGRAPH,* 1987.
[27] Y.J. Liu, K. Tang, and A. Joneija, "Modeling Dynamic Developable Meshes by Hamilton Principle," *Computer-Aided Design,* vol. 39, no. 9, pp. 719-731, 2007.
[28] S.T. Tan, T.N. Wong, Y.F. Zhao, and W.J. Chen, "A Constrained Finite Element Method for Modeling Cloth Deformation," *The Visual Computer,* vol. 15, no. 2, pp. 90-99, 1999.
[29] P. Volino, M. Courchesne, and N. Magnenat-Thalmann, "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects," *Proc. ACM SIGGRAPH,* 1995.
[30] P. Volino and N. Magnenat-Thalmann, *Virtual Clothing: Theory and Practice.* Springer, 2000.
[31] P. Volino and N. Magnenat-Thalmann, "An Evolving System for Simulating Clothes on Virtual Actors," *IEEE Computer Graphics and Applications,* vol. 16, no. 5, pp. 42-51, 1996.
[32] P. Volino and N. Magnenat-Thalmann, "Stop-and-Go Cloth Draping," *The Visual Computer,* vol. 23, no. 8, pp. 669-677, 2007.
[33] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun, "Efficient Simulation of Inextensible Cloth," *ACM Trans. Graphics (Proc. ACM SIGGRAPH '07),* vol. 26, no. 3, 2007.

Kai Tang received the BEng degree in mechanical engineering from the Nanjing Institute of Technology, Nanjing, China, in 1982 and the MSc degree in information and control engineering and a PhD in computer engineering from the University of Michigan, Ann Arbor, in 1986 and 1990, respectively. He is currently a faculty member in the Department of Mechanical Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong. Before joining HKUST in 2001, he had worked for more than 13 years in the CAD/CAM and IT industries. His research interests concentrate on designing efficient and practical algorithms for solving real-world computational, geometric, and numerical problems. He is a member of the IEEE.

Ming Chen received the BS and a master's degree in mechanical engineering from the Huazhong University of Science and Technology, P.R. China in 2001 and 2004, respectively. He is currently a PhD student in Mechanical Engineering Department, Hong Kong University of Science and Technology, Hong Kong, P.R. China. He worked as a software specialist and research assistant in software industry. His research is about solid modeling and numerical optimization.