Developable Triangulations of a Strip

Charlie C. L. Wang[1] and Kai Tang[2]

[1]Chinese University of Hong Kong, cwang@acae.cuhk.edu.hk
[2]Hong Kong University of Science and Technology, mektang@ust.hk

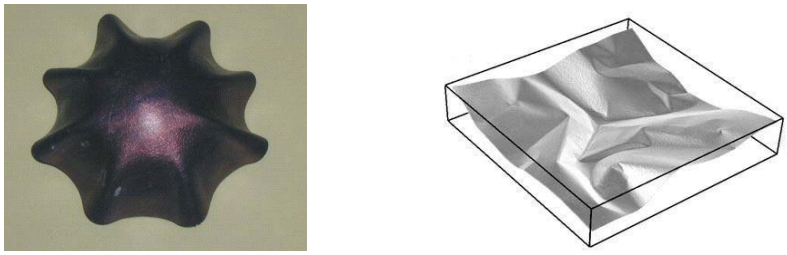
ABSTRACT

This paper presents a modeling algorithm that helps automatically generate a "most" developable triangulated surface interpolating a given closed 3D boundary curve. The boundary curve is given in the form of a narrow strip which is often encountered in CAD applications like shoe and clothing wrinkle design. The triangulation geometrically simulates the folding process of a sheet as it would occur when rolled from one end of the strip to the other. The objective of maintaining the surface developability is realized by finding the "best rate" of the rolling along the boundary curve. To achieve this optimization, we convert the problem into a single-source shortest path problem on a special directed weighted graph, and utilize the famous Dijkstra's algorithm to find a minimum-weight path. Examples are provided to demonstrate useful applications of the presented methodology in CAD.

Keywords: strip triangulation, developability, bending energy, weighted graph.

1. INTRODUCTION

Designing a surface of folding pattern over a closed 3D curve has many practical interests, especially in the *computer-aided design* (CAD) of clothes and footwear. Previous research in computer graphics studying folds focused mostly on cloth modeling or in animations, which are driven more by visual realism, but allow large elastic deformations and usually completely ignore or avoid the material issue. However, fold design in apparel and shoe industries has a critical request to consider – the designed surface must be able to be manufactured. A product in the apparel and shoe industries is usually produced by sewing several two-dimensional patterns together. In the majority of cases, the material of patterns is considered to be inextensible and uncompressible, but easy to be bent or folded. Therefore, the 3D surface of a product is considered feasible only if it can be flattened into a planar pattern without any area distortion, i.e., there exists an isometric transformation [1] that maps the surface to its 2D pattern such that the surface distance between any two points does not change in the flattening process. This is in fact the concept of developable surface in *differential geometry* [1]. Depending on the material properties, two developable wrinkled surfaces could look rather drastically different. They can be smooth (see Fig.1a) or contain creases (see Fig.1b). Since large amount of strain energy occurs at creases or places with high curvatures (cf. [2]), a sound folding surface design should naturally try to minimize or limit them.



(a) (b)

Fig. 1. Wrinkled surfaces: (a) a spherical mold of thick leather, and (b) a crumpled square of paper.

In this paper, we propose a modeling method for designing a *developable* triangular mesh surface that interpolates a closed 3D curve in the form of a narrow strip. Narrow strips are often utilized for designing wrinkle patterns. For

example, Fig. 11 shows the process of designing a wrinkled strip on a shoe upper surface: the designer first designs two three-dimensional curves (Fig. 11b) based on the boundary curve of a strip portion carved from the upper surface (Fig. 11a); a "wrinkle" surface (Fig. 11c) then is formed to interpolate these two "pattern" boundary curves; and finally this wrinkle surface is sewed back to the original upper surface of the shoe (Fig. 11d). The distance between the two boundary curves is usually much smaller compared to the lengths of the curves themselves, thus the name "narrow strip". The most natural way to interpolate such a narrow strip is to construct a developable ruled surface over its boundary curve. The triangulation modeling method proposed in the paper offers a practical solution for realizing this ruled surface. The rest of the paper is organized as follows. We first review some related work. The mathematical model underlying our developable triangulation algorithm is next presented. The triangulation is formulated as an optimization problem that is further converted into a single source-shortest path problem on a weighted graph, to which efficient optimal path-finding algorithms such as the Dijkstra's can then be applied. Finally, some experimental examples are presented to illustrate the proposed method.

2. RELATED WORKS

Wrinkle design algorithms were widely investigated in computer graphics for textile applications, where most research work focuses on how to predict the shape that a given two-dimensional piece of textile will take when draped over a particular solid form with certain given boundary conditions. Usually, the resulting shape is a wrinkled surface. Several physically-based methods were proposed for simulating the final shape of given patterns [3-5]. More successful studies used tessellated approximations of the pattern, imposed 3D boundary conditions, and used finite element techniques to iteratively compute the equilibrium state of the pattern [6]. Some more recent works on physically based cloth simulation can be found in [7-10]. There were also geometry-oriented wrinkling methods [11-14], where the nodes on a mesh are moved based on pure geometric rules.

As mentioned at the beginning of the paper, the developability of a wrinkled surface is seldom considered in computer graphics [3-14]. In differential geometry, the developable property is defined on a ruled surface, which is a surface generated by a family of straight lines. A parametric representation of a ruled surface is

$$S(u,v) = (1-v)P(u) + vQ(u) \tag{1}$$

where $P(u)$ and $Q(u)$ are two 3D curves with C^1 continuity, called *directrices*. The line passing through $P(u)$ and $Q(u)$ for any u in the parametric interval of the two curves is called a *ruling* of the ruled surface. In general, ruled surfaces are not developable. But it is well known [1] that if the rulings move along the directrices in such a way that the tangent plane to the surface remains the same at all the points along each ruling, the surface is developable. This common tangent plane condition thus distinguishes developable ones from non-developable ruled surfaces.

Base on the common tangent plane condition, several approaches to approximating developable surfaces have been investigated. In [15], Pottmann and Wallner derived linear approximation algorithms for developable NURBS surfaces. Peternell [16] developed an approach to fitting a developable surface on a cloud of data points. Recently, Frey [17] proposed using a boundary triangulation to approximate a developable surface interpolating a closed space curve. His work is mainly targeted at application in the design of the blankholder for a sheet metal in stamping, where the space curve to be interpolated is in general of small curvature. Akin to the idea of boundary triangulation, in [18], we explored using a variance, called *bridge boundary triangulation* (BBT), to interpolate a narrow strip whose boundary curves are allowed to be highly convoluted with large curvature and curvature changes. However, the solution given in [18] is only of local search nature – it can't guarantee to find a developable triangulation, if it exists. Therefore, as a further enhancement, in this paper we introduce a new bridge boundary triangulation method based on global optimal search. As to be seen, the new method converts the triangulation problem into a shortest path problem on a weighted graph, which can then be solved by means of the well-known Dijkstra's algorithm [19].

We point out that the topology construction step in our approach is somewhat similar to the contour triangulation approach as given in [20] where triangles are constructed to connect two polygons lying on two parallel planes according to the local shortest linking edge criterion. Although that simple method can be directly adopted here to triangulate a strip, because it ignores the developability, the resulting mesh is usually not satisfactory from real manufacturing point of view.

3. MATHEMATICAL MODEL

3.1 Boundary Bridge Triangulation

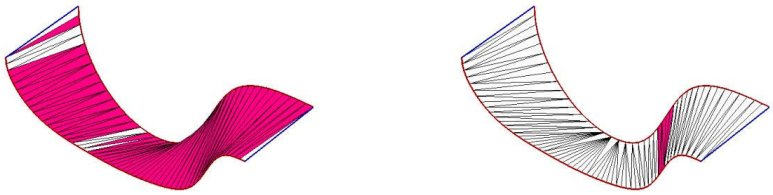
Given two C^1 continuous parametric curves $P(u)$ and $Q(u)$, we aim at constructing a developable surface $S(u, v)$ interpolating them. The $S(u,v)$ is often represented as a ruled surface in parametric form (e.g., Eq.(1)). Inspired by the classical boundary triangulation development method in descriptive geometry [21], we first introduce a special type of triangulation for $S(u,v)$. The boundary triangulation development approximates $S(u,v)$ by a collection of triangles with their vertices lying only on either $P(u)$ or $Q(u)$. Let $P=\{p_1, p_2, \cdots, p_m\}$ and $Q=\{q_1, q_2, \cdots, q_n\}$, each with m and n respectively, be two piecewise linear curves that approximate $P(u)$ and $Q(u)$. Our triangulation of $S(u,v)$ is a collection of (m+n-2) triangles. Following the terms of [18], the line segment between two adjacent vertices on the same boundary curve is defined as a *bank edge* (e.g., $B_i^P =< p_i, p_{i+1} >$ or $B_j^Q =< q_j, q_{j+1} >$); the segment linking two vertices from different curves is referred to be a *bridge edge* (e.g., $E_{i,j} =< p_i, q_j >$). Two bridge edges are *adjacent* to each other if they share a common vertex; $E_{1,1}$ and $E_{m,n}$ are respectively called the *starting* and *ending* bridge. Every triangle in our triangulation is required to be made of two adjacent bridge edges and one bank edge – thus any two neighboring triangles share a common bridge edge. This special type of triangulation on P and Q will be called a *boundary bridge triangulation* (BBT).

The construction of a BBT is actually a process of iteratively performing two topological operations – *P-succeed* and *Q-succeed* – on the given boundary curves P and Q. Specifically, the two operations are:

1) *P-succeed*: when this operator is applied to a bridge edge $E_{i,j}$, a new triangle made of three edges $E_{i,j}$, $E_{i+1,j}$ and B_i^P is formed;

2) *Q-succeed*: this operator constructs a new triangle with three edges $E_{i,j}$, $E_{i,j+1}$ and B_j^Q, when applied on a bridge edge $E_{i,j}$.

We note that to an edge $E_{m,j}$, $1 \le j < n$, only the *Q-succeed* operator can be applied; likewise, only the *P-succeed* operator can be applied to an edge $E_{i,n}$, $1 \le i < m$.



(a) (b)

Fig. 2. Example I – Two boundary bridge triangulations with different limit ruled surfaces.

3.2 Developability

When the sampling points m and n tend to infinity, the discrete approximation BBT converges to a ruled surface, called the *limit surface*. Conceivably, different BBTs have different limit surfaces. Fig. 2 shows two bridge boundary triangulations on the same boundary curves, whose limit surfaces are quite different. To obtain a developable limit ruled surface, based on the common tangent plane condition for developable ruled surfaces, Frey [17] proposed the local convexity theorem: every interior edge must be locally convex in a developable boundary triangulation. Since our bridge boundary triangulation is a special type of boundary triangulation, this theorem also applies. Explicitly in terms of the limit surface, we have the following proposition on non-inflection rulings (the detail of inflection rulings refers to [22]).

Proposition: a bridge boundary triangulation has a developable limit surface if and only if every bridge edge $E_{i,j}$ in it lies on the convex hull of the six points $\{p_{i-1}, p_i, p_{i+1}, q_{j-1}, q_j, q_{j+1}\}$.

It is imperative to distinguish between the flattenability and developability. Flattenability appraises whether a BBT can be flattened into its 2D pattern without area distortion. Since triangles are planar, it is trivial to see that a BBT is always

flattenable as long as the flattened triangles do not overlap each other in the plane. On the other hand, developability pertains to the limit surface of a BBT – a BBT is developable only if its limit surface is developable. As an example, Fig. 2 shows two BBTs of a same strip, where those edges violating the local convexity condition are colored in red. Both of them can be flattened onto the plane without any overlapping. However, in terms of developability, if one tries to fold the flattened 2D pattern of a stiff material (which implies that the final folded surface is G^1 continuous and that is the most of case in garment and footwear applications) back onto the strip, much more material failure will result for the BBT in Fig. 2a than in Fig. 2b, as exactly those red edges will incur stretching or compression. This translates into a natural desire of finding a BBT that minimizes the number of (locally) non-convex edges (e.g., Fig. 2b vs. Fig. 2a).

3.3 Minimization of Bending Energy

The above proposition tells that for a given strip we should try to find a BBT with as many locally-convex edges as possible, hopefully all of them. On the other hand, for two BBTs with a same level of developability (i.e., the same number of locally-convex edges), they can further be compared based on some other important characteristics, such as, in particular, the *bending energy*. If we view the bridge boundary triangulation as a process of rolling a sheet along the two boundary curves of the strip, then a good rolling strategy should be a one that generates little strain energy and limits extraneous surface area on the final surface. As a BBT is discrete, its strain energy is represented in the form of bending energy. Suppose that the bridge $E_{i,j}$ is shared by two adjacent triangles T_k and T_{k+1} with T_k lying in the x-y plane and $E_{i,j}$ coincident with the y-axis. Fig. 3 shows how the bending between T_k and T_{k+1} along the bridge edge $E_{i,j}$ is formed. Assuming the bending angle is very small, the energy due to this bending is (ref. [18])

$$U = \int_0^L \frac{EI(s)}{2R^2} ds \tag{2}$$

which can be further simplified into a form

$$U = K \frac{A \sin^2 \theta}{L^2} \tag{3}$$

where K is a coefficient determined by the thickness of the sheet and the Young's modulus (for detail derivations, see [18]). Since only relative value is needed for comparison purpose, we simply set K to *one*. For a boundary bridge triangulation, there are exactly $m+n$ bridge edges, so the total bending energy can be computed by

$$U = \sum_{k=2}^{m+n-1} U_k \tag{4}$$

with U_k representing the bending energy on the kth bridge edge. The first (k=1) and last bridge edges on a BBT are assumed to be in natural state, i.e., free of bending.

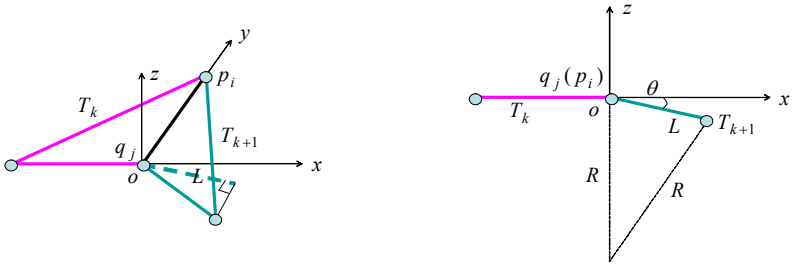


Fig. 3. Bending energy computation on a bridge edge.

3.4 Searching For An Optimal BBT

Given two directrices P and Q with m and n vertices respectively, it can be shown that (see [18]) there are a total of

$$\binom{m+n-2}{m-1} = \binom{m+n-2}{n-1} = \frac{(m+n-2)!}{(m-1)!(n-1)!}$$

distinct boundary bridge triangulations. Our task then is to find one from them that will maximize the number of locally-convex bridges edges and meanwhile at the same time achieve other desired objectives – i.e., minimum

bending energy and/or minimum surface area. Since the numbers m and n are usually very large – in order to judiciously approximate a limit surface – the simple exhaustive search is neither practical nor plausible. Better and more insightful methods thus should be sought. The approach we take is to formulate the search as a shortest path planning problem on a weighted graph, as we entail in the rest of the paper.

4. LOCAL OPTIMAL TRIANGULATIONS

In this section the sheet rolling simulation algorithm is described, which also leads to our local optimum search method. To triangulate a strip, we first discretize the two given *directrices* P and Q respectively to two polygons $P=\{p_1, p_2, \cdots, p_m\}$ and $Q=\{q_1, q_2, \cdots, q_n\}$. The sampling points p_i and q_j on P and Q can be either uniformly distributed or sampled adaptively according to a user specified approximation tolerance. After that, all the possible bridge edges except the starting and ending ones are checked for their local-convexity. A bridge edge will be deemed *valid* if it is locally-convex; otherwise, it is *invalid*. The validity of all the bridge edges can be represented by an m by n binary *validity matrix* Ψ, where the assignment $\Psi[i,j]=$"1" indicates a valid bridge between vertices p_i and q_j, and "0" otherwise. As an example, Fig. 4a shows the validity matrix of the boundary curves in Fig. 2. The sheet rolling simulation is then performed to generate triangles in a BBT from one side of the strip to the other by applying *P-succeed* and *Q-succeed* operators iteratively. The triangulation starts from the first bridge edge $E_{1,1}$ and stops at the last $E_{m,n}$; this implies that both $E_{1,1}$ and $E_{m,n}$ will always be in the BBT. Suppose at the current iteration the active bridge edge is $E_{i,j}$. To decide the next bridge edge which together with $E_{i,j}$ will form a new triangle, we apply either a *P-succeed* or a *Q-succeed* operator, depending on the validity of edges $E_{i+1,j}$ and $E_{i,j+1}$, and also a minimization criterion – i.e., whether the bending energy should be minimized. In specifics:

Local Optimal Developability Triangulation (LODT):

- If $E_{i+1,j}$ is valid but $E_{i,j+1}$ not, the *P-succeed* operator is applied; on the other hand, if only $E_{i,j+1}$ is valid, the *Q-succeed* operation is performed;
- If both $E_{i+1,j}$ and $E_{i,j+1}$ are valid, we choose the operator which generates the less bending energy on $E_{i,j}$;
- Otherwise, if none of $E_{i+1,j}$ and $E_{i,j+1}$ are valid, we find the nearest valid edge $E_{i',j'}$ in Ψ (with $i'>i$ and $j'>j$), and then perform the *Local Minimum Bending Triangulation* (**LMBT**, as defined below); this triangulation will triangulate the strip between $E_{i,j}$ and $E_{i',j'}$;
- When $j=n$, only *P-succeed* can be applied; likewise, if $i=m$, *Q-succeed* is the only choice.

(a) (b)

Fig. 4. Local optimal developability triangulation of boundary curves given in Fig. 2: (a) the validity matrix Ψ; (b) the steps of triangulation represented in green zigzag curve – result shown in Fig. 2b.

Local Minimum Bending Triangulation (LMBT): When triangulating a sub-strip between two bridge edges $E_{i,j}$ and $E_{i',j'}$ using the minimum bending energy criterion, if neither endpoints of the current bridge edge is on the ending bridge edge, we choose *P-succeed* or *Q-succeed* operator depending on which will give the less bending energy at the current bridge edge. In case the current bridge edge has one vertex on the ending bridge edge, only one appropriate operator can be applied.

As an illustrative example, the green path shown in Fig. 4b represents the sequence of operators applied in the local optimal developability triangulation that generates the BBT in Fig. 2b, where vertical segments represent *P-succeed* and horizontals the *Q-succeed*.

5. GLOBAL OPTIMAL TRIANGULATIONS

It is not hard to see that the thus prescribed **LODT**+**LMBT** algorithm is only of local optimum nature – one can not ensure that such a BBT obtains a maximum number of valid bridge edges. Fig. 5 gives a schematic example to illustrate this point: Fig. 5a depicts all the valid bridge edges, Fig. 5b is a *completely developable* BBT (i.e., all the edges are valid) out of these valid edges, and Fig. 5c could be a BBT triangulation as a result of the application of **LODT**. Notice that the BBT in Fig. 5c is inferior to that of Fig. 6b in terms of surface developability – in the shaded area the local-convexity is violated.

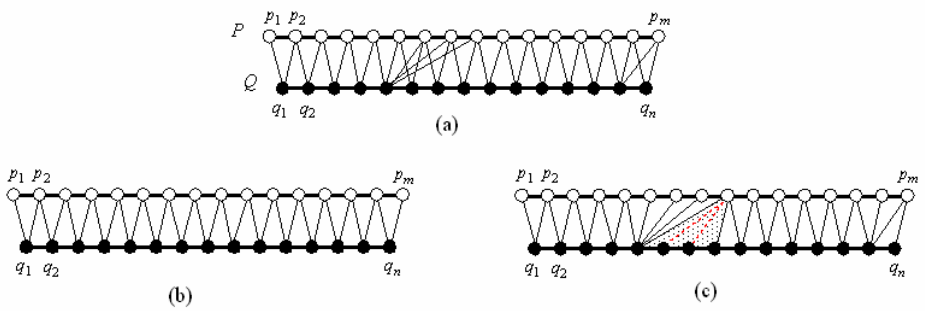


Fig. 5. Failure of algorithm **LODT** in finding the globally optimal result.

In this section, we introduce a global-optimum search algorithm which guarantees to find a completely developable BBT, if it exists. Our basic idea is to convert the triangulation problem into a single-source shortest path problem on a weighted graph; the Dijkstra's algorithm is then utilized to determine a global optimum.

$$\Psi_{m\times n} = \begin{bmatrix} 0 & 0 & 1 & \cdots & 0 & 1 & 1 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \Rightarrow$$

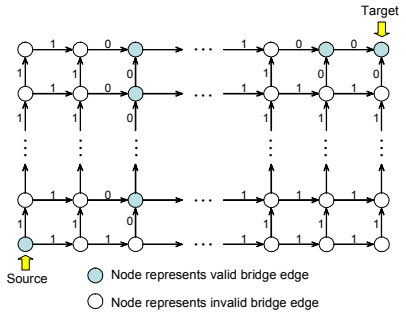


Fig. 6. Converting the validity matrix $\Psi_{m\times n}$ into a directed weighted graph Γ.

Global Optimal Developability Triangulation (GODT):

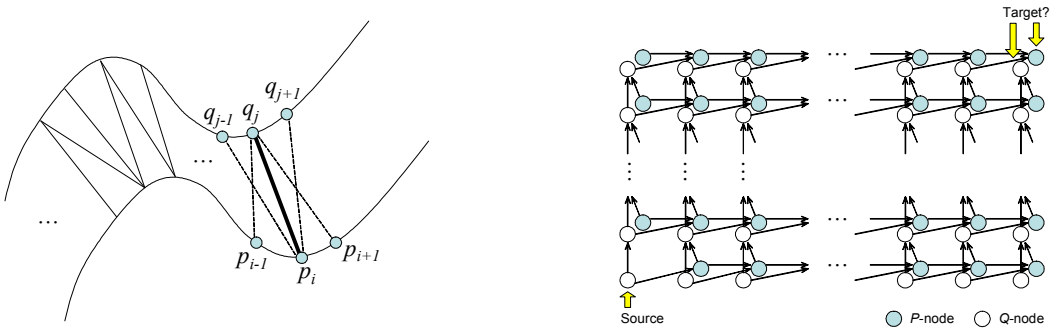
Considering the validity matrix $\Psi_{m\times n}$. Refer to Fig. 6, following the rules below, the validity matrix $\Psi_{m\times n}$ is converted into a directed weighted graph Γ:

- Every bridge edge $E_{i,j}$ corresponds to a node $V_{i,j}$ in Γ;
- A directed edge $<V_{i,j}, V_{i+1,j}>$ is defined for every pair of "horizontally" neighboring nodes, i=1,2,…, m-1; and similarly a "vertical" directed edge $<V_{i,j}, V_{i,j+1}>$ is defined for every pair of "vertically" neighboring nodes, j=1,2,…, n-1; and
- If a bridge edge $E_{i,j}$ is valid in $\Psi_{m\times n}$, both edges $<V_{i,j}, V_{i+1,j}>$ and $<V_{i,j}, V_{i,j+1}>$ are assigned of a zero weight; otherwise, the two edges have weight equal to "1".

After constructing the weighted graph, we set the node $V_{0,0}$ corresponding to a virtual bridge edge $E_{0,0}$ as the source, with both $<V_{0,0}, V_{1,0}>$ and $<V_{0,0}, V_{0,1}>$ having a "1" weight. The Dijkstra algorithm [19] is then applied to the directed graph Γ to find minimum-weight paths from $V_{0,0}$ to every other node in the graph; and among these minimum-weigh paths, the one from $V_{0,0}$ to $V_{m,n}$ determines a sequence of *P-succeed* and *Q-succeed* operations that generates a BBT with the maximal number of locally-convex bridge edges.

The **GODT** algorithm guarantees to find a BBT with the maximal number of locally-convex edges. If this number is $n+m$-3, then the thus constructed BBT is a completely developable triangulation. In most situations, nevertheless, this number is less than the ideal case, and what one gets is a *partially* developable BBT. To such a partially developable BBT, its corresponding path in Γ must contain a least one *gap* whose total weight is not zero. Explicitly, a gap is a sub-path connecting some two nodes $V_{i1,j1}$ and $V_{i2,j2}$, with $i_1 \le i_2$, $j_1 \le j_2$, and $(i_1+j_1)<(i_2+j_2)$, such that every edge on this sub-path has a "1" weight. Rather than settling with an arbitrary one, we would like to minimize the total bending energy on a gap, analogous to the local minimum bending case, as described next.

Global Minimum Bending Triangulation (**GMBT**): We will still utilize the Dijkstra's algorithm to achieve this minimization. However, compared to algorithm **GODT**, the construction of the weighted graph for the minimization of bending energy is not that straightforward. This is because that, when using Eq. (3) to calculate the bending energy at a bridge edge $E_{i,j}$, the bending energy depends not only on which of the two edges $E_{i+1,j}$ or $E_{i,j+1}$ to choose to define the ensuing triangle, but also on which of the two edges $E_{i-1,j}$ or $E_{i,j-1}$ has already be chosen that defines the current triangle with $E_{i,j}$. More specifically, as shown in Fig. 7a, there are four possible amounts of bending energy associated with $E_{i,j}$, all depending on which two of the four pertinent triangles to be chosen on the final BBT: (1) $\Delta p_{i-1}p_i q_j$ and $\Delta p_i p_{i+1} q_j$, (2) $\Delta p_{i-1}p_i q_j$ and $\Delta p_i q_{j+1} q_j$, (3) $\Delta p_i q_j q_{j-1}$ and $\Delta p_i p_{i+1} q_j$, and (4) $\Delta p_i q_j q_{j-1}$ and $\Delta p_i q_{j+1} q_j$. Therefore, if we follow the same manner as in algorithm **GODT** to construct the weighted graph, the weights (which are the amounts of the associated bending energy) on edges $<V_{i,j}, V_{i+1,j}>$ and $<V_{i,j}, V_{i,j+1}>$ are not static – they are path-dependent, i.e., depending on the current path of search that arrives at node $V_{i,j}$.



(a) (b)

Fig. 7. Building the weighted graph for global minimum bending triangulation: (a) path-dependency of the weight (bending energy) on an edge in the graph; (b) the dual nodes weighted graph.

In order to cater to this dynamic nature of weights, and still be able to utilize the Dijkstra's algorithm, we need to develop a new method to build the weighted graph. The graph, denoted as Ω, is called a *dual nodes weighted graph*. When building Ω, every bridge edge is corresponded by two nodes in the graph – one, called *P*-node, indicates that this bridge edge is generated by a *P-succeed* operation, and the other, called *Q*-node, tells that the *Q-succeed* operation was used to generate this bridge edge. Based on this dual node configuration, the non-unique weight problem is elegantly resolved. Specifically, when applying a *P-succeed* (respectively *Q-succeed*) operator on a graph node, regardless *P*- or *Q*-node, the graph edge should point to a *P*-node (respectively *Q*-node); and, for any graph node V, knowing whether it is a *P*- or *Q*-node, the weights on the two outgoing graph edges of V are uniquely determined by Eq.(3). See Fig. 7b for a pictorial illustration of graph Ω: all the vertical graph edges are preluded by *Q-succeed* operations and all the horizontal graph edges by *P-succeed*. To simplify the implementation, only one graph node is constructed for the virtual bridge edge $E_{0,0}$ – so the triangulation is still a single-source problem. Using the Dijkstra algorithm on Ω, the minimum-weight paths from the source node to all the other nodes in the graph can then be determined. For the two dual graph nodes of the ending bridge edge $E_{m,n}$, we choose the one whose path from the source node has the less weight– the path from the source to this node then determines a sequent of *P*- or *Q-succeed* operations that generates a BBT of the strip between *P* and *Q* with guaranteed (globally) minimal total bending energy. Note that the minimum-weight path thus computed between the nodes of $E_{0,0}$ and $E_{m,n}$ designates a BBT of minimum

bending energy for the entire strip; by setting the source and target at other nodes in Ω, one can then triangulate the sub-strip corresponded by the two nodes with the minimum bending energy.

Besides **GMBT**, another type of optimization is minimization of the total area of the triangles in the triangulation, i.e., **Global Minimum Area Triangulation** (**GMAT**). This objective is desirable in certain applications in Computer Graphics or CAD. It is trivial to see that one can use the same logic of **GODT** to realize **GMAT**, except this time the weight on an edge in the graph Γ is irrelevant to the local-convexity, but instead the area of the triangle formed by the two involved bridge edges.

6. COMPLEXITY ANALYSIS

The running time of the local optimal triangulation (including both **LODT** and **LMBT**) is easily seen to be O(m+n), and since the triangulation is constructed in an incremental manner, the required memory is also O(m+n). For the global optimal triangulation algorithms, the computing time is largely dictated by the Dijkstra's algorithm. Since in either Γ or Ω, the number of nodes |V| and the number of edges |E| are both O(mn), as shown in [19], when |E| is much smaller than the square of |V|, which is our case as m and n are large, the computing time of the Dijkstra's algorithm is O((|V|+|E|)log|E|). Therefore, both algorithms **GODT** and **GMBT** require a running time of O((mn)log(mn)); the memory requirement for them is obviously O(mn).

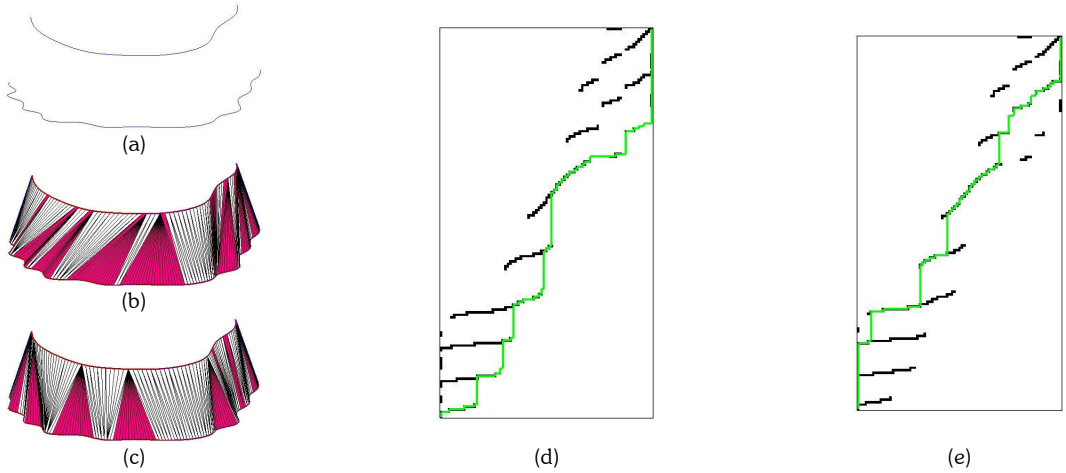


Fig. 8. Example II – Strip triangulation for designing a lace of a dress: (a) the P and Q curves of the lace; (b) local optimal developable BBT (**LODT**+**LMBT**); (c) global optimal developable BBT (**GODT** + **GMBT**); (d) the sequence of P- and Q-succeed operations for (b); and (e) the sequence for (c).

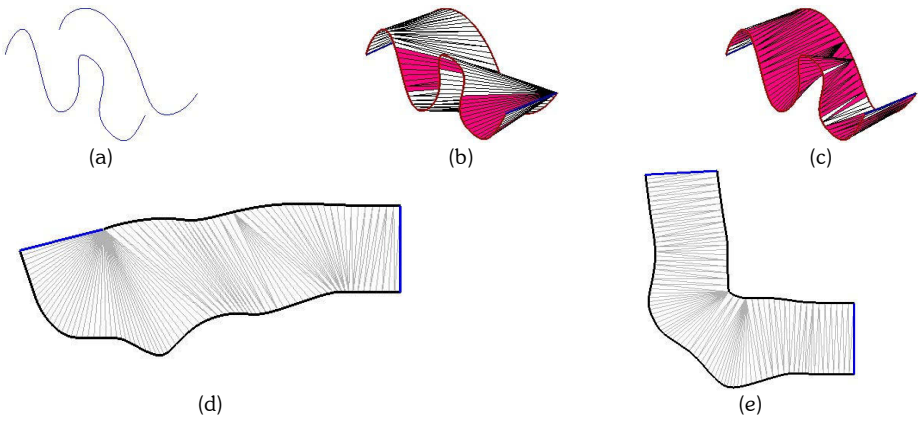


Fig. 9. Example III – **GODT**+**GMBT** vs. **GMAT** only: (a) the given P and Q; (b) the result of **GODT**+**GMBT**; (c) the result of **GMAT**; (d) the flattened 2D pattern of (b); and (e) the flattened 2D pattern of (c).

7. EXPERIMENTAL RESULTS

First of all, we compare the results between the local optimal BBT (**LODT**+**LMBT**) and the global optimal BBT (**GODT**+**GMBT**). In our first example, Example I (Fig. 2b), both the local and global optimal BBTs give the same triangulation. However, in Example II (Fig. 8), the global optimal BBT offers a better result than the local optimal BBT – out of a total of only 225 bridge edges, the triangulation from the global optimal BBT has 149 valid bridge edges, whereas that from the local one is only 113.

In Example III (Fig. 9), the results from **GODT**+**GMBT** and **GMAT** only are compared. The **GODT**+**GMBT** algorithm generates a triangulation with an of area 7.4 unit^2, whereas the area of the triangulation from **GMAT** is only 6.6 unit^2. However, when measured in terms of the developability, the former triangulation contains 95 valid bridge edges while the latter triangulation has only 21. The total number of bridge edges is 126.

In the third comparison, putting developability aside, we match up local minimum bending triangulation with global minimum bending triangulation. For the P and Q given in Fig. 9a, the triangulation results of **LMBT** and **GMBT** are shown in Fig. 10a and Fig. 10b respectively. The total bending energy ratio of (local : global) is (3.3 : 1), with the distribution of bending energy across each bridge edge shown in Fig. 10c.

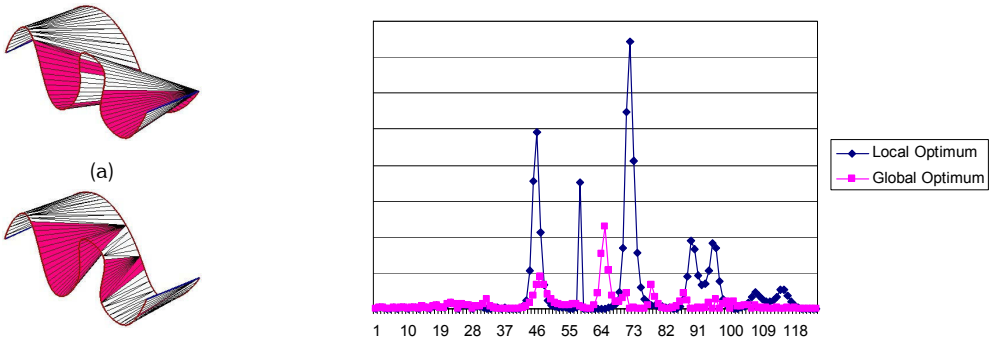


Fig. 10. Example IV – Local vs. global minimum bending triangulation: (a) **LMBT** result; (b) **GMBT** result; and (c) the distribution of bending energy across bridge edges.

Finally, in our last example, Fig. 11 shows the application of the presented triangulation algorithms to the wrinkle design on shoes.

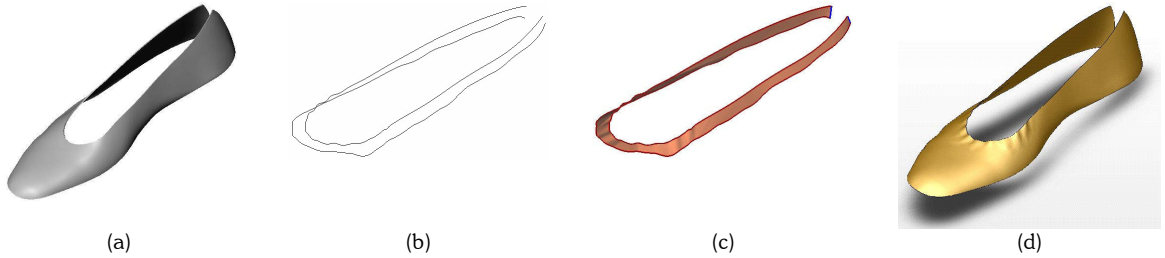


Fig. 11. Example V – Application of developable triangulation: (a) the upper surface of the shoe; (b) profiles P and Q specifying the desired wrinkle pattern; (c) **GODT**+**GMBT** result; and (d) the final upper surface with wrinkles superimposed.

8. CONCLUSION AND DISCUSSION

This paper presents modeling algorithms that help automatically generate a "most" developable boundary triangulation interpolating a given closed 3D boundary curve in the form of a narrow strip. The triangulation algorithm geometrically simulates the folding process of a sheet as it would occur when rolled from one end of the strip to the other, and the rate of the rolling along the strip is controlled for the purpose of achieving maximum developability. By converting this optimal rate controlling problem into a shortest path problem on a directed weighted graph, standard graph search algorithms such as the Dijkstra's can then be used to find the optimal solution. In summary, our approach

has the following advantages: (1) the solution found is globally optimal – it guarantees to find the best triangulation of the given strip (in terms of the local-convexity criterion for the developability); (2) the algorithm is straightforward to implement, robust and efficient – the conversion to the directed weighted graph is straightforward and the Dijkstra's algorithm is well-known to be robust and fast; and (3) with no or only minor modification on the proposed algorithm, global optimization objectives other than developability can be easily realized – such as minimum bending energy, minimum surface area, minimum total edge lengths, etc.

On the algorithmic side, the Dijkstra's is a graph search algorithm applicable to a general directed weighted graph. In our situation, however, the graph Γ is corresponded by a matrix; in addition, all the paths in Γ are monotone – every (directed) edge points to either the north or the east. Invitingly, it should be asked if these additional properties can help improve the O((*mn*)log(*mn*)) time bound. In the aspect of design, a more positive objective will be on how to modify the two given boundary curves *P* and *Q*, but within certain specified tolerance range, so that the resultant BBT from **GODT**+**GMBT** is the optimum among all the possible designs of *P* and *Q* within the tolerance range.

9. REFERENCES

[1] DoCarmo, M., *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, NJ.

[2] Ivars, P., Deciphering the wrinkles of crumpled sheets, *Science News*, Week of May 24, 2003, Vol. 163(21).

[3] Baraff, D. and Witkin, A.P., Large steps in cloth simulation, *Proc. of SIGGRAPH '98*, pp 43-54, 1998.

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

[5] Terzopoulos, D. and Fleisher, K., Modeling inelastic deformation: viscoelasticity, plasticity, fracture, *Proc. SIGGRAPH '88*, pp 269-278, 1988.

[6] Ng, H.N. and Grimsdale, R..L., Computer graphics techniques for modeling clothes, *IEEE Computer Graphics and Applications, Special issue on Computer Graphics in Textiles and Apparel*, pp 28-41, Sept 1996.

[7] Baraff, D., Witkin, A. and Kass M., Untangling cloth, *ACM Transactions on Graphics*, Vol.22, No.3, July 2003, pp 862-870.

[8] Choi, K.-J. and Ko, H.-S., Stable but responsive cloth, *ACM Transactions on Graphics*, Vol.21, No.3, July 2002, pp 604-611.

[9] Hadap, S., Bangerter, E., Volino, P. and Thalmann, N.M., Animating wrinkles on clothes, *Proc. IEEE Visualization '99*, pp 175-182, 1999.

[10] Fan, J., Wang, Q., Yuen, M.-F. and Chan, C. C., A spring-mass model-based approach for wrapping cloth patterns on 3D objects, *The Journal of Visualization and Computer Animation*, v 9, pp 215-227, 1998..

[11] Bando, Y., Kuratate, T., and Nishita, T., A simple method for modeling wrinkles on human skin, *Proc. of Pacific Graphics '02*, pp 166-175, 2002.

[12] Boissieux, L., Kiss, G., Thalmann, N.M. and Kalra P., Simulation of skin aging and wrinkles with cosmetics insight, *Proc. Eurographics Workshop on Computer Animation and Simulation 2000*, pp 15-27, 2000.

[13] Volino, P. and Thalmann, N.M., Fast geometric wrinkles on animated surfaces, *Proc. WSCG '99*, 1999.

[14] Wu, Y., Kalra, P., Moccozet, L. and Thalmann, N.M., Simulating wrinkles and skin aging, *The Visual Computer*, Vol. 15, No. 4, pp 183-198, 1999.

[15] Pottmann, H. and Wallner, J., Approximation algorithms for developable surfaces, *Computer Aided Geometric Design*, vol.16, no.6, July 1999, pp 539-56.

[16] Peternell, M., Recognition and reconstruction of developable surfaces from point, *Proc. Geometric Modeling and Processing 2004*, IEEE Comput. Soc. 2004, pp 301-310.

[17] Frey, W.H., Modeling buckled developable surfaces by triangulation, *Computer-Aided Design*, vol.36, no.4, pp 299-313, 2004.

[18] Tang, K. and Wang, C.C.L., Modeling developable folds on a strip, *Journal of Computing and Information Science in Engineering (ASME Transactions)*, vol.4, 2004.

[19] Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., *Introduction to Algorithms* (2nd ed.), MIT Press, 2001.

[20] Meyers, D., Skinner, S. and Sloan K., Surface from contours, *ACM Transaction on Graphics*, vol. 11, no. 3, 1992, pp. 228-258.

[21] Watts, E.F. and Rule, J.T., *Descriptive Geometry*, Prentice-Hall, New York, 1946.

[22] Pottmann, H., and Wallner, J., *Computational line geometry*, Publisher: Berlin; New York: Springer, 2001, pp. 396-405.