Modeling Developable Folds on a Strip

Kai Tang*
Department of Mechanical Engineering,
The Hong Kong University of Science and Technology,
Clear Water Bay, N.T., Hong Kong
E-mail: mektang@ust.hk

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

Abstract

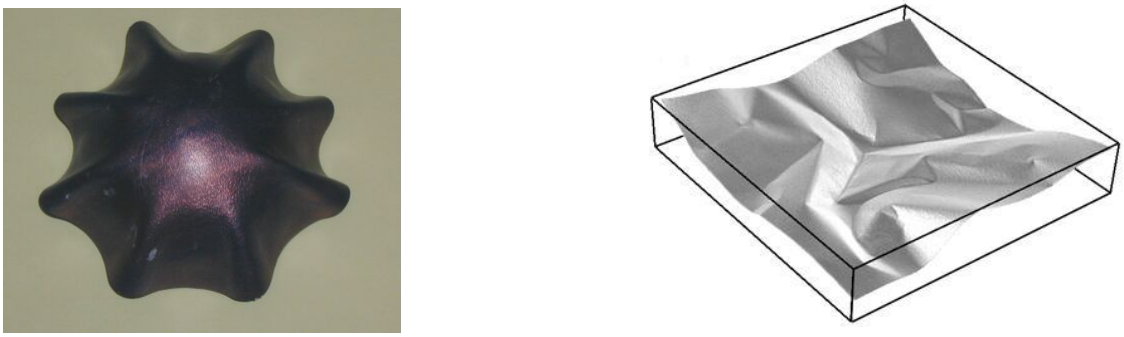
A common operation in clothing and shoe design is to design a folding pattern over a narrow strip and then superimpose it with a smooth surface; the shape of the folding pattern is controlled by the boundary curve of the strip. Previous research results studying folds focused mostly on cloth modeling or in animations, which are driven more by visual realism, but allow large elastic deformations and usually completely ignore or avoid the surface developability issue. In reality, most materials used in garment and shoe industry are inextensible and uncompressible and hence any feasible folded surface must be developable, since it eventually needs to be flattened to its 2D pattern for manufacturing. Borrowing the classical boundary triangulation concept from descriptive geometry, this paper describes a computer-based method that automatically generates a specialized boundary triangulation approximation of a developable surface that interpolates a given strip. The development is achieved by geometrically simulating the folding process of the sheet as it would occur when rolled from one end of the strip to the other. Ample test examples are presented to validate the feasibility of the proposed method.

Keywords: folds, wrinkles, developable surfaces, boundary triangulation, strain energy

* Corresponding author

1. Introduction

Fold (some time also referred as wrinkle) design refers to designing a smooth surface with special folding or wrinkling patterns. Such an operator has some practical interest, especially in the Computer-Aided Design (CAD) of garment and footwear. Unlike wrinkle modeling in computer graphics and animation where the theme is the realism (i.e., the goal is to model wrinkles as realistic as possible), fold design in garment and shoe industries has an extra and also critical requirement to consider: the designed folded surface must be feasible, i.e., it should be able to be manufactured. The manufacturing process of a dress or shoe is sewing together various pieces of 2D patterns. In the majority of cases, the material of the cloth is considered to be inextensible and uncompressible, but easily to be bent or folded. As a result, a 3D shape is feasible only if it can be flattened into a planar pattern without any distortion – the surface distance between any two points does not change in the flattening process. In the terminology of Differential Geometry, this means there exists an *isometric* transformation [1] that maps the surface to its 2D pattern. Or more specifically, a feasible surface must be a *developable* surface – it should be able to be developed back to its initial planar state.



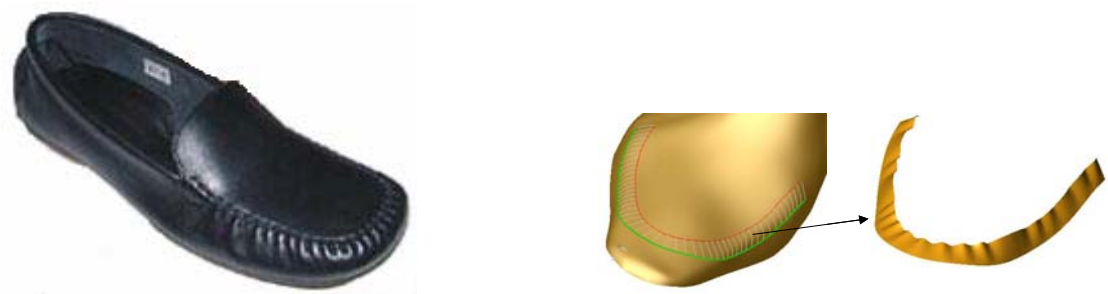
(a) (b)

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

Depending on the material properties of the cloth, two (developable) wrinkled surfaces could look rather drastically different. They can be smooth (such as the one shown in Fig. 1a), or contain creases along curves (see Fig. 1b). Stiff materials like thick leather fold into smooth surfaces, while very pliable materials such as aluminum foil usually form many creases. Regardless its appearance though, a developable surface can only be a ruled surface with zero Gaussian curvature everywhere

(referred to as *developable ruled surface*), or a composite of such ruled surfaces [1]. The most familiar developable ruled surfaces are cylinders and cones.

In this paper, we propose a modeling algorithm for designing developable wrinkled surfaces specified by narrow strips. When defining such a wrinkled surface, a designer first designs two 3D boundary curves and then tries to interpolate them by a developable surface. The distance between the two boundary curves usually is much smaller compared to the lengths of the curves themselves, thus forming a narrow strip. This kind of wrinkled surfaces are frequently encountered in footwear design and clothing design for women and children. An example is given in Fig. 2a where a strip of wrinkles is superimposed on the circumference of the upper layer of the shoe; the process of designing such a wrinkled strip is also schematically shown in Fig. 2b – a strip is "carved" out from the original smooth surface which will then be replaced by its counter-part wrinkled strip. While trying to fit a general 3D closed space curve by a developable surface is an extremely difficult and unsettled problem, the proposed modeling approach will demonstrate that this developable interpolation task becomes attainable if the closed curve is in the form of a narrow strip.



(a) Shoe with a wrinkled strip (b) Carving a strip for wrinkle design

Fig. 2 Developable wrinkle strip design in footwear

2. Related work

Wrinkle design algorithms customarily falls into two categories: those that are used in computer graphics and animation and those for textile (clothing) applications. In the former category, most work relates to how to simulate or model wrinkles on human skin, e.g., facial expression animation. The physically-based simulation methods (cf. [2, 3, 4]) try to use physical laws to describe the behavior of the skin and model wrinkles by establishing and solving a set of

partial differential equations. In contrast, geometric methods usually work on a mesh and, by drawing wrinkle properties observed from physical wrinkles, set up a set of rules for moving the nodes on the mesh to simulate wrinkle generation. Details on these methods and their variants can be found in [5, 6, 7, 8, 9, 10, 11, 12, 13, 14] and the references therein. As for the second category which deals with textile materials, most recent research focuses on how to predict the shape that a given piece of textile, in the form of a 2D shape pattern, will take when draped over a particular solid form at a given position. The resulting shape is a possibly wrinkled surface. Similar to the first category, physically-based methods were proposed for modeling the final shape of the piece of textile (cf. [15, 16, 17]). 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. An earlier survey of such modeling techniques can be found in [18], with some more recent work reported in [19, 20, 21, 22, 23]. There are also some modeling techniques specially suited for computer rendering of wrinkles, for example, refer to [24, 25, 26, 27, 28].

None of the above cited methods can answer our quest: to interpolate two space curves by a developable surface. Actually, the issue of "developability" is completely ignored or avoided in all the existing wrinkle modeling systems. Recently, Frey [29] proposed using boundary triangulations to approximate developable surfaces interpolating a closed space curve. His work is mainly targeted at application in the design of the blankholder for a sheet-metal stamping operation, where the space curve to be interpolated, while some times irregular, is in general fairly plain with small curvature. In this paper, we explore using a variant of boundary triangulations – so called *bridge boundary triangulations* - to interpolate a narrow strip whose boundary curves are usually highly convoluted with large curvature and curvature change. By relying on a geometric method of simulating the sequential folding of the cloth along the strip, which is also inspired by the idea of simulating bending of the sheet during the punch closing process, we demonstrate that for a very large class of strips their approximate developable surface interpolation can be accomplished by bridge boundary triangulations. Ample test examples are provided that exhibit the versatility and feasibility of the proposed modeling algorithm.

3. Basic methodology

Let C_1 and C_2 be two simple C^1 continuous curves in space; they together with the two line segments connecting between their respective two ends form a simple and closed spatial curve, as shown in Fig. 3. The Hausdroff distance between C_1 and C_2 is supposed to be much smaller than the lengths of the two curves themselves; accordingly, the thus formed closed curve will be called a *strip*. Our objective is then to find a developable surface that will interpolate a given strip.

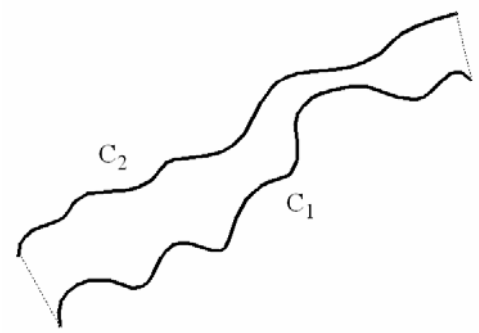


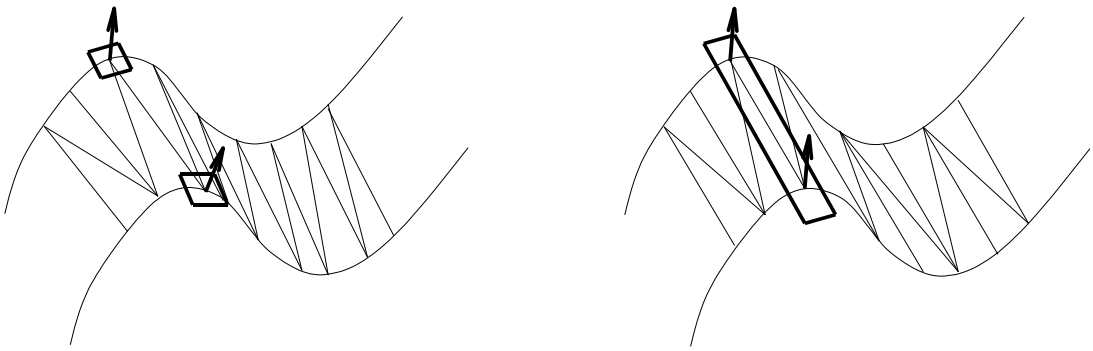
Fig. 3 Two C^1 continuous curves linked by two line segments

3.1. Developable ruled surfaces

A ruled surface is a surface generated by a family of straight lines, and can therefore be parametrically expressed in general by the equation

$$S(u,v) = (1 - v)C_1(u) + vC_2(u) \tag{1}$$

where $C_1(u)$ and $C_2(u)$ are two simple C^1 curves referred to as the *directrices*. The line passing through $C_1(u)$ and $C_2(u)$ for any u in the u-domain of the two curves is called a *ruling* (also called *generator* some time) of the ruled surface. In general, ruled surfaces are not developable. But it is well known 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 and every ruling, then the surface is developable. This common tangent plane condition thus distinguishes developables from the rest of ruled surfaces. On a general ruled surface, the tangent plane changes (twists) from point to point along a ruling, which makes it impossible for the surface to be flattened into the plane without distortion. Not so surprisingly, two pairs of *geometrically identical* directrices can define two rather different ruled surfaces, one developable but the other not, as illustrated in Fig. 4.



(a) non-developable (b) developable

Fig. 4 Developable vs. non-developable ruled surfaces interpolating the same directrices

3.2. Interpolation of a strip by developable ruled surface

Interpolating a general closed 3D curve by a developable surface is an unsettled and extremely difficult problem. An interpolating surface in most cases is an aggregate of several developable ruled surfaces that are G^0 or G^1 connected at some rulings. In such an aggregate, a constituent ruling may very much lie in the interior of the surface, i.e., only one or neither of its two end points fall on the boundary curve. The example given in Fig. 1b illustrates this point for the G^0 case. Examples for the G^1 case are also abundant (just imagine folding a square ply of thick leather into a sphere and then relax it to a certain shape).

However, in our specified setting of interpolating a narrow strip between two C^1 continuous curves C_1 and C_2, based on observation drawn from physical experiments and also due to design intend, it is fairly reasonable to assume that any ruling in an interpolating surface must be of "*bridge*" type – its two end points lie on C_1 and C_2 respectively. More explicitly, let $C_1(u)$ and $C_2(w)$ be the parametric representations of C_1 and C_2 respectively, and, without loss of general, suppose that their parameter ranges are all normalized to [0:1]. A G^1 continuous surface S that interpolates the strip of C_1 and C_2 is uniquely corresponded by a twice differentiable and monotone mapping f(u) from [0:1] to [0:1] such that S is a ruled surface in the form

$$S(u,v) = (1 - v)C_1(u) + vC_2(f(u)), \qquad (u,v)\in[0,1]\times[0,1]. \tag{2}$$

The interpolation task is now simplified to finding a suitable mapping function f(u) given the two curves C_1 and C_2 that will result in a developable ruled surface as specified by the equation

above. If a G^1 continuous $f(u)$ could be found, the result $S(u,v)$ is G^1 (e.g., Example V shown in our paper). Generally, we can at least find an approximation $f(u)$ with G^0 continuity, so a G^0 developable $S(u,v)$ can be found.

3.3. Boundary triangulation and its limit surface

The two input curves C_1 and C_2 are usually fairly convoluted. A typical C_1 or C_2 is a sinusoidal space curve with varying both magnitude and frequency [30]. Attempting to find an exact analytical solution of mapping $f(u)$ for the general case is doomed to fail. Inspired by the classical method in descriptive geometry of boundary triangulation development (cf. [31, 32]) and also the work from [29], this practical numerical approach is also adopted in our modeling algorithm, but with a variation. The boundary triangulation development approximates a surface by a collection of triangles whose vertices lie only on the boundary of the surface. More specifically to our case of a strip, let $P = \{p_1, p_2, \ldots, p_m\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$ be two polygonal chains that are approximations of C_1 and C_2 respectively; a *bridge boundary triangulation* (BBT) of the strip of C_1 and C_2, in terms of P and Q, is a collection of *ordered N* triangles in space $\sum = \{T_1, T_2, \ldots, T_N\}$ that meet the following criteria:

(1) The line segment between two adjacent nodes on the same boundary curve is defined as a *bank edge*, i.e., $<p_i, p_{i+1}>$ or $<q_j, q_{j+1}>$;

(2) The line segment linking the nodes from different boundary curves is defined as a *bridge edge*, e.g., $<p_j, q_h>$;

(3) Two bridge edges are *adjacent* if they have one node shared;

(4) The triangles of $\sum$ are all formed by two adjacent bridge edges and one bank edge, and two neighboring triangles (e.g., T_k and T_{k+1}) should have one bridge edge shared.

We can interpret a bridge boundary triangulation $\sum$ as a discrete approximation of mapping $f(u)$ – the partial orderings exerted on P and Q by the ordered triangles in $\sum$ define a discrete mapping for $f(u)$. This discrete approximation approaches to a continuum $f(u)$ when the sampling points *m* and *n* on P and Q tend to infinity, which then defines a ruled surface. Therefore, provided that the numbers of sample points *m* and *n* are reasonably large enough, every bridge boundary

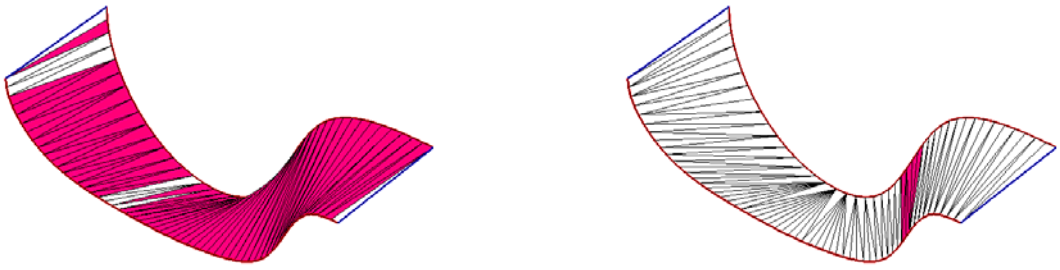
triangulation can be viewed as a facet representation of a ruled surface, called its *limit ruled surface*. The examples given in Fig. 4 show two bridge boundary triangulations of a same strip, whose limit ruled surfaces are quite different. Since we are only interested in developable surfaces, the desired property for a bridge boundary triangulation is that its limit surface is developable. To characterize such a developable BBT, we notice that, when *m* and *n* are large enough, every bridge edge on the triangles in a BBT is either a ruling of the limit ruled surface or a line segment "sandwiched" between two very close rulings. Based on the common tangent plane condition for developable ruled surfaces, Frey [29] proposed the local convexity theorem: in a developable boundary triangulation, every interior edge (those not on the boundary curve) must be locally convex: the edge must be on the convex hull of a narrow region near the edge on the triangulation. To be more specific pertaining to our case, a bridge edge $<p_i, q_j>$ is locally convex if it lies on the convex hull of the six points $\{p_{i-1}, p_i, p_{i+1}, q_{j-1}, q_j, q_{j+1}\}$. Conversely, if every bridge edge in a bridge boundary triangulation $\sum$ satisfies the local convexity condition, given that the numbers *m* and *n* are sufficiently large, its limit ruled surface, if it exists, ought to be developable. We summarize this assertion by the following proposition.

Proposition: a bridge boundary triangulation $\sum$ has a developable limit surface if and only if every bridge edge in $\sum$ is locally convex.

3.4. Developability vs. flattenability

It is imperative to distinguish between the flattenability and developability of a BBT. Flattenability measures whether a BBT can be flattened into its 2D pattern without area distortion. Since triangles are planar, it is trivial to see that any 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. For instance, suppose C_1 is a sinusoidal curve in the plane $x = 0$ while C_2 is a straight line segment in the plane $x = 1$, then it is obvious that no mapping function $f(u)$ exists that defines a developable ruled surface as given in Eq. (2) to interpolate the strip of C_1 and C_2. In such a case, although any BBT of the strip is flattenable, none of their limit surfaces are developable. Topologically, in terms of local convexity, a non-developable BBT can always be uniquely divided into consecutive subsets such that within each subset either all the edges are locally-convex or none of them is locally convex. As an example, Fig. 5a shows a BBT of a strip where those edges that are not

locally convex are displayed in red color. Physically, when the material is stiff (which is the most of case in garment and footwear industries) and thus the final surface is assumed to be G^1 continuous, if one tries to fold the flattened 2D pattern of this BBT back onto the strip, material failure always happens at those red edges, i.e., they will incur stretching or compression. This translates into a natural desire of finding a BBT that minimizes the "red" edges. For instance, the BBT shown in Fig. 5b interpolates the same strip as that of Fig. 5b; apparently it is much better than that of Fig. 5a as it has a much smaller number of "red" edges compared to the one in Fig. 5a.



(a) (b)

Fig. 5 Two BBTs with different local-convexity of a same strip

4. Simulating folding

As already alluded in the preceding sub-section, for a given strip a desirable BBT is one whose "red" edges should be as few as possible. If P and Q, with m and n vertices respectively, are coplanar, then any bridge boundary triangulation of them is also a regular triangulation of the region bounded by the strip of P and Q. Consequently the number of triangles N in any bridge boundary triangulation is always $m+n-2$. In the Appendix, it is shown that, in the worst case, when every bridge edge is locally convex, there can be as many as $\binom{m-1}{m+n-2}=\binom{n-1}{m+n-2}$ distinct bridge boundary triangulations for any pair of P and Q with m and n vertices respectively. Facing this factorial number of candidates, attempting to find a good BBT by exhaustively searching through all the possible candidates is neither plausible nor practical (given large m and n). Local and heuristic alternative approaches have to be explored. Our solution is similar in principle to the simulation idea of Frey in [29]. However, the physical event simulated in [29] is the punch closing process; in our case, we simulate folding a sheet along a narrow strip. The difference resides in that while the order of the rulings generated in a punch closing simulation is random, it is strictly

sequential in a folding simulation, as we elaborate next.

Imagine folding a stiff sheet over the strip of P and Q. If the final shape of the folded sheet is developable, i.e. without any distortion, the folding process can be viewed as bending the sheet sequentially about the bridge edges from one end of the strip to the other. Suppose as a priori the line segment between p_i and q_j is known to be a ruling in the current bridge boundary triangulation being developed, and we want to find the next ruling about which the sheet is to be bent. The local convexity and the bending energy presented in this section are the criteria utilized to choose the next ruling. Since the folding is a continuous process, the next folding ruling should be very close to the current one. We define two particular edges in relation to edge $< p_i, q_j >$ as follows:

Q-succeeding edge: edge $<p_{i'}, q_{j'}>$ is the Q-succeeding edge of $<p_i, q_j>$ if $j'>j$, $i' \geq i$, and if there is any other *different* edge $<p_{i''}, q_{j''}>$ with $j''>j$ and $i'' \geq i$, then it must be either $j''>j'$ or $i''>i'$.

P-succeeding edge: edge $<p_{i'}, q_{j'}>$ is the P-succeeding edge of $<p_i, q_j>$ if $i'>i$, $j' \geq j$, and if there is any other *different* edge $<p_{i''}, q_{j''}>$ with $i''>i$ and $j'' \geq j$, then it must be either $j''>j'$ or $i''>i'$.

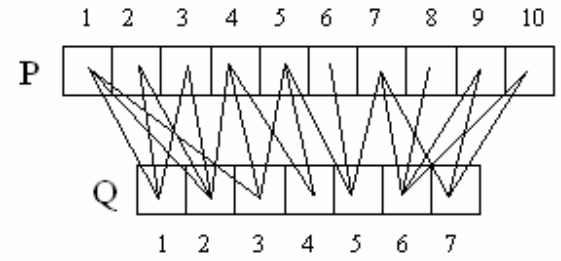


Fig. 6 Valid edges

Figure 6 gives a schematic example of all the valid (locally convex) edges between a pair of P and Q, each with 10 and 7 vertices respectively. For edge $<p_5, q_4>$, its Q-succeeding edge is $<p_5, q_5>$, and the P-succeeding edge is $<p_6, q_5>$. Typically, which is also verified in our numerous experiments, the Q-succeeding and P-succeeding edges of $<p_i, q_j>$ are $<p_i, q_{j+1}>$ and $<p_{i+1}, q_j>$ respectively. Of these two edges, we need to determine one to be the next ruling edge. The criterion we use for selection is based on the reasoning that folding always tends to the direction that requires minimum work, or in other words, the change of the geometry of the sheet after bent about ruling edge $<p_i, q_j>$ should result in a minimum change of the strain energy of the sheet. Consider

the case of P-succeeding edge of $<p_i, q_j>$ and assume it is $<p_i, q_{j+1}>$. $<p_i, q_j>$ is on some triangle T_k in the partial $\sum$ so far constructed. For discussion purpose, suppose T_k lies in the *xy* plane and $<p_i, q_j>$ on the *y*-axis. Figure 7 depicts the view in the $<p_i, q_j>$ direction (the *y*-axis) of bending the triangle $\Delta p_i q_j q_{j+1}$ about $<p_i, q_j>$. Assuming the bending angle is very small, the effective strain energy due to this bending is (cf. [29])

$$U = \int_0^L \frac{EI(s)}{2R^2} ds, \tag{3}$$

where E is the Young's modulus of elasticity, R is the effective bending radius as given in Fig. 7, L is the length of the projection of edge $<q_j, q_{j+1}>$ in the *xz* plane, and $I(s)$ is the moment of inertia of the cross-section of triangle $\Delta p_i q_j q_{j+1}$ at a distance s from edge $<p_i, q_j>$. For a rectangular cross-section with width $w(s)$ and thickness T, $I(s)$ is given

$$I(s) = \frac{T^3}{12} w(s). \tag{4}$$

For triangle $\Delta p_i q_j q_{j+1}$, $w(s) = W \frac{L-s}{L}$, with W being the length of $<p_i, q_j>$. Plug $w(s)$ into Eq. (4) and then into Eq. (3) and do the integral, the strain energy U becomes

$$U = \frac{1}{24} E T^3 A \frac{1}{R^2}, \tag{5}$$

where A is the area of the triangle $\Delta p_i q_j q_{j+1}$. Since $R = \frac{L}{2 \sin\theta}$, the final equation for describing the increase of the strain energy due to this bending becomes

$$U = K \frac{A \sin^2 \theta}{L^2}, \tag{6}$$

where K is a constant.

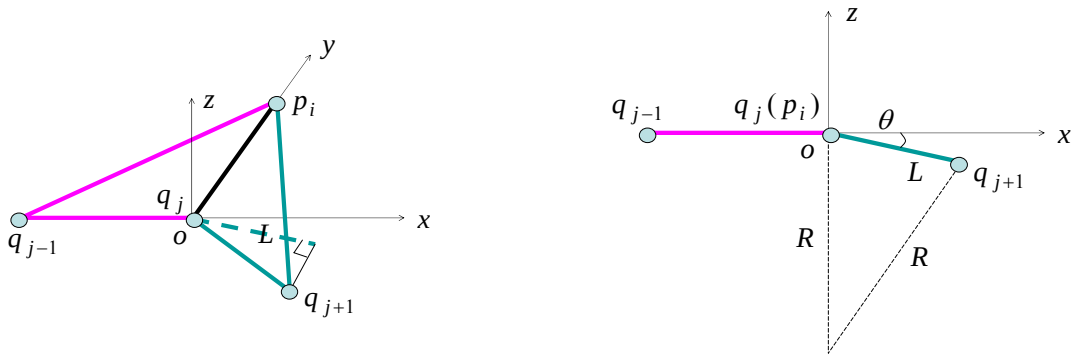


Fig. 7 Bending energy calculation

For a point on P curve, if there are *m* P-succeeding edges passing it, the bending energy at this point is

$$U_p = \sum_m U_i \tag{7}$$

where U_i is the bending energy on the *i*th P-succeeding edge that can be determined by Eq. (6). We also define the total bending energy on the P curve as the summary of bending energy at all points,

$$U^P = \sum_p U_p . \tag{8}$$

In a similar way the strain energy increase caused by bending the Q-succeeding edge can also be defined. The total energy on the Q curve is represented by U^Q. It is not hard to see that $U^P = U^Q$.

5. Modeling Scheme

In this section, the modeling scheme of boundary bridge triangulation is discussed. Figure 8 shows the flowchart of the system – the words inside each box (module) describe the functionality of that module and the ovals represent user input. They are described in detail next.

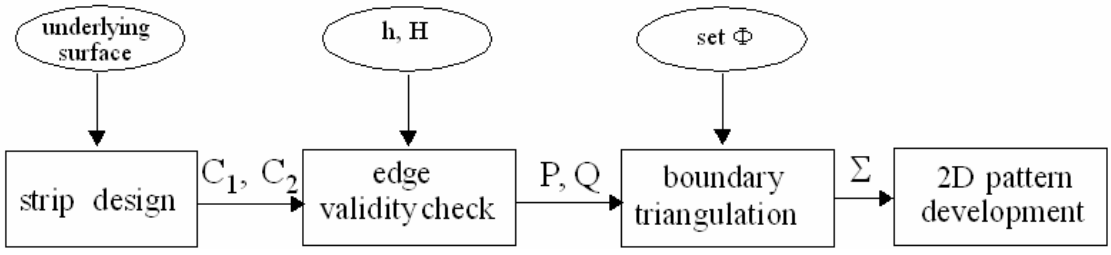


Fig. 8 Modeling system

5.1. Strip design module

Two ways are provided for defining the two boundary curves C_1 and C_2: decoupled and coupled. In the decoupled mode, the user simply uses the graphics interface to specify two spatial parametric curves. In the coupled mode, the user first provides a smooth surface to be used as the underlying surface (e.g., the upper layer of a shoe). He then specifies two curves X_1 and X_2 on this underlying surface; in addition, a magnitude threshold A and a frequency threshold ω are indicated for each of X_1 and X_2. The system then will automatically generate a sinusoidal curve C_1 (C_2) based on X_1 (X_2), it's A and ω, and the curvature information of X_1 (X_2) on the underlying surface. For details on how to design practical boundary curves for wrinkles on a surface, refer to [30].

5.2. Edge validity checking module

Once C_1 and C_2 are defined, they are approximated by two polygons $P = \{p_1, p_2, \ldots, p_m\}$ and $Q = \{q_1, q_2, \ldots, q_n\}$. The sampling points p_i and q_j are either uniformly distributed on the curves with the user specified *m* and *n*, or sampled adaptively based on an approximation error tolerance δ given by the user. The edge validity checking module then checks every edge $<p_i, q_j>$ ($1 \le i \le m$ and $1 \le j \le n$) against two criteria: local convexity and edge length limit. When checking the local convexity of $<p_i, q_j>$, in order to avoid numerical errors, points $\{p_{i-h}, p_{i-h+1}, \ldots, p_i, p_{i+1}, p_{i+2}, \ldots, p_{i+h}, q_{j-h}, q_{j-h+1}, \ldots, q_j, q_{j+1}, q_{j+2}, \ldots, q_{j+h}\}$ are used to construct the local convex hull, where $h \ge 1$ (called *level of local convexity*) is user specified. In all our test examples, h is set to 2. The edge length limit checking filters out those edges whose lengths are longer than αH, where H is the Hausdorff distance between C_1 and C_2, and α is a user specified factor (e.g., in all our testing examples, we choose α=1.5). This is motivated by the fact that in practice a valid ruling usually is shorter than certain length in relation to C_1 and C_2; for instance, no ruled surface of C_1 and C_2 would have edge $<p_1, q_n>$ as a ruling whose length is usually much longer than the rest of possible rulings.

5.3. Boundary triangulation module

This is the main module of the system. After the edge validity checking, the output is an *m* by *n* binary array M (called *edge validity matrix*): the entry M[*i*,*j*] is "1" if edge $<p_i, q_j>$ has passed the edge validity check, and "0" otherwise. By default, M[1,1] and M[*m*,*n*] are always "1". Besides array M, the user input to this module is a set of *a priori* rulings $\Phi = \{(i_1, j_1), (i_2, j_2), \ldots, (i_k, j_k)\}$, where each entry $(i,j) \in \Phi$ tells the system that edge $<p_i, q_j>$ should be taken as a ruling in the boundary triangulation. The entries in Φ are lexicographically ordered; that is, if entry $(i_{k'}, j_{k'})$ is in

front of entry ($i_{k''}$, $j_{k''}$), then $i_{k'} \leq i_{k''}$ and $j_{k'} \leq j_{k''}$. By default, the minimum for Φ is $\{(1,1), (m, n)\}$. The bridge boundary triangulation operation is *independently* performed on P and Q between each pair of (i_l, j_l) and (i_{l+1}, j_{l+1}), for l=1,2,…, k-1. For lucidity of the discussion, let's assume that the given Φ is the simplest, i.e., $\Phi = \{(1,1), (m, n)\}$.

Starting from edge $<p_1, q_1>$, we construct the bridge boundary triangulation by simulating the folding process. Suppose at present edge $<p_i, q_j>$ is determined to be the ruling and we need to decide the next ruling edge. By looking at the entries in array M in rows and columns after the ith and jth, we can quickly identify the P-succeeding and Q-succeeding edges of $<p_i, q_j>$. The algorithm is outlined as follows.

Algorithm Min_Folding_BBT ()

1. While ($i<m$ *OR* $j<n$) {
2. *if* (p_i is a point on a *priori* ruling but q_j is *NOT*)
3. *Apply* Q-succeeding, so $j = j+1$;
4. *else if* (q_i is a point on a *priori* ruling but p_j is *NOT*)
5. *Apply* P-succeeding, so $i = i+1$;
6. *else* {
7. *if* ((M[i,j+1]=="1") *AND* (M[i+1,j]=="0"))
8. *Apply* Q-succeeding, so $j = j+1$;
9. *else if* ((M[i+1,j]=="1") *AND* (M[i,j+1]=="0"))
10. *Apply* P-succeeding, so $i = i+1$;
11. *else* {
12. *if* ((M[i,j+1]=="0") *AND* (M[i+1,j]=="0")) {
13. Find the closest element M(i',j')="1" in M with i'>i and j'>j;
14. Add (i',j') into Φ as a prior ruling;
15. }
16. *if* (Bending energy of P-succeeding is less than that of Q-succeeding)
17. *Apply* P-succeeding, so $i = i+1$;
18. *else*

19. *Apply* Q-succeeding, so $j = j+1$;

20. }

21. }

Actually, the limit surface of the triangles constructed with a set of consecutive invalid edges is a conical surface. In essence, in ***Algorithm*** Min_Folding_BBT(), our strategy of triangulation is to link all valid (locally convex) edges as many as possible so to minimize the total area of the conical surfaces.

In computer graphics applications, where developability is usually overridden by realism, one particular BBT scheme is popularly adopted to construct a mesh surface linking two 3D curves – the *shortest edge triangulation* ([33, 34, 35]), where the rule to choose P-succeeding or Q-succeeding is based on the length of edge to be created – the shorter one is chosen. For comparison purpose, we have also implemented this triangulation – referred to as Min_Dist_BBT – and applied it to our test examples. As the local-convexity is completely ignored by Min-Dist_BBT, it should be expected that for a same strip the Min_Dist_BBT would have more "red edges" than the Min_Folding_BBT, as well as more bending energy U^P (U^Q). This assertion is verified by our experiments, as to be seen in Section 6.

5.4. 2D pattern development

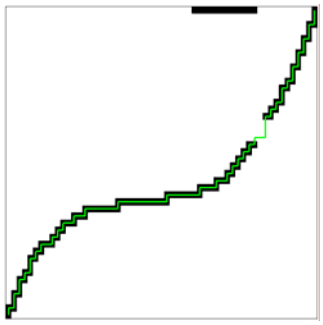
The output from the boundary triangulation module is a set of triangles $\Sigma = \{T_1, T_2, \ldots, T_N\}$ that constitute a bridge boundary triangulation. Since T_i's are ordered, they can be *uniquely* developed (flattened) into their corresponding 2D patterns. Specifically, starting at T_1, we treat the rest of triangles $\{T_2, T_3, \ldots, T_N\}$ as a rigid body and rotate them about the edge between T_2 and T_1 so that, after the rotation, T_2 becomes coplanar with T_1. We then move to T_2 and rotate $\{T_3, T_4, \ldots, T_N\}$ altogether about the edge between T_3 and T_2 to bring T_3 to befall coplanar with T_2 (and T_1), and so on, until the last triangle T_N is reached and rotated. The resulting 2D pattern is a simple polygon that satisfies (1) its area equals the sum of the areas of the N triangles T_i, and (2) its edges are only those corresponding to P and Q and the two special bridge edges $<p_1, q_1>$ and $<p_m, q_n>$.

6. Experiments

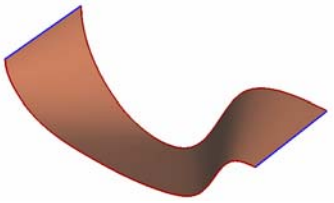
Ample experimental examples are presented in this section to validate the feasibility of our method. Figure 9 shows the comparison results between the proposed Min_Folding_BBT method and the existing Min_Dist_BBT method of the first example, Example I, which was already briefly visited in Fig. 5. The input curves C_1 and C_2 are depicted in Fig. 9a. In Fig. 9b we give the edge validity matrix M of the BBT ("1" is presented by a black point and "0" by a white point), where the green curve corresponds to the order of triangles generated by ***Algorithm*** Min_Folding_BBT(). The red colored triangles indicate those where local-convexity is violated. The bending energy (computed by Eq. (7)) at the sample points on the two curves C_1 and C_2 is exhibited in Fig. 9i. Comparing Fig. 9e with 9f and the bending energy in Fig. 9i, it is apparent that, compared with the proposed Min_Folding_BBT approach, Min_Dist_BBT incurs much more bending energy as well as more violation of local-convexity. In the same display format, the comparison results of Example II and III are shown in Fig. 10 and 11 respectively. It is worth noting that, by examining the derivatives of the two input curves in Fig. 10 and 11, one can see that the strips in Example II and III are impossible to admit a developable ruled surface; in other words, when folded back to the strip, any 2D pattern will always incur either some surface G^1 discontinuity (creases) or area distortion (stretching or compression). Besides, by Example II (Fig. 10), it is easy to find that the resultant surface of Min_Dist_BBT looks better, and thus would be preferable in graphics applications.

Figure 12 shows the experimental result of the fourth example, which demonstrates the effect of a priori edges. Two a priori rulings are added to Φ, i.e., $\Phi = \{(1,1),(\frac{m}{3},\frac{n}{3}),(\frac{2m}{3},\frac{2n}{3}),(m,n)\}$, which are shown as blue bolded edges in Fig. 12. Since our folding simulation is inherently a *local* optimization, it may miss the global optimum. Adding additional a priori rulings compensates this locality to certain extent. For example, the resultant BBTs with and without the a priori rulings in Fig. 12 have a total bending energy of $U^P=U^Q=23.77$ and $U^P=U^Q=32.08$ respectively, a 26% reduction.

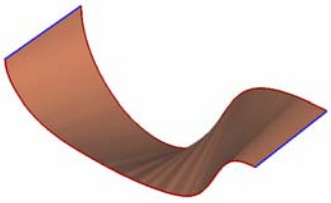
(a) the input curves



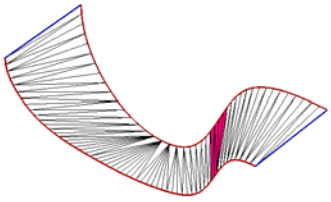
(b) validity matrix of BBT



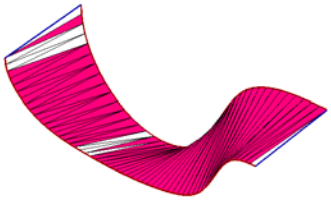
(c) shaded result of Min_Folding_BBT



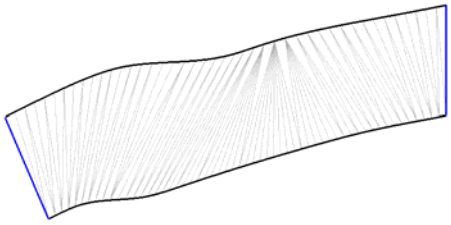
(d) mesh result of Min_Dist_BBT



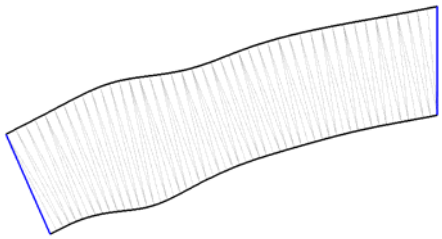
(e) mesh result of Min_Folding_BBT



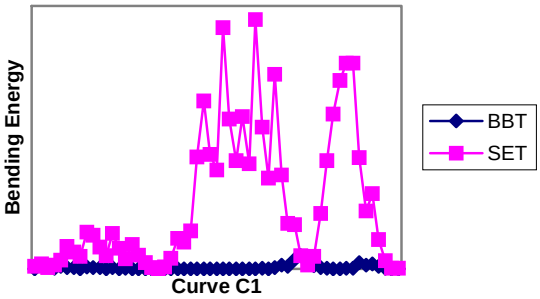
(f) mesh result of Min_Dist_BBT

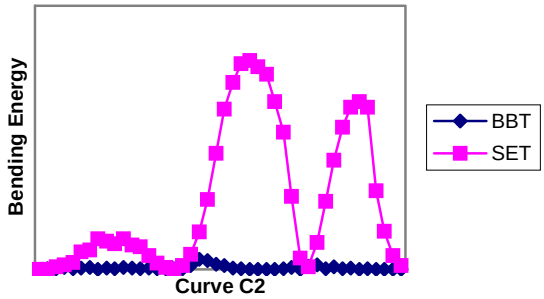


(g) flattened 2D shape of Min_Folding_BBT result



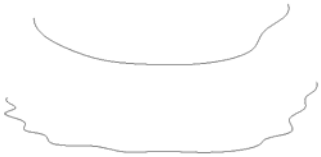
(h) flattened 2D shape of Min_Dist_BBT result



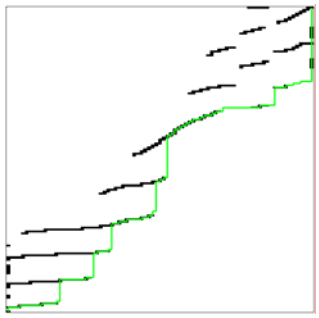


(i) bending energy distribution comparison, where BBT is the curve of Min_Folding_BBT and SET represents the curve of Min_Dist_BBT

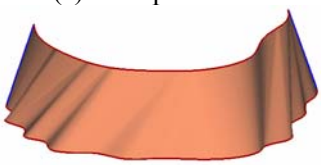
Fig. 9 Example I



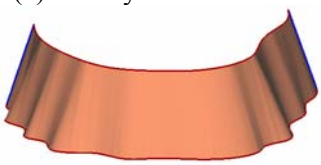
(a) the input curves



(b) validity matrix of BBT



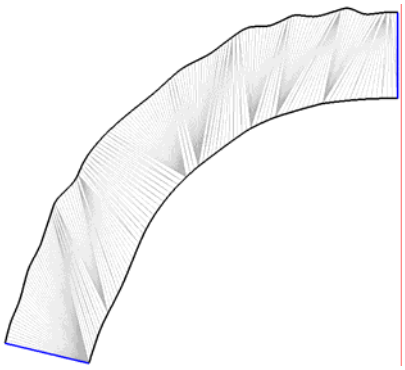
(c) shaded result of Min_Folding_BBT



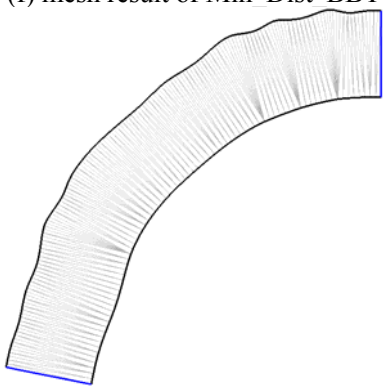
(d) shaded result of Min_Dist_BBT

(e) mesh result of Min_Folding_BBT

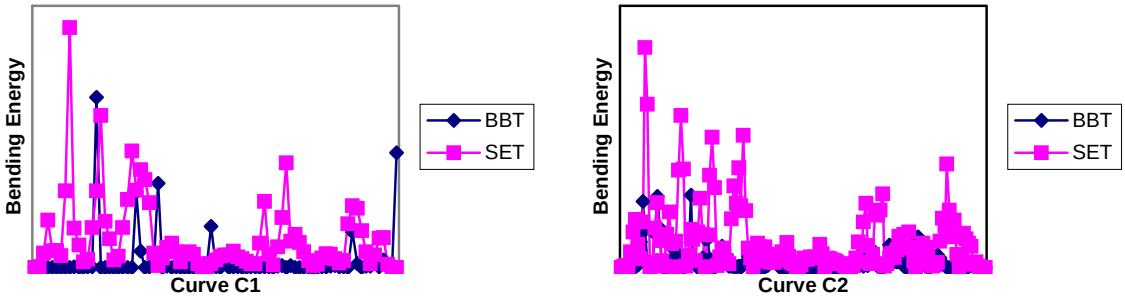
(f) mesh result of Min_Dist_BBT



(g) flattened 2D shape of Min_Folding_BBT



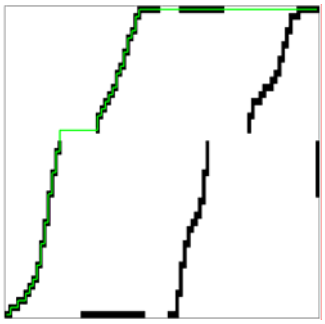
(h) flattened 2D shape of Min_Dist_BBT



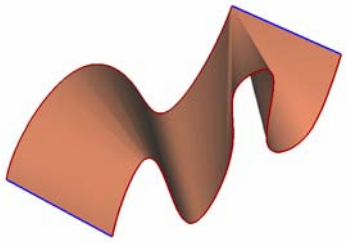
(i) bending energy distribution comparison

Fig. 10 Example II

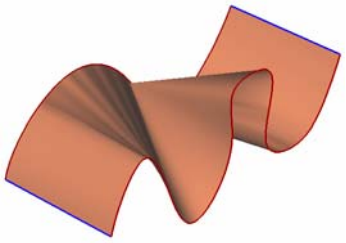
(a) the input curves



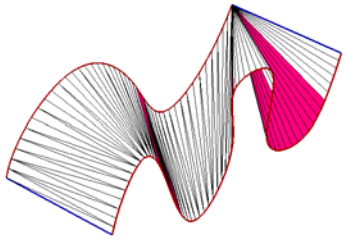
(b) validity matrix of BBT



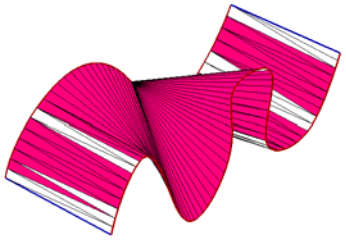
(c) shaded result of Min_Folding_BBT



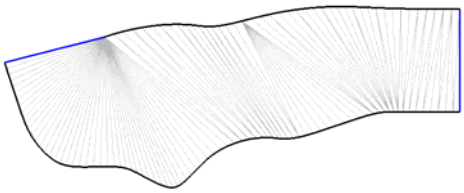
(d) shaded result of Min_Dist_BBT



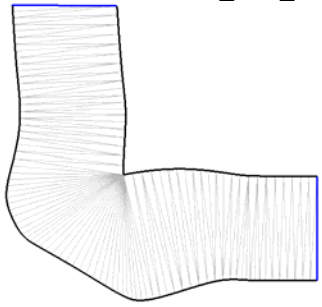
(e) mesh result of Min_Folding_BBT



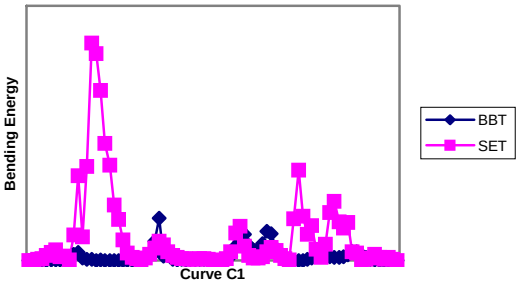
(f) mesh result of Min_Dist_BBT

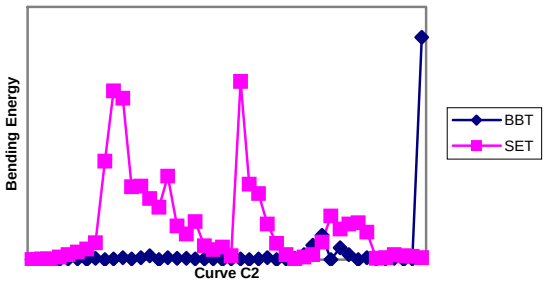


(g) flattened 2D shape of Min_Folding_BBT



(h) flattened 2D shape of Min_Dist_BBT

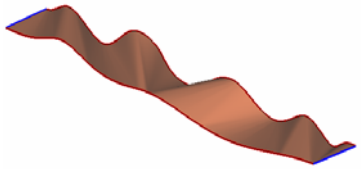




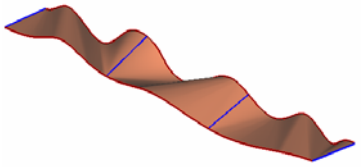
(i) bending energy distribution comparison

Fig. 11 Example III

(a) the input curves



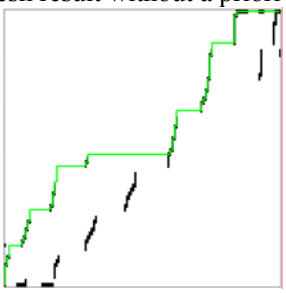
(b) shaded result without a priori edges



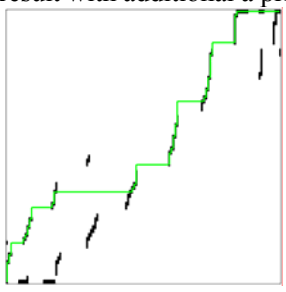
(c) shaded result with additional a priori edges

(d) mesh result without a priori edges

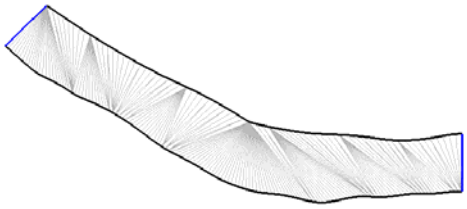
(e) mesh result with additional a priori edges



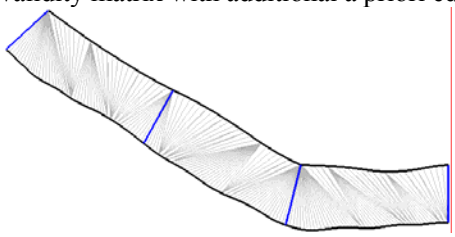
(f) validity matrix without a priori edges



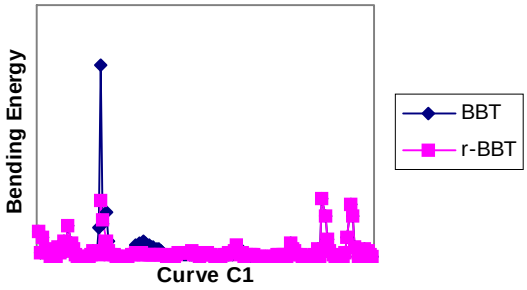
(g) validity matrix with additional a priori edges

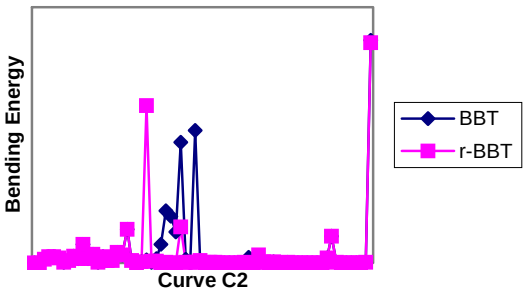


(h) flattened 2D shape without a priori edges



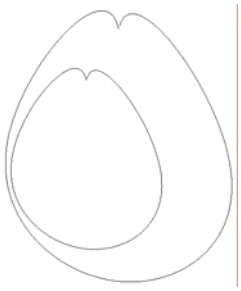
(i) flattened 2D shape with additional a priori edges



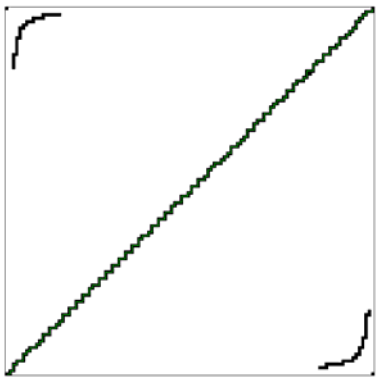


(j) bending energy distribution comparison, where r-BBT represents the curve of BBT with priori rulings

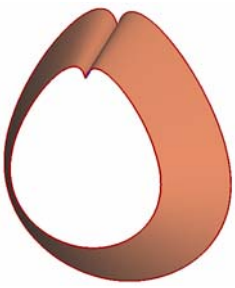
Fig. 12 Example IV – with vs. without a prior rulings



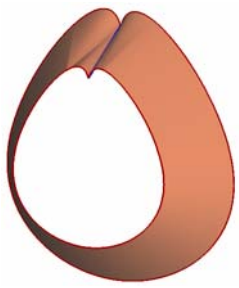
(a) the input curves



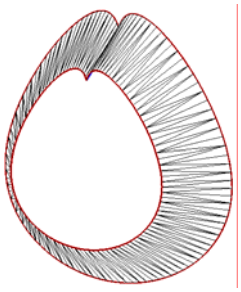
(b) validity matrix of BBT



(c) shaded result of Min_Folding_BBT

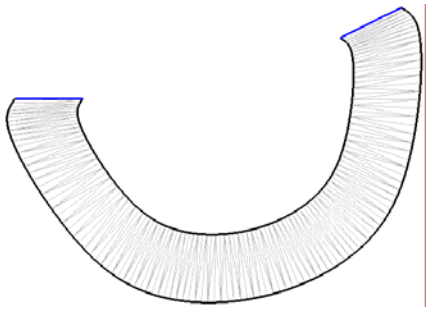


(d) shaded result of Min_Dist_BBT

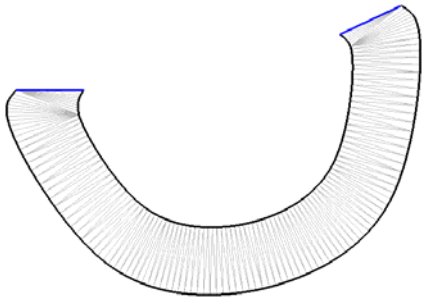


(e) mesh result of Min_Folding_BBT

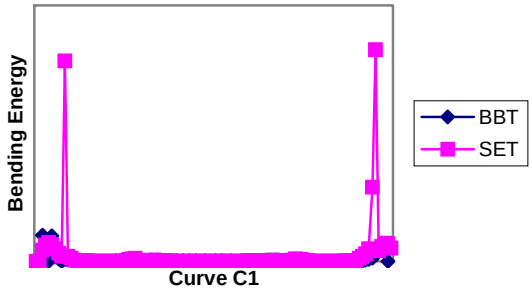
(f) mesh result of Min_Dist_BBT

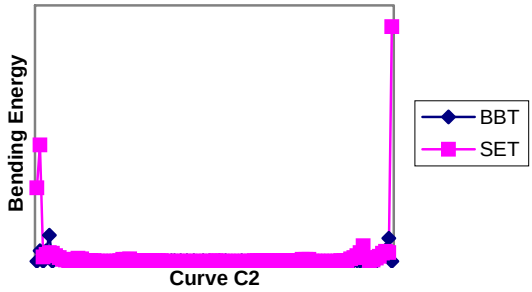


(g) flattened 2D shape of Min_Folding_BBT



(h) flattened 2D shape of Min_Dist_BBT



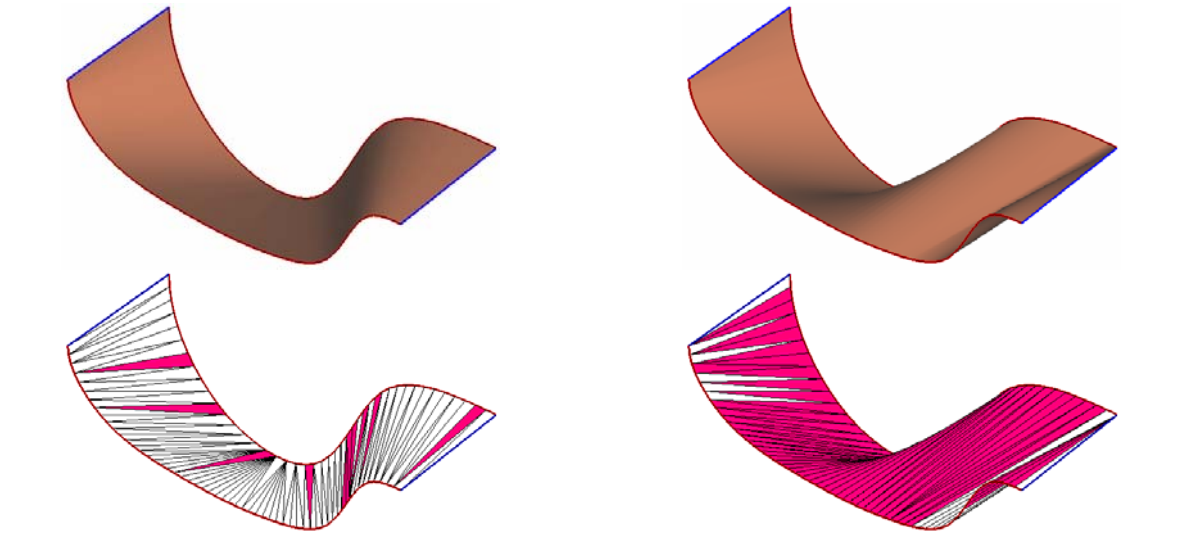


(i) bending energy distribution comparison

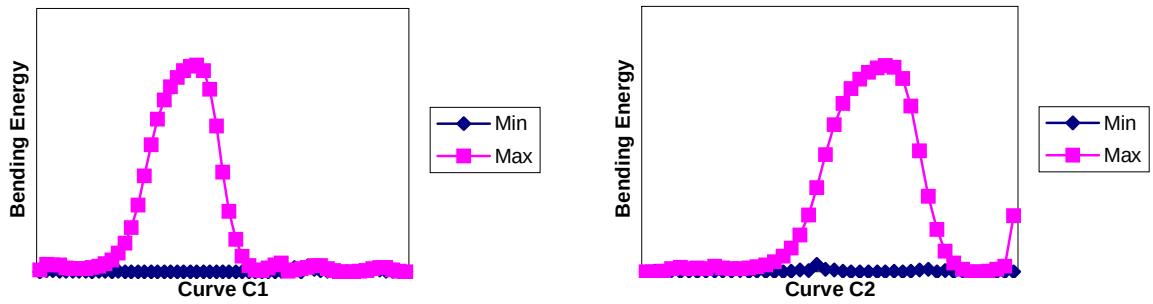
Fig. 13 Example V

In the fifth example, given in Fig. 13, the two input curves admit a developable ruled surface. This example testifies to the *response condition* of the proposed modeling system – how close it is able to capture the original developable surface if it exists. As shown in Fig. 13, in this particular example, the original developable surface is successfully captured by the proposed Min_Folding_BBT algorithm, whereas the intuitive Min_Dist_BBT scheme failed miserably.

Example VI (Fig. 14) is used to further validate the rationale of the minimum bending energy criterion. The opposite of the minimum bending energy is the maximum bending energy criterion – the next ruling edge is selected to be the one that causes larger bending energy increase. If our minimum bending energy assertion is valid, then the folding result based on the maximum bending energy criterion would incur more violation of local-convexity. This is clearly illustrated by Example VI.



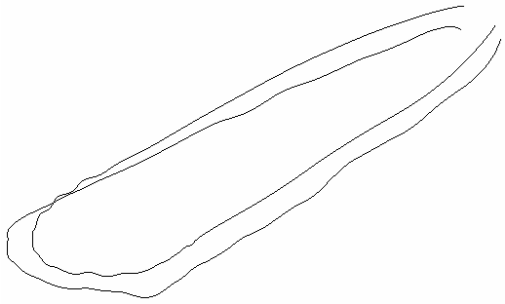
(a) the result from the minimum bending criterion (b) the result from the maximum bending criterion



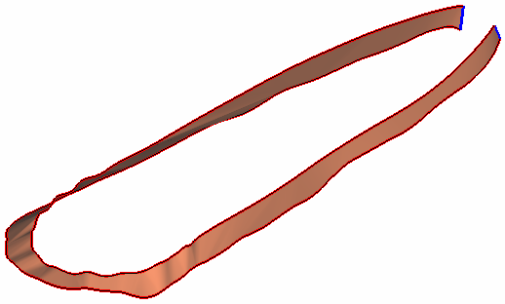
(c) bending energy distribution comparison

Fig. 14 Example VI: Comparison between the minimum vs. the maximum bending criterion

The last example, Example VII (Fig. 15), is an application in shoe wrinkle design. Very often wrinkles are deliberately designed on the shoe upper surface to suit fashion or other purposes. In this test example, a strip (Fig. 15a) is carved out from the shoe upper surface based on the surface curvature and geodesic offset information (cf. [30]) and a folding surface (Fig. 15b) is generated using Min_Folding_BBT over this strip. This folding surface is then superimposed on the original shoe upper surface to form the final wrinkled surface (Fig. 15d).

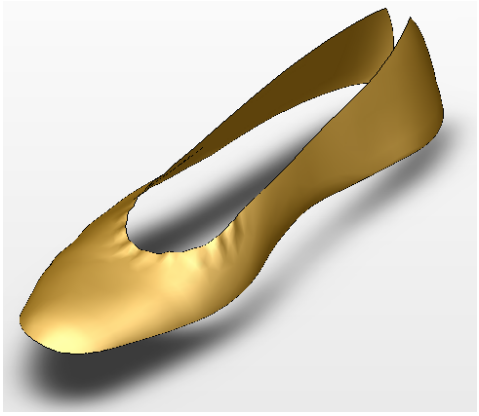


(a) the input curves



(b) the shaded result of Min_Folding_BBT

(c) the original shoe upper surface



(d) wrinkled upper combined by (b) and (c)

Fig. 15 Example VII: Shoe surface wrinkle design based on Min_Folding_BBT

Finally, for comparison purpose, Table 1 lists the total bending energy data for Examples I through V. Table 2 lists the running time comparison of Min_Folding_BBT and Min_Dist_BBT. From Table 2, we can find that the complexities of the two algorithms are almost the same – this is because that both algorithms are linear to the number of nodes on boundary curves.

Table 1. Total bending energy comparison

Example	Energy generated by Min_Folding_BBT		Energy generated by Min_Dist_BBT	
	U^P	U^Q	U^P	U^Q
I	2.46	2.46	113.63	113.63
II	45.51	45.51	182.81	182.81
III	24.75	24.75	122.07	122.07
IV	32.08	32.08	217.93	217.93
IV with two priori rulings	23.77	23.77		
V	8.24	8.24	34.19	34.19

Table 2. Runtime comparison of Min_Folding_BBT and Min_Dist_BBT

Example	Node number on curve C_1	Node number on curve C_2	Time by Min_Folding_BBT	Time by Min_Dist_BBT
I	57	44	0.02s	0.01s
II	83	153	0.04s	0.05s
III	83	44	0.021s	0.03s
IV	134	70	0.03s	0.03s
IV with 2 priori rulings			0.03s	
V	112	112	0.03s	0.04s

*All the tests are performed on a PC with AMD 2400+ CPU by a program written in Visual C++.

7. Conclusions

This paper presents a novel method of approximating developed surfaces on given boundary strips for modeling developable wrinkle surfaces. This modeling task is often required in clothing and shoe design where the manufacturing material is usually inextensible and uncompressible. The interpolating developable surfaces are approximated by a special triangulation called bridge boundary triangulation. The proposed modeling algorithm generates a bridge boundary triangulation by simulating the minimum energy folding process of a sheet when it is rolled from one end of the strip to another. Ample test examples are provided to validate the feasibility of the proposed modeling method. In summary, the following remarks can be made about the proposed algorithm:

- Compared to the existing wrinkle modeling schemes, which in general concern with only realism and ignore the developability (which though is an important requirement in manufacturing), our wrinkle modeling algorithm takes the developability of the final wrinkled surface into consideration;

- While maximizing the developable region of the final surface, the strain energy due to bending is also utilized for selecting succeeding edges in the triangulation; therefore, the bending energy on the final surface is minimized;

- Our algorithm is geometry-oriented, so it is fast and robust; and

- Our algorithm shows good response condition – it usually captures the original developable surface if it exists.

One major limitation of the proposed method is that it is a local optimization approach. Therefore, the solution might stagnate at some local optimum and misses the original developable surface (if it exists) or better approximations with less violation of local-convexity. Augmenting the local optimization with a priori rulings (e.g. Example IV shown in Fig. 12) may lessen the level of the local dependence and provides certain guidance for broader searches; more work is however needed in finding sound and realistic rules that will help automate the selection of a priori rulings. Another important issue, as a future study, is the design of the strip itself. Very often, when a strip is designed, one of the two curves is fixed (so to ensure the desired wrinkle pattern) while the other can have some degree of freedom for modification (e.g., its Hausdorff distance to a nominal curve must not exceed a certain amount). One potential extension is to use the proposed minimum folding energy based bridge boundary triangulation as a platform and couple it with the design of the strip – the boundary curve is interactively modified within the given constrain so to minimize (and hopefully remove all) the "red" edges in the corresponding BBT from the algorithm Min_Folding_BBT. This is currently under investigation.

8. References

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

[2] Terzopoulos, D., and Waters, K., 1990, "Physically-based Facial Modeling, Analysis, and Animation", Journal of Visualization and Computer Animation, 1, pp 73-80.

[3] Wu, Y., Thalmann, N. M., and Thalmann, D., 1994, "A Plastic-visco-elastic Model for Wrinkles in Facial Animation and Skin Aging", *Proc. Pacific Graphics '94*, pp 201-213.

[4] Wu, Y., Kalra, P., and Thalmann, N.M., 1997, "Physically-based Wrinkle Simulation & Skin Rendering", *Proc. Eurographics Workshop Computer Animation and Simulation '97*, pp 69-79.

[5] Bando, Y., Kuratate, T., and Nishita, T., 2002, "A Simple Method for Modeling Wrinkles on Human Skin", *Proc. of Pacific Graphics '02*, pp 166-175.

[6] Boissieux, L., Kiss, G., Thalmann, N.M., and Kalra P., 2000, "Simulation of Skin Aging and Wrinkles with Cosmetics Insight", *Proc. Eurographics Workshop on Computer Animation and Simulation 2000*, pp 15-27.

[7] Ishii, T., Yasuda, T., Yokoi, S. and Toriwaki, J., 1993, "A Generation Model for Human Skin Texture", *Proc. Computer Graphics International '93*, pp 139-150.

[8] Montagna, W., Kligman, A.M., and Charlisle, K.S., 1992, *Atlas of Normal Human Skin*, Springer.

[9] Viaud, M.L., and Yahia, H., 1992, "Facial Animation with Wrinkles", *Proc. Eurographics Workshop on Animation and Simulation '92*, pp 1-13.

[10] Vince, J., 2000, *Essential Computer Animation Fast*, Springer.

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

[12] Waters, K., 1987, "A Muscle Model for Animating Three-dimensional Facial Expression", *Proc. SIGGRAPH '87*, pp 17-24.

[13] Wu, Y., Kalra, P. and Thalmann, N.M., 1996, "Simulation of Static and Dynamic Wrinkles of Skin", *Proc. of Computer Animation '96*, pp 90-97.

[14] Wu, Y., Kalra, P., Moccozet, L. and Thalmann, N.M., 1999, "Simulating Wrinkles and Skin Aging", The Visual Computer, 15(4), pp 183-198.

[15] Baraff, D., and Witkin, A.P., 1998, "Large Steps in Cloth Simulation", *Proc. of SIGGRAPH '98*, pp 43-54.

[16] Breen, D.E., House, D.H., and Wozny, M.J., 1994, "Predicting the Drape of Woven Cloth using Interacting Particles", *Proc. SIGGRAPH '94*, pp 365-372.

[17] Terzopoulos, D., and Fleisher, K., 1988, "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture", *Proc. SIGGRAPH '88*, pp 269-278.

[18] Ng, H.N., and Grimsdale, R..L., 1996, "Computer Graphics Techniques for Modeling Clothes," *IEEE Computer Graphics and Applications*, *Computer Graphics in Textiles and Apparel*, pp 28-41.

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

[20] Combaz, J., and Neyret, F., 2002, "Painting folds using expansion textures", *Proc. Pacific Graphics '02*, pp 176-182.

[21] Aono, M., 1990, "A Wrinkle Propagation Model for Cloth," *CG International*, *Computer Graphics Around the World*, Springer-Verlag, pp 95-115.

[22] Kunii, T.L., and Gotoda, H., 1990, “Singularity Theoretical Modelling and Animation of Garment Wrinkle Formation Processes,” The Visual Computer, 6(6), pp 326-336.

[23] Fan, J., Wang, Q., Yuen, M.-F., and Chan, C. C., 1998, “A Spring-mass Model-Based Approach for Wrapping Cloth Patterns on 3D Objects,” The Journal of Visualization and Computer Animation, 9, pp 215-227.

[24] Perlin, K., 1985, “An Image Synthesizer”, *Proc. SIGGRAPH ’85*, pp 287-296.

[25] Perlin, K. and Hoffert, E.M., 1989, “Hypertexture”, *Proc. SIGGRAPH ’89*, pp 253-262.

[26] Fleischer, K.W., Laidlaw, D.H., Currin, B.L. and Barr, A.H., 1995, “Cellular Texture Generation”, *Computer Graphics*, 29 (Annual conference series), pp 239-248.

[27] Blinn, J.F., 1978, “Simulation of Wrinkled Surfaces”, *Proc. SIGGRAPH ’78*, pp 286-292.

[28] Kajiya, J.T. and Kay, T.L., 1989, “Rendering Fur with Three-dimensional Textures”, *Proc. SIGGRAPH ’89*, pp 271-280.

[29] Frey, W. H., 2002, “Boundary Triangulations Approximating Developable Surfaces that Interpolate a Closed Space Curve”, International Journal of Foundations of Computer Science, 13(2), pp 285-302.

[30] Fu, J., Joneja, A., and Tang, K., 2004, “Modeling Wrinkles on Smooth Surfaces for Footwear Design”, *Proc. CAD ’04.*

[31] Bradley, H.C., and Uhler, E.H., 1937, *Descriptive Geometry for Engineers*, International Textbook Company, Scranton, PA.

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

[33] Keppel, E., 1975, “Approximating Complex Surfaces by Triangulation of Contour Lines”, IBM Journal Res. Develop, pp.2-10.

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

[35] Wang, C.C.L. and Yuen, M.M.F., "Freeform extrusion by sketched input", *Computers & Graphics*, vol.27, no.2, pp.255-263, 2003.

9. Appendix

We prove that given a pair of polygonal chains with *m* and *n* vertices respectively, there are exactly a total of $\binom{m+n-2}{m-1} = \binom{m+n-2}{n-1} = \frac{(m+n-2)!}{(m-1)!(n-1)!}$ distinct bridge boundary triangulations between the two.

Let P and Q be two arrays, of size m and n respectively. Any bridge boundary triangulation of P and Q corresponds uniquely to an ordered list $\Psi = \{e_1, e_2, \ldots, e_{m+n-1}\}$ of $m+n-1$ bridge edges between P and Q, where the order of Ψ means the vertices of its edges are lexicographically ordered along P and Q. Excluding edge e_1, an edge $e_i = \langle P[j], Q[k] \rangle$ is said to be generated by a movement of P-type if $e_{i-1} = \langle P[j-1], Q[k] \rangle$, otherwise it is generated by a movement of Q-type as e_{i-1} must be $\langle P[j], Q[k-1] \rangle$. For example, in the bridge boundary triangulation shown in Fig. 16, edge $e_7 = \langle P[5], Q[3] \rangle$ is obtained by a P-type movement since $e_6 = \langle P[4], Q[3] \rangle$, while edge $e_{12} = \langle P[7], Q[6] \rangle$ is obtained by a Q-type movement since $e_{11} = \langle P[7], Q[5] \rangle$.

As the first edge $e_1 = \langle P[1], Q[1] \rangle$ is fixed, we need exactly $m+n-2$ movements to generate the remaining $m+n-2$ edges for a bridge boundary triangulation. Let us call an ordered list of $m+n-2$ movements a *valid* list of movements if it has exactly $m-1$ P-type movements and $n-1$ Q-type movements, otherwise it is an invalid list. It is not hard to see that only a valid list of movements can generate a bridge boundary triangulation. Conversely, it is also conceivable that any bridge boundary triangulation is corresponded by a unique valid list of movements. For instance, the bridge boundary triangulation depicted in Fig. 16 is generated by the valid list of movements {Q-type, Q-type, P-type, P-type, P-type, P-type, Q-type, Q-type, P-type, P-type, Q-type, P-type, P-type, Q-type, P-type}. Therefore, there is a one-to-one correspondence between a valid list of movements and a bridge boundary triangulation. Two valid lists (of movements) are distinct from each other if there is at least one position where the movements of the two lists are of different types. Therefore, a valid list is decided by positions in the list where the $m-1$ P-type movements are placed; obviously, there are a total of $\binom{m+n-2}{m-1} = \frac{(m+n-2)!}{(m-1)!(n-1)!}$ distinct ways for the placement. Consequently, there are exactly a total of $\binom{m+n-2}{m-1} = \binom{m+n-2}{n-1}$ distinct bridge boundary triangulations over a pair of P and Q with m and n vertices respectively.

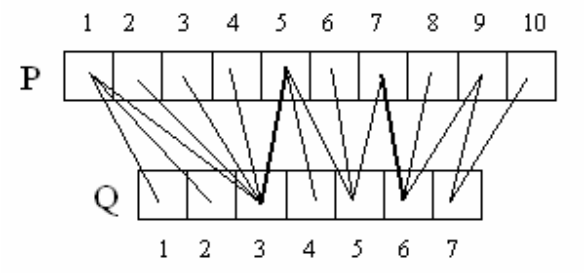


Fig. 16 Valid list of movements.