Design of Developable Interpolating Strips

Ming Chen[1], Kai Tang[2]* and Ajay Joneja[3]

[1] Shenzhen Graduate School, Harbin Institute of Technology, mecm@hitsz.edu.cn
[2]Dept of Mech Engg, Hong Kong Univ. of Sci. and Tech., mektang@ust.hk
[3]Dept of IELM, Hong Kong Univ. of Sci. and Tech., joneja@ust.hk

ABSTRACT

Surface development is used in many manufacturing planning operations, e.g. for garments, ships and automobiles. However, most freeform surfaces used in design are not developable, and therefore the developed patterns are not isometric to the designed surface. In some domains, the CAD model is created by skinning operations that interpolate smooth strips between a specified set of skeleton curves. In this paper, we propose a method to approximate a strip with a developable surface between the two space curves bounding it. We allow one of the bounding curves to be perturbed within a controllable tolerance and meet some other special engineering requirements. We formulate the problem as a combination of a discrete combinatorial optimization problem and a constrained nonlinear optimization problem, and propose an efficient iterative approach to solve the problem.

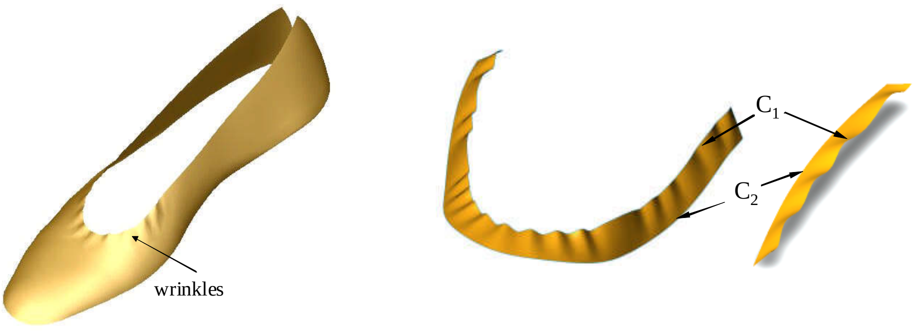
Keywords: developable surfaces, wrinkle design, garment design, optimization.
DOI: 10.3722/cadaps.2011.557-570

1 INTRODUCTION

We address a problem that arises in the CAD/CAM of garments and footwear. The designer often creates the required shape by interpolating, e.g. by skinning, between a pair (or sometimes a sequence) of skeleton curves. Such generator curves may even contain features such as aesthetic wrinkling, adding complexity to the designed surface [17]. For easy manufacture, it is desirable that such strips can be flattened into planar patterns with little or no distortion. Similar requirements also exist in sheet-metal applications [18, 22] and windshield design [13]. However, modeling developable surfaces is nontrivial and few current CAD modelers support it.

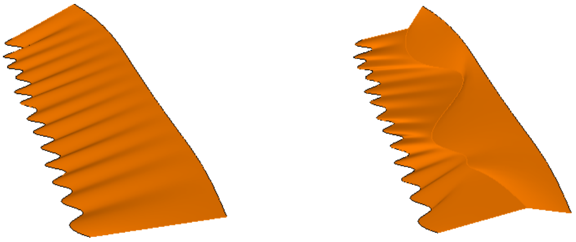
In this paper, we propose an algorithm for designing developable interpolating strips (see Fig.1). Informally, we isolate the problem as follows: given two curves, C_1 and C_2 that define a (possibly wavy) surface S interpolating them, we wish to approximate S by a developable surface S' by possibly allowing small variations of one of the generator curves, say C_2.

A simple approach would be to find an appropriate parametrization of C_1 and C_2 and then create a ruled surface interpolating them. However, in general such a ruled surface may be far from developable. At the same time, it is desirable that the interpolating surface is smooth in its interior – thus, the surface in Fig. 2 (a) is more desirable than the one in Fig. 2 (b).



(a) (b)

Fig. 1: (a) CAD model of a shoe upper with designed wrinkles (b) Example of a designed strip.



(a) (b)

Fig. 2: (a) A smooth surface interpolating the generator curves, (b) An undesirable interpolation.

The rest of the paper is organized as follows. Section 2 reviews the related works; section 3 presents our approach, followed by the problem formulation and an outline of our proposed algorithm. Sections 4 and 5 give details of our methodology, and section 6 briefly describes how we can smooth the resulting surface. Section 7 gives some experimental results, and section 8 concludes the paper with some discussions.

2 RELATED WORK

There have been a few studies of developable surfaces, especially in the context of NURBS or B-Spline surfaces [1-3, 9, 12, 19, 20]. [3] proposed the condition under which a developable Bezier surface can be constructed with two boundary curves. The boundary curves in his approach are of at most degree 3 and are restricted to lie in parallel planes. Their work was extended by Frey and Bindschadler [12] by generalizing the degree of the directrices. Maekawa and Chalfant [20] extended Aumann's algorithm to B-Spline curves by segmenting the original B-Spline curves into multi-segment planar Bezier curves. Chu and Sequin [9] proposed a new method to design a developable Bezier patch. In their method, after one boundary curve is freely specified, five more degrees of freedom are available for a second boundary curve of the same degree. Aumann [1, 2] utilized De Casteljau algorithm to design developable Bezier surfaces through a Bezier curve of arbitrary degree and shape. However, the approach often results in unexpected or undesirable surfaces. Projective geometry methods exploiting point-plane duality were investigated by the groups of Ravani and Pottmann [4, 5, 15, 23, 24].

Some researchers have used spatial kinematics and line geometry to approximate a given surface by a developable one [7, 14, 16]. The idea is to generate a developable surface sweeping a line along a helical motion. This method is widely used in reverse engineering. Given a set of scattered data points $\{P_j\}$, it is required to find a developable surface of which the generators g_j are as close as possible to the given points. In Hoscheck and Schneider [14, 16], the distance from a point to a line was used, leading to a nonlinear optimization problem. Pottmann and Wallner [24] and Hoscheck and Schwanecke [14, 16] independently introduced the error measurement between planes, leading to linear algorithms. Special attention was paid to controlling the regression curve and how to combine the pieces of patches with imposed continuity conditions.

Recently, methods based on discrete data representation have been proposed for design of developable surfaces, owing to the rapid increase of computing power and popularity of 3D meshes. In this manner, triangle or quad meshes are sought to achieve maximum developability, with the given interpolating constraints satisfied. In his pioneering work, Frey [11] showed how to approximate buckled binder wrap surfaces by calculating out the d-vertices based on the fact that Gaussian curvature at every point on a developable surface is zero, and this condition can be expressed by requiring the included angle at every internal vertex to be 2π. In [21], triangle strips are designed by grouping original triangles on the mesh which share similar topological distances, and the resulting pattern gives out a near-developable unfolding of the mesh. Wang and Tang [27] minimized the total discrete Gaussian curvature for polygonal surface by relocating each mesh vertex. Tang and Chen [25] satisfied some interpolation requirements and minimized the developability change by a mesh deformation method and the method was further applied into cloth simulation [8]. In [26, 28, 29] efficient algorithms are given for finding the best parameterization of an interpolating ruled surface for a range of optimization objectives, developability included. The algorithm given in [28] is based on the well-known Dijkstra's shortest path algorithm, is deterministic, and is able to find the global optimum. It will be a key component in our optimization algorithm.

3 PROBLEM FORMULATION

This section gives the necessary preliminaries, followed by our formulation of the problem as a constrained optimization.

3.1 Optimal Ruled Surface

A ruled surface, $S(u,v)=C_1(u)+(C_2(u)-C_1(u))v,\ u,v\in[0,1]$, is developable if $dC_1/du, dC_2/du$ and $C_2(u)$-$C_1(u)$ are co-planar for all u in the parameter domain of S (where $C_1(u)$, $C_2(u)$ are called directrices and $C_2(u)$-$C_1(u)$ is a direction vector of a ruling) [doCarmo76]. Thus, given a pair of directrices, different ruled surfaces can be defined depending on their parameterization, see Figure 3. If we define a monotone parameterization function, $\xi(u)$: [0,1] → [0, 1], an interpolating ruled surface of C_1 and C_2 then is defined as $S(u,v)=(1-v)C_1(u)+vC_2(\xi(u)).,\ (u,v\in[0,1])$.

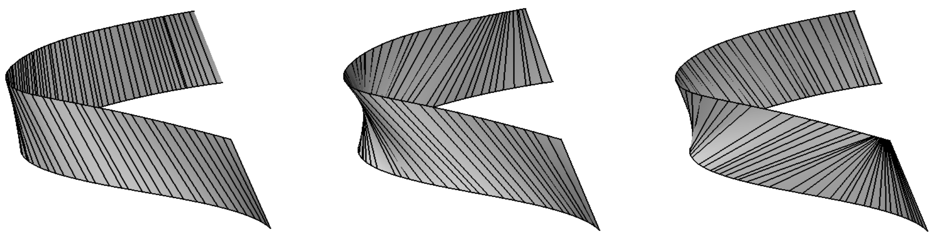
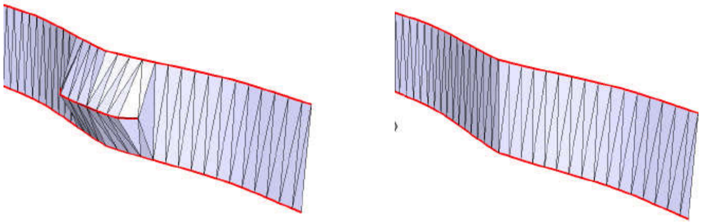


Fig. 3: Different parameterizations on the same two rails lead to different ruled surfaces.

Given two directrices, we seek the optimum mapping $\xi(u)$ that will maximize the developability of the resulting ruled surface. Finding such a mapping in analytical form is difficult, but when the two directrices are given in discrete form (i.e., polygons), an optimal mapping can be found efficiently [Wang05a].

3.2 BBT Mesh Representation of a Strip

We assume that the given strip has no interior cut-lines such as the ones shown in Fig. 4(a), i.e. it should have a form as shown in Fig. 4(b); note that situations like Fig. 4(a) can be subdivided into a set of strips that have the regular structure as desired.



(a) (b)

Fig. 4: Interpolating strips in BBT form (a) with internal cut-seams (b) no internal cuts.

We represent an interpolating strip using BBT (Bridge Boundary Triangulation) as in [Wang05a], see Fig. 4(b). A BBT approximates a ruled surface by a collection of triangles whose vertices lie only on the rails. More specifically, let $P=\{p_1,p_2,...,p_m\}$ and $Q=\{q_1,q_2,...,q_n\}$ be an ordered dense points sampling of the boundary curves C_1 and C_2, respectively. A collection of triangles $\mathrm{T}=\{\mathrm{T}_1,\mathrm{T}_2,...,\mathrm{T}_\mathrm{N}\}$ defined on *P* and *Q* is a BBT if all the following criteria are met:

1) T_1 is defined by line segment $< p_1,q_1 >$ and one of p_2 and q_2 and T_N is defined by $<p_\mathrm{m},q_\mathrm{n}>$ and one of p_{m-1} and q_{n-1};
2) each triangle T_k, $1 < k <$ N, is defined by a line segment $<p_i,q_j>$, called a bridge edge, for some *i* and *j*, and either vertex p_{i+1} or vertex q_{j+1};
3) every p_i and q_j belongs to at least one triangle; and finally
4) both *P* and *Q* are partial orderings of T: for any two vertices p_i and p_j with $i < j$, all the triangles having p_i as a vertex are earlier constructed than all the triangles having p_i as a vertex in T; the same property applies to *Q*.

The above BBT, T, can be interpreted as a discrete approximation of desired mapping $\xi(u)$. Note that it cannot represent all $\xi(u)$ due to the introduction of criteria 4; however, this criteria is necessary to disallow undesirable self-intersections. This discrete approximation approaches a continuum $\xi(u)$ as *n* and *m* increase, defining a ruled surface in the limit. Finding the best desired mapping $\xi(u)$ is thus translated into the equivalent problem of finding a BBT with maximum developability. To quantify this notion, we discuss below a metric for developability.

3.3 Evaluation of Developability in BBT

The *normal twist* T_E of a bridge edge $<p_i,q_j>$ (bold line segment in Fig.5) is defined as $T_E(<p_i,q_j>)=1-n_p.n_q$, where n_p and n_q are the unit polygon normal vectors at the point p_i and q_j respectively.

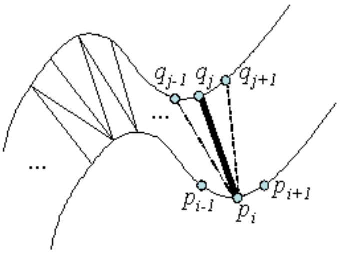


Fig.5: A bridge edge $< p_i, q_j >$ in a BBT.

n_p and n_q can be calculated as: $n_p = \frac{(p_i - q_j) \times t_p}{\| (p_i - q_j) \times t_p \|}$ and $n_q = \frac{(p_i - q_j) \times t_q}{\| (p_i - q_j) \times t_q \|}$, where t_p and t_q are the unit tangents on the boundary curves at points p_i and q_j, respectively. During the movement of poly-line PL_1 (curve C_1), the original parametric formula of the curve C_1 will not apply to calculate the tangent at p_i, thus t_p cannot be evaluated directly from its original parametric formula, nor by standard numerical method such as backward-forward or central difference method. Considering that most curves in practice are polynomial curves, we locally fit the points p_{i-1}, p_i and p_{i+1} by a quadratic curve $C(t)=a_0+a_1t+a_2t^2$. So, t_p can be calculated as $a_1 + 2\gamma a_2$, where $\gamma = \frac{\| p_i - p_{i-1} \|}{\| p_i - p_{i-1} \| + \| p_i - p_{i+1} \|}$. We only need the direction of t_p, which is given by:

$$\boldsymbol{t}_p = (p_i - p_{i-1})(1-\gamma)^2 + \gamma^2 (p_{i+1} - p_i) \tag{1}$$

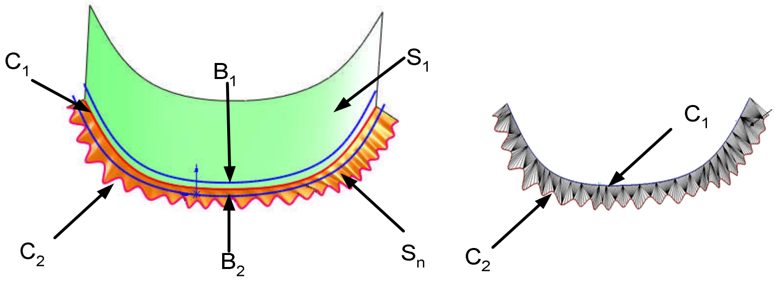
The *BBT normal twist* $T_w(T)$, for a given BBT, T, is defined as the sum of the normal twists of all the bridge edges in T. $T_w(T)$ is always non-negative, and is zero if and only if the integral normal twist of every bridge edge $< p_i, q_j >$ in T is zero, in which case the strip is developable.

3.4 Optimal BBT for a Fixed PL_1

When PL_1 is fixed, we can find a BBT T of *P* and *Q* that minimizes the total twist $T_w(T)$. We will use $T_{\min}(P,Q)$ to represent this optimal BBT. Using the BBT normal twist as the minimization objective, and given P and Q with *m* and *n* vertices respectively, $T_{\min}(P,Q)$ can be found in O((*mn*)***ln***(*mn*)) time and space [Wang05a]. Note that even for the modest *m* and *n*, the total number of distinct BBTs of P and Q is combinatorial -- there are a total of $\binom{m-1}{m+n-2} = \binom{n-1}{m+n-2} = \frac{(m+n-2)!}{(m-1)!(n-1)!}$ distinct BBTs to search through for the optimum $T_{\min}(P,Q)$. The algorithm in [Wang05a] establishes an equivalence between this search problem and a multi-layer graph search problem that allows an efficient solution.

3.5 The Problem and the Outline of Algorithm

In Fig. 6, S_1 is a ruled surface and C_1 is a curve embedded on S_1. In garment applications, S_1 is usually developed with its own corresponding flattened panel. Curve C_2, referred to as the *pattern curve*, bears design intent of wrinkles and is fixed. Curve C_1 however can move within a band on S_1 between two embedded curves B_1 and B_2. We call curve C_1 the *objective* or *design curve*. From manufacturing consideration, the interpolating surface S_n should be developable and, in order to ensure that S_n can be sewed together with S_1, we should design C_1 under the condition that C_1 is on surface S_1.



(a) (b)

Fig.6: Schematic descriptions of curves and surfaces involved.

Assuming that curves C_1 and C_2 are in discrete form as polygons P and Q, respectively, any S_n is a BBT as shown in Fig. 6(b). The variational optimization problem to be solved can be formulated as:

$$\arg\min_{P,T} T_W(T) \quad \textit{subject to} \begin{cases} \text{P is on } S_1 \\ \text{P is inside the band between } B_1 \text{ and } B_2 \\ \text{others} \end{cases}$$

where T represents any BBT of a fixed P and Q and the "others" in the constraint field indicate some other constraints during the modification of P (i.,e., C_1) such as the avoidance of undesirable distortion, to be discussed in the following section. The algorithm we use to solve this is outlined below:

Algorithm Developable_Strip_Design ()
Input: curve C_2, initial curve C_1, surface S_1, bounding curves B_1 and B_2.
Output: a BBT S_n
Step 0: $P, Q \leftarrow$ discretizations of C_1 and C_2
While (termination conditions are not met) **Do:** {
  Step 1: $\xi \leftarrow$ the mapping of the BBT $T_{min}(P, Q)$
  Step 2: Move P in the band of B_1 and B_2 on S_1 to minimize the total twist of the BBT with the same ξ;
  Go to Step 1.
}
$S_n \leftarrow T_{min}(P, Q)$.

In the algorithm, Step 1 is achieved by the method in [Wang05a]. The rest of the paper then will focus on Step 2. The algorithm terminates when any of the three following requirements is met:

i) $T_w(T)$ is less than a given threshold.

ii) The improvement of $T_w(T)$ between consecutive iterations is less than a given threshold (<0.005).

iii) The number of iterations exceeds a given number. (In our example, this is set as 10).

4 LOCAL OPTIMIZATION

Considering the computation efficiency and easy control of the final shape of curve C_1 when perturbing curve C_1, the strategy that we first adopt is a one-point-movement scheme, i.e., at Step 2, the vertices of *P* are moved one by one, and each and every vertex is moved once and only once. In order to improve the efficiency, we only update the vertex p_i when its incident edge integral normal twist $T_E(<p_i, q_j>)$ is larger than a specified threshold. When moving vertex p_i, one alters the integral normal twist $T_E(e)$ of any bridge edge *e* in the current BBT, as long as *e* is incident to one of p_{i-1}, p_i, or p_{i+1}, e.g., the colored edges in Fig. 7. To better describe this relationship, let $\Phi(p_i)$ be the set of the incident bridge edges of p_i, e.g., $\{q_j, q_{j+1}, \ldots, q_{j+4}\}$ in Fig. 7.

So, for each vertex p_i, we denote $\sum_{e\in\Phi(p_i)} T_E(e)$ by $E(p_i)$, and choose $Sum(p_i) = E(p_{i-1}) + E(p_i) + E(p_{i+1})$ as the final objective function for the movement of p_i in algorithm Developable_Strip_Design.

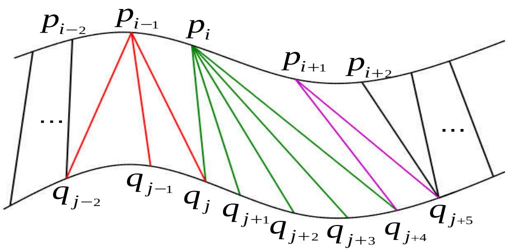


Fig. 7: Bridge edges whose integral normal twists are affected by vertex p_i.

4.1 *v*-Direction Optimization

As already stated, the movement of p_i should be restricted to the original ruled surface S_1, so that the interpolating surface S_n can be sewed together with S_1. We propose a particular movement scheme called *v-directional-only optimization* in which p_i moves along the ruling direction of S_1. As p_i is moved along a ruling of S_1, it must stay on S_1. Another benefit of this method is that the final curve C_1 will not be badly distorted or twisted as compared to its initial form.

S_1, B_1 and B_2 (see Fig. 6) are known in advance; we assume that S_1 has no self-intersection, and is given as: $S_1(u,v) = vR_1(u) + (1-v)R_2(u)$, $u, v \in [0,1]$, where R_1 and R_2 are the corresponding directrix curves. B_1 and B_2 are embedded curves on S_1, which are defined by mappings from the *uv* parameter space to the object space of S_1, and the parametric forms of the two curves can be expressed as:

$$B_1(t_1) = S(u_1(t_1), v_1(t_1)) = v_1(t_1)R_1(u_1(t_1)) + (1-v_1(t_1))R_2(u_1(t_1)) \qquad t_1 \in [0,1]$$

$$B_2(t_2) = S(u_2(t_2), v_2(t_2)) = v_2(t_2)R_1(u_2(t_2)) + (1-v_2(t_2))R_2(u_2(t_2)) \qquad t_2 \in [0,1]$$

For point $p_i = S_1(u_0, v_0)$, the unit ruling vector is calculated as: $\tau = \dfrac{R_2(u_0) - R_1(u_0)}{\| R_2(u_0) - R_1(u_0) \|}$, and the point p_i moves along the direction τ within an interval. The ruling passing through p_i will intersect B_1 and B_2 at two points, say M_1 and M_2: $M_1 = S_1(u_0, v_1(t_1))$, $M_2 = S_1(u_0, v_2(t_2))$, where $t_1 = u_1^{-1}(u_0)$, and $t_2 = u_2^{-1}(u_0)$.

Thus, $[v_1, v_2]$ is the interval for the movement of p_i, with $v_0 \in [v_1, v_2]$ being the initial *v*-value.

4.2 The Objective Function

Since the interval $[v_1, v_2]$ is often narrow, the change $\delta \in [v_1 - v_0, v_2 - v_0]$ over the initial value v_0 is small. This suggests that we can directly expand our objective function Sum(p_i) at $v = v_0$ into its Taylor series of an acceptable degree. Here, we can expand it into a polynomial of variable v. We set:

$$\mathrm{Sum}(p_i(v)) = \mathrm{Sum}(p_i(v_0)) + \beta_1\delta + \beta_2\delta^2 + +\beta_3\delta^3 + \beta_4\delta^4 + 0(\delta^5) \tag{2}$$

In practice, the coefficients β_i are obtained by sampling uniformly distributed samples of δ. Thus the original problem is formulated into a constrained univariate optimization. Eq. (2) is locally monotonic in each interval separated by the bound and its stationary points, thus once all stationary points between the interval $[v_1 - v_0, v_2 - v_0]$ are found, the interval is further divided into k+1 sub-intervals, where k is the number of the stationary points in $[v_1 - v_0, v_2 - v_0]$. Because of the monotonic properties of each sub-interval, local optimum δ for each subinterval can be easily found, and the best value for δ can be selected from these k+1 local optima. The stationary points of Eq. (2) are computed by using a standard polynomial root-finding technique. After the optimal δ is selected, vertex p_i is correspondingly updated. The algorithm we use to locally update p_i is outlined below:

Algorithm Local_Opt_Strip ()
Input: Initial directrix curve R_1 and R_2, $S_1(u,v)$, objective curve C_1, bounding curves B_1 and B_2.
Output: Optimal $PL_1 = \{p_i\}$
While ($i \le m$) **Do:** {

Step 1: $p_i \leftarrow$ Select p_i from curve C_1 as the objective point.

Step 2: Calculate the ruling direction τ of point p_i; the ruling line passing p_i intersects B_1 and B_2 at point M_1 and M_2; for p_i, M_1 and M_2, evaluate v_0, v_1, and v_2, respectively, from uv parameter space of S_1.

Step 3: Sample δ uniformly in the interval $\delta \in [v_1 - v_0, v_2 - v_0]$ to obtain Eq.(2).

Step 4: Search the optimal δ for the Eq.(2) in the interval $[v_1 - v_0, v_2 - v_0]$ based on standard polynomial root-finding technique. }

Step 5: Update p_i as $S_1(u_0, v_0+\delta)$。

---

5 GLOBAL OPTIMIZATION

While the local optimization strategy is efficient, we also wish to relax the constraint on the points, allowing them to move in both u- and v-directions of S_1 to explore for better solutions. Below, we consider how to do so, while allowing all vertices of PL_1 to move in each iteration step.

As point p_i is now allowed to move in both u and v directions, if no constraints are imposed, the final C_1 may be distorted severely and undesirably. To avoid this, we add some restrictions.

Constraint 1: *Shape preservation constraint*: The final curve shape should remain close to the original shape defined by poly-line PL_1, thus we need to minimize the distances of the perturbed PL_1 vertices from the original positions by minimizing:

$$E_{close} = \sum_{i=1}^{m} \| p_i^* - p_i \|^2 \tag{3}$$

where m is the number of vertices on PL_1 and p^*_i is the current updated point of the original p_i.

Constraint 2: *Band constraint.* This constraint pertains to the *v*-direction-only case; that is, the curve C_1 should always lie inside the band delimited by the two boundary curves $B_1(u_1(t),v_1(t))$ and $B_2(u_2(t),v_2(t))$. We denote the implicit functions of B_1 and B_2 in the *uv* domain as $F_1(u,v)$ and $F_2(u,v)$, respectively. So $p_i(u,v)$ should meet the following band restrictions:

$$E_{band} = \begin{cases} F_1(u,v) \le 0 \\ -F_2(u,v) \le 0 \end{cases} \tag{4}$$

These constraints, along with developability, allow us to define the Lagrangian function as:

$$E_{final} = w_1 T_W(\mathrm{T}) + w_2 E_{close} + \lambda^T E'_{band} \tag{5}$$

In (3), constants w_1 and w_2 are used to control the relative weight of geometric fidelity and the developability of the final shape, λ are Lagrange multipliers, and E'_{band} are only subsets of constraints E_{band} in Eq. (5). In each iteration of the numerical solver, only the subset of these constraints for which the equality holds in (3), called the active set, are used to update the incumbent solution. The remaining constraints are ignored, and the working set is updated in the next iteration. A sequential quadratic programming (SQP) approach [Boggs95] was adopted to solve Eq. (5). SQP formulates the objective function to solve a sequence of quadratic programming (QP) sub-problems. Each QP sub-problem minimizes a quadratic model of a certain modified Lagrangian function subject to linear constraints. In each iteration, the Hessian matrix and gradients of Eq. (6) are computed to approximate the objective function as a local quadratic problem at the current point. The Hessians and gradients of E_{band} and E_{close} with respect to *uv* parameters of $S_1(u,v)$ are derived analytically; for $T_w(T)$, the Hessians and gradients are evaluated numerically.

Let us rewrite Eq. (5) in the form $F(x,\lambda) = f(x) - \lambda^T E'_{band}(x)$, where x denotes the unknown *uv* coordinates of vertex p_i of PL_1. Let J be the Jacobian matrix of the constraints E_{band}, and H denote the Hessian matrix of $F(x, \lambda)$. The update step x → x + Δx is solved from:

$$\begin{bmatrix} H & -J^T \\ -J & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ E'_{band}(x) \end{bmatrix} \tag{6}$$

The numerical method maintains feasibility by adaptively shortening the step size when required, using x → x + αΔx. If all the corresponding constraints belong to the working set, then α is set as 1, otherwise the corresponding step size should satisfy $E_i + \nabla E_i(\alpha \Delta x_i) \le 0$, where $E_i \in \{E_{band} - E'_{band}\}$ and ∇E_i is the gradient of the constraints. In summary, α is computed as: $\alpha = \min(1, \min\left(\frac{-E_i}{\nabla E_i \Delta x_i}\right))$. Any constraint that yields $\alpha < 1$ is a *blocking constraint* and is added to the working set for the next iteration. After solving Eq. (6), we check the components of λ: if $\lambda_i < 0$, we can decrease E_{final} further by dropping some certain active constraints E_i from the working set, then the iteration is repeated. The coefficient matrix of Eq. (6) is highly sparse and has size $(2m+N)(2m+N)$, where m is the vertex number of PL_1 and N is the number of active constraints. The SQP algorithm terminates when at least one of the following conditions is met:

(*i*) the current x and λ_i meet the KKT condition, (*ii*) the maximum number of iterations is exceeded, (*iii*) the step length for the current x becomes too small. In practice, we find that the initial value significantly affects the algorithm performance and the results, but the iterations converge rapidly in the neighborhood of the optimum. Thus a practical solution is to use the local method to quickly derive a starting point, and then run the global method to locate the final positions of *P*.

Algorithm Global_Opt_Strip ()

Input: Objective curve C_1=PL_1= { p_i }, *bounding condition* E_{band}, and *shape preservation condition* E_{close} and Set $E'_{band} = \Phi$, w_1=15, w_2=1

Output: Optimal PL_1 = { p_i }

While (Termination conditions are not met) **Do:** {

Step 1: Construct the Lagrangian function Eq.(5).

Step 2: Rewrite Eq.(5) as $F(x,\lambda)=f(x)-\lambda^T E'_{band}(x)$, where x denotes the unknown uv coordinates of vertex p_i of PL_1.

Step 3: Calculate Jacobian Matrix of the constraints E'_{band} and Hessian matrix of $F(x, \lambda)$, and obtain the linear equation Eq.(6), solve Eq.(6) and get the solution x and λ_i.

Step 4: Update working set E'_{band} and uv coordinates of vertex p_i

4.1 **If** $E_i + \nabla E_i(\Delta x_i) > 0$, **Do** x → x + Δx, where $E_i \in \{E_{band} - E'_{band}\}$

4.2 **Else Do: {**

4.2.1 Add E_i **to** E'_{band}

4.2.2 x → x + αΔx, where $\alpha = \min(1, \min\left(\frac{-E_i}{\nabla E_i \Delta x_i}\right))$, ∇E_i is the gradient of the constraints E_i. }

Step 5: **If** $\lambda_i < 0$, Remove E_i from E'_{band}, and Go to Step 2.}

Step 6: Update PL_1

6 SMOOTHING PROCESS

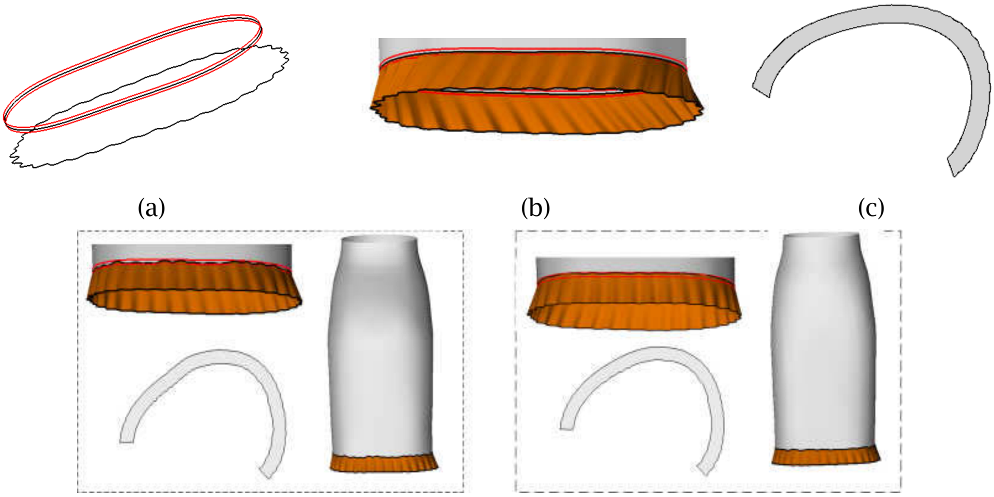
It is desirable that the final curve C_1 is smooth. To achieve this, a smoothing operation based on the Gaussian kernel $g(z) \propto \exp(-z^2/(2\sigma^2))$ is applied to the final P to remove unwanted zigzag points. Zigzag points can be identified either by user-interaction, or by comparing the length ratios $\frac{\| p_i - p_{i-1} \|}{\| p_{i-1} - p_{i-2} \|}$ and $\frac{\| p_{i+1} - p_i \|}{\| p_{i+2} - p_{i+1} \|}$; they are modified by using an averaging formula: $p_i = \frac{\sum_{p_j \in \Omega} p_j g(l(p_j))}{\sum_{p_j \in \Omega} g(l(p_j))}$, where Ω is the set of neighboring points of the point p_i plus p_i itself. In our system, we use $\Omega = \{p_{i-2}, p_{i-1}, p_i, p_{i+1}, p_{i+2}\}$, with the vertices near the ends treated appropriately if C_1 is open; $l(p_j)$ is the curve length on C_1 between p_j and p_i. The parameter σ in the original Gaussian kernel provides a way to control how the neighboring points impact the new p_i and hence the level of smoothing: if σ = 0, none of the neighboring points will have any impact on p_i, while an infinitely large σ would give a same weight to all the neighboring points and the new p_i is set as their arithmetic mean. The smoothed points may lie off the surface S_1, so we finally project them back to S_1 by using a Newton iteration method. In some applications, discrete points are not desired and users hope to use a smooth parametric curve to fit the vertices of P. Our implementation also provides this option based on the least squares fitting strategy. The details are omitted here.

7 EXPERIMENTAL RESULTS

The proposed algorithms were programmed in C++, and tested on a 3GHz, 2GB DDR2 PC. We generated strips for two given curves using the method [Wang05a] as our benchmark. All final strips were developed onto a plane and our algorithm was verified by comparing both, the integral edge normal twist and the distortion rate, i.e., $\Delta A = (A-A')/A$, where A denotes the total triangle area of the strip and A' is that of the corresponding 2D planar pattern. The constants σ , w_1 and w_2 are initially set as 15, 1 and 20, respectively; all computational statistics are given in Table 1.

Example I.

Fig. 8 shows a dress skirt to which a wrinkled strip is added at the bottom. The initial curves (B-spline curves of degree 3) are given as shown in Fig. 8(a): a fixed wavy black curve, and a black design curve that is allowed to move within a tolerance zone (depicted by two red curves). The design curve and bounding curves belong to one surface S_1, and the pattern curve is on the other surface S_n. The initial shape is shown in Fig. 8(b), and its planar development in Fig. 8(c). The local and global optimized strips together with the corresponding planar patterns and final skirts are shown in Fig. 8(d) and Fig. 8(e), respectively. From the table and the figures, we can see that both local and global optimization methods significantly reduce the total normal twist and area change rates while maintaining an acceptable design curve.



(a) (b) (c)

(d) (e)

Fig. 8: Design of a skirt frill. (a): the inputs including two bounding curves (red), which are on the same surface S_1 and one fixed wavy pattern curve (lower black curve) and one design curve (upper black curve) which will be perturbed between the bounding curves and at the same time kept on S_1; (b): the initial strip using the method [Wang05a] and it is adopted as a benchmark; (c): the flattened planar pattern after unrolling the strip in (b); (d): the results using the local method and (e): the results using the global method.

Example II.

Fig. 9(d) shows a designed strip that provides a transition between the back and toe parts of a shoe upper. The initial state is determined just as Example I. As expected, total normal twist and distortion rates can be greatly reduced making the final strip more developable and allowing the strip to be developed with significantly reduced distortion than the original design.

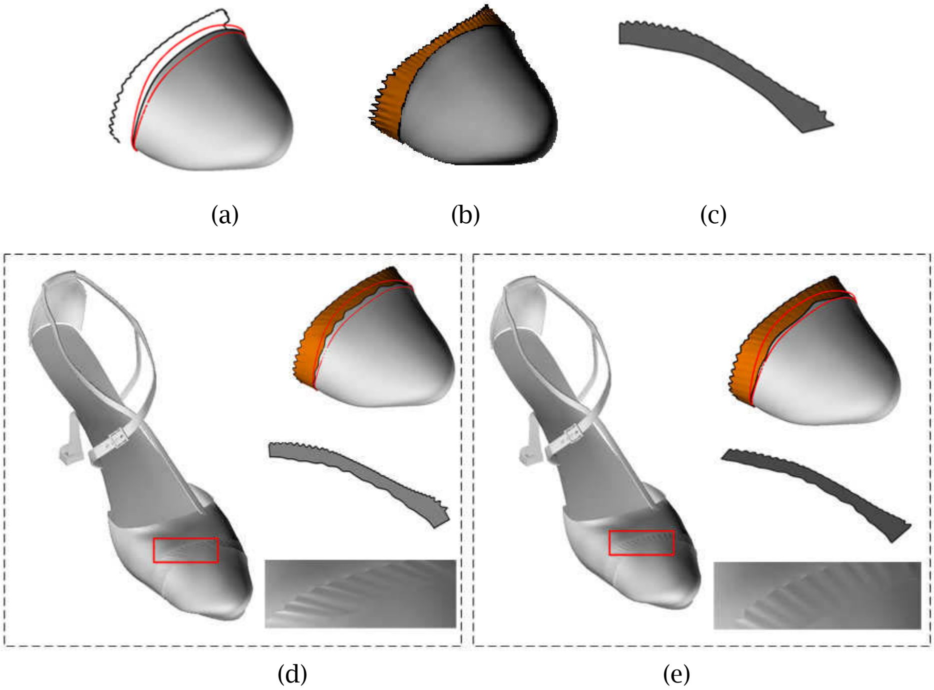


Fig. 9: Design of a transition narrow strip for a women's shoe. (a): the inputs including two bounding curves (red), which are on the head surface S_1 and one fixed wavy pattern curve (left-back black curve) and one design curve (right-front black curve) which will be perturbed between the bounding curves and at the same time kept on S_1; (b): the initial strip using the method [Wang05a] and it is adopted as a benchmark; (c): the flattened planar pattern after unrolling the strip in Figure 9(b); (d): the results using the local method; and (e): the results using the global method.

Example	*# of Points*	*Figure*	$T_w(T)$	ΔA *(%)*	*Time (sec)*	*# of Iterations*
Example I	*m*=500, *n*=800	Fig.8(b)	390.21	9.91%	0.76	N/A
		Fig.8(d)	57.33	2.03%	5.71	3
		Fig.8(e)	9.46	-0.21%	25.30	3
Example II	*m*=300, *n*=500	Fig.9(b)	75.50	4.82%	0.22	5
		Fig.9(d)	25.76	1.79%	2.12	5
		Fig.9(e)	3.25	0.18%	22.87	5

Tab 1: Computational statistics.

The examples also verify that while the local method is faster, the global method converges to a much better solution, as expected.

8 CONCLUSION

In many industrial applications, designers are required to determine the shape of an interpolating surface of maximum developability between two given space curves. In this paper, we propose a method that interpolates the input curves by a ruled surface with maximum developability, allowing one of the input curves to be perturbed within a band of specified tolerance. The algorithm first utilizes a previous work efficiently finding the best parametric correspondence of two space curves for maximizing the developability of their ruled surface, and then introduces two constrained nonlinear optimization schemes that help establish the geometry of the design curve for a fixed parametric correspondence. These two approaches can be used in sequence to converge to a good solution efficiently. Initial experiments have shown that the proposed algorithm is fast, numerically stable, and yields a surface significantly more developable than alternate interpolating surfaces generated by CAD systems.

Our work is restricted to strips. One possible extension is to allow the interpolating ruled surface to be of relatively arbitrary shape. In such situations the developability is still desired but requiring the two curves to be the directrices may be too restrictive. Another future work is to allow the use of cut-lines and find a series of narrow strips to interpolate them. Finally, the proposed method is a discrete method, and its efficacy depends somewhat on the sampling density of the input curves. The trade-off, as the sampling density increases, is between higher developability and computation time. A possible extension to our work is to apply an adaptive sampling method that allows high density as well as better computation times.

9 ACKNOWLEDGEMENTS

Thanks to Prof. C.C.L Wang for the source code to evaluate the optimal BBT, which was used as part of our algorithm. The authors also acknowledge the support of this work by RGC thorough GRF grants #614407 and #614205.

REFERENCES

[1] Aumann, G.: A simple algorithm for designing developable Bézier surface, Computer Aided Geometric Design, 20, 2003, 601-616. DOI:10.1016/j.cagd.2003.07.001.
[2] Aumann, G.: Degree elevation and developable Bézier surfaces, Computer Aided Geometric Design, 20, 2004, 661-670. DOI: 10.1016/j.cagd.2004.04.007.
[3] Aumann, G.: Interpolation with developable Bézier patches, Computer Aided Geometric Design, 8, 1991,409-420. DOI: 10.1016/0167-8396(91)90014-3.
[4] Bodduluri, R.M.C.; Ravani B.: Design of developable surfaces using duality between plane and point geometries, Computer Aided Design, 25(10), 1993, 621-632. DOI: 10.1016/0010-4485(93)90017-I.
[5] Bodduluri, R.M.C.; Ravani B.: Geometric design and fabrication of developable Bezier and B-spline surfaces, ASME Transaction, Journal of Mechanical Design, 116, 1994, 1024-1048. DOI: 10.1115/1.2919485.
[6] Boggs, T.; Tolle, J.W.: Sequential quadratic programming, Cambridge, UK: Cambridge University, 1995.
[7] Chen, H.Y.; Lee, IK Leopoldseder, S.; Pottmann, H.; Wallner, J.: On surface approximation using developable surfaces, Graphical Models and Image Processing, 61(2), 1996, 110-124. DOI: 10.1006/gmip.1999.0487.
[8] Chen, M.; Tang K.: A Fully Geometric Approach for Developable Cloth Deformation, The Visual Computer, 26(6-8), 2010, 853-863. DOI: 10.1007/s00371-010-0467-5.
[9] Chu, C.H.; Séquin, C.H.: Developable Bézier patches: properties and design, Computer Aided Design, 34(7), 2002, 511-527. DOI: 10.1016/S0010-4485(01)00122-1.
[10] Do Carmo, M.: Differential geometry of curves and surfaces, Englewood Cliffs, NJ: Prentice-Hall, 1976.
[11] Frey, W.: Modeling buckled developable surface by triangulation, Computer Aided Design, 36(4), 2004, 299-313. DOI: 10.1016/S0010-4485(03)00105-2.
[12] Frey, W.H.; Bindschadler, D.: Computer aided design of a class of developable Bézier surfaces, General Motors R&D Publication R&D- 8057, 1993.
[13] Hinds, B.K.; McCartney, J.; Woods G.: Pattern development for 3D surfaces, Computer Aided Design, 23(8), 1991, 583-592. DOI: 10.1016/0010-4485(91)90060-A.
[14] Hoschek, J.; Schwanecke, U.: Interpolation and approximation with ruled surfaces, in R. Cripps, editor. The mathmematics of surfaces VIII. Information Geometers. Winchester, 1998, 213-232.

[15] Hoschek, J.J.: Surfaces, in: R.E. Barnhill and W. Boehm, editors. Surfaces in computer aided geometric design. Amsterdam, North Holland, 1983:147-156.

[16] Hoschek,J.; Schwanecke, U.: Interpolation and approximation with developable surfaces, in: Le Mehaute, A., Rabut, C. and Schumaker, L.L., editors. Curves and surfaces with applications in CAGD. Nashville: Vanderbilt University, 1997,185-203.

[17] Joneja, A; Fu, J.; Tang, K: Modeling Wrinkles on Smooth Surfaces for Footwear Design, Computer-Aided Design, 37(8), 2005, 815-823. DOI: 10.1016/j.cad.2004.09.010.

[18] Lamb, T.: Shell development computer aided lofting – Is there a problem or not?, Journal of Ship Production,11(1),1995,34-46.

[19] Lang, J.; Röschel, O.: Developable (1, n)-Bézier surfaces, Computer Aided Geometric Design, 9, 1992, 291-298. DOI: 10.1016/0167-8396(92)90036-O.

[20] Maekawa, T.; Chalfant, J.S.: Design and tessellation of B-spline developable Surfaces, ASME Transaction Journal of Mechanical Design, 120, 1998, 453-461. DOI:10.1115/1.2829173.

[21] Mitani, J.; Suzuki, H.: Making papercraft toys from meshes using strip-based approximate unfolding, ACM Transaction Graphics, 23(3), 2005, 259-263. DOI:10.1145/1015706.1015711.

[22] Norlan, T.J.: Computer aided design of developable surfaces, Marine Technology, 8, 1971, 233-242.

[23] Pottman, H.; Wallner, J.: Computational Line Geometry, NJ: Springer-Verlag, 2001.

[24] Pottmann, H.; Wallner J.: Approximation algorithms for developable surfaces, Computer Aided Geometric Design 1999;16(6): 539-556, DOI: 10.1016/S0167-8396(99)00012-6.

[25] Tang, K.; Chen, M.: Quasi-Developable Mesh Surface Interpolation via Mesh Deformation, IEEE Transaction on Visualization and Computer Graphics, 15(3),2009, 518-528. DOI: 10.1109/TVCG.2008.192.

[26] Tang, K.; Wang, C.C.L.: Modeling developable folds on a strip, ASME Journal of Computing and Information Science in Engineering, 5(1), 2005, 35-47, DOI: 10.1115/1.1804206.

[27] Wang, C.C.L.; Tang, K.: Achieving Developability of a Polygonal Surface by Minimum Deformation: A Study of Global and Local Optimization Approaches, The Visual Computer, 20(8-9),2004, 521-539, DOI: 10.1007/s00371-004-0256-0.

[28] Wang, C.C.L.; Tang, K.: Developable triangulation of a strip, Computer Aided Design & Applications, 2(1-4), 2005, 233-242.

[29] Wang, C.C.L.; Tang, K.: Optimal boundary triangulations of an interpolating ruled surface, ASME Journal of Computing and Information Science in Engineering, 5(4), 2005, 291-301. DOI: 10.1115/1.2052850.