Stripification of Free-Form Surfaces With Global Error Bounds for Developable Approximation

Yong-Jin Liu, Yu-Kun Lai, and Shimin Hu

***Abstract*—Developable surfaces have many desired properties in the manufacturing process. Since most existing CAD systems utilize tensor-product parametric surfaces including B-splines as design primitives, there is a great demand in industry to convert a general free-form parametric surface within a prescribed global error bound into developable patches. In this paper, we propose a practical and efficient solution to approximate a rectangular parametric surface with a small set of C^0-joint developable strips. The key contribution of the proposed algorithm is that, several optimization problems are elegantly solved in a sequence that offers a controllable global error bound on the developable surface approximation. Experimental results are presented to demonstrate the effectiveness and stability of the proposed algorithm.**

***Note to Practitioners*—This paper was motivated by a joint industrial project which uses CATIA V5R16 as the design platform. The shape of product was modeled by free-form parametric patches including NURBS in the CATIA system. For efficient manufacturing, the parametric surfaces are required to convert into developable patches with controllable global error bounds. Given a small tolerance, if this cannot be achieved, then cut the surface into pieces, each of which is developable. However, the CATIA system does not provide such a functionality. With the development of this project, we design an efficient algorithm which is presented in the paper to achieve this goal. We also build the algorithm into a plugin module in CATIA using CAA V5.**

***Index Terms*—Developable surface approximation, free-form parametric surfaces, geometric optimization, triangle strip.**

I. Introduction

DEVELOPABLE surfaces, which can be unfolded into plane without stretch, are widely used in engineering. While developable surfaces can be directly used to model some simple shapes such as cones and cylinders, most existing CAD/CAM systems use general parametric surfaces including B-splines as design primitives to model complicated free-form shapes. Therefore, there is a great demand in industries to convert a general parametric surface within a prescribed global error bound into a small set of developable pieces. These pieces are afterwards cut from planar material, bent back without stretch, moved into their final positions and stitched together to form the final product.

If sufficient differentiability is assumed, developable surfaces can only be part of plane, cone, cylinder, tangent surface of a curve or a composition of them. Surface approximation using conical and cylindrical patches are studied in [10], [14], [22], and [25]. Tangent surfaces of B-spline curves in dual space based on projective geometry are studied in [1], [2], [9], [21], and [23]. Cone spline surfaces that can be used as transition of smooth joining developables are studied in [12]. If the free-form surface is nearly developable, a spherical curve segmentation and fitting technique is presented in [3] that approximates such a surface by a G^1 developable surface. If the surface is far from developables, Elber [5] proposes an approximation method that trims the surface using isolines and interpolates each trimmed piece by a ruled patch. This method is extended by Subag and Elber [26] in which a semi-automatic algorithm is proposed.

Recent advances in both CAD and production automation have revealed that the developable surfaces can be in effect approximated by triangle strips [5], [6], [7], [13], [17], [27]. Two closely related topics in CAD and computer graphics are triangulation of free-form surfaces [18]–[20] and stripification of mesh models [8], [11], [16], [29]. Triangulation of a free-form surface usually satisfies a global error bound; but the resulting triangulation cannot be used for developability. The strips from graphics model stripification are mainly used for fast graphical rendering since each triangle in the strip (except for the first one) can be encoded by an integer in OpenGL; however, if these strips are directly developed into plane, the shape may be self-intersected and very winded with arbitrary width everywhere. The strips are more desired for developability if they have almost uniform width after developing into plane. Besides triangles, strips of planar quadrilaterals are also good candidates for developable approximation [10], [28].

In this paper, a practical and efficient algorithm is proposed to achieve developable strip approximation by trimming a rectangular parametric surface into a small set of strips; each strip consists of a chain of triangles that have almost uniform width after development. The novelty of the presented algorithm is that we introduce a controllable global error bound into the trimming process. In the method, geodesics are used as the basic primitive to trim the surface. An application scenario is presented in Section VI, showing several advantages that could be achieved by geodesic cutting.

II. Algorithm Overview

The basic idea of the presented algorithm is simple. Given a rectangular parametric surface (e.g., a tensor-product B-Spline surface) with boundaries e_1, e_2, e_3, e_4 in counter clockwise

Manuscript received December 17, 2007; revised March 16, 2008; June 20, 2008; September 11, 2008; and November 04, 2008. First published April 17, 2009; current version published September 30, 2009. This paper was recommended for publication by Associate Editor K. Saitou and Editor M. Wang upon evaluation of the reviewers' comments. This work was supported by the Natural Science Foundation of China under Projects 60736019, 60603085, and 90718035, by the 863 Program of China under Project 2007AA01Z336, and by the 973 Program of China under Project 2006CB303102.

The authors are with the Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Tsinghua University, Beijing, China (e-mail: liuyongjin@tsinghua.edu.cn; laiyk03@tsinghua.edu.cn; shimin@tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2008.2009926

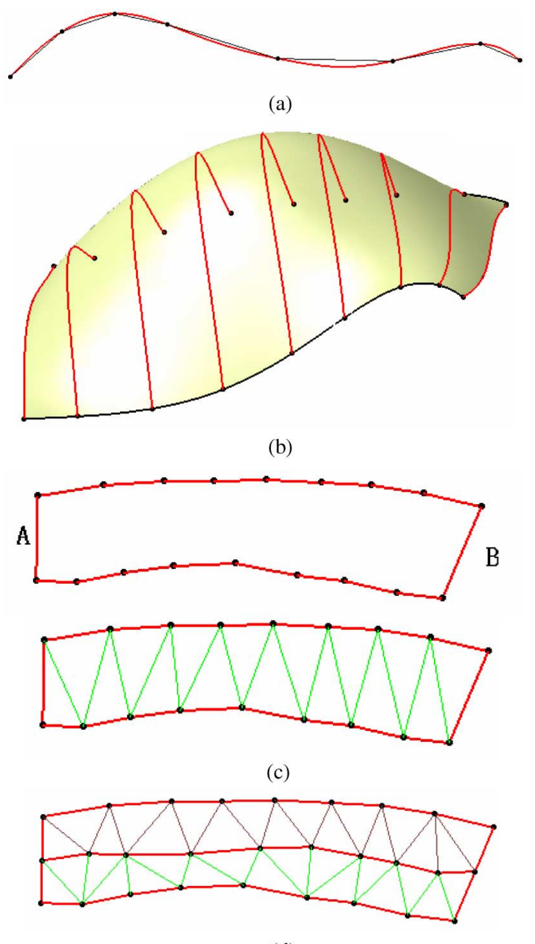


Fig. 1. Overview of the algorithm. (a) Optimal polyline approximation of parametric curve, (b) trim the parametric surface by geodesics which connect corresponding vertices in opposite boundary polylines, (c) optimally discretize the geodesics and construct an initial triangle approximation for each strip, and (d) if the approximate error is larger than a prescribed tolerance, split the strip into two and repeat the process.

order, from two pairs of opposite edges (e_1, e_3) and (e_2, e_4), an optimal pair is identified. Two strategies are presented in Section III-D for this identification. Without loss of generality, assume the optimal pair is (e_1, e_3). The curves e_1 and e_3 are discretized by polylines with a prescribed tolerance: the number of vertices on each polyline is maintained to be the same. Then, the corresponding vertices between two polylines are connected by geodesics on the surface. These geodesics trim the surface into pieces which afterwards are optimally approximated by developable triangle strips. Refer to Fig. 1. The overall algorithm, called **DevAppr**, is summarized below.

1) Section III: Optimally discretize curves (e_1, e_3) into two polylines (p_1, p_3) with the same number n of vertices; the discretization error satisfies $\max\{\|e_1-p_1\|, \|e_3-p_3\|\} < \varepsilon$, where ε is a prescribed tolerance [ref. Fig. 1(a)].
2) Section IV: Construct the geodesics by connecting the corresponding vertices $(v_i^1, v_i^3), i = 2, \dots, n-1$, in the polylines $p_1 = \{v_2^1, v_3^1, \dots, v_{n-1}^1\}$ and $p_3 = \{v_2^3, v_3^3, \dots, v_{n-1}^3\}$. These geodesic curves partition the parametric surface into strips $\{S_1, S_2, \dots, S_{n-1}\}$ [ref. Fig. 1(b)].
3) Discretize curves e_2, e_4 and each geodesic g_i into polylines p_2, p_4 and l_i, with the minimum number of vertices, respectively, such that $\|e_j - p_j\| < \varepsilon, j = 2, 4$ and $\|g_i - l_i\| < \varepsilon$ [ref. Fig. 1(c)].
4) Section V: For every pair (l_i, l_{i+1}) of adjacent polylines in the set $\{l_1 = p_2, l_2, \dots, l_{n-1}, l_n = p_4\}$, construct an optimal strip of triangles T_i that minimizes an elaborately designed energy functional [ref. Fig. 1(c)].
5) Given a trimmed strip S_i and its associated counterpart of triangles T_i, if the error $\|S_i - T_i\| > \varepsilon$, then subdivide the strip S_i into two and repeat the process [ref. Fig. 1(d)].
6) Develop all triangle strips into the same plane.

III. Polygonal Approximation of Curves

The steps 1 and 3 in the Algorithm **DevAppr** require the solutions to the following two optimization problems:

1) **Min-# problem**: given a parametric curve c, approximate it by a polyline p with the minimum number of segments, such that the approximation error does not exceed a given tolerance.
2) **Min-ε problem**: given a parametric curve c, approximate it by a polyline p with a given number of line segments n, such that the approximation error is minimized.

Given the solutions to the above two problems, the step 1 in Algorithm **DevAppr** is performed first by optimally discretizing e_1 and e_3 with a given tolerance ε; that solves a Min-# problem. Denote the resulting numbers of vertices on p_1 and p_3 by n_1 and n_3, respectively. Without loss of generality, let $n_1 \le n_3$. To maintain the same vertex number on p_1 and p_3, we need to resample e_1 with a given number of line segments n_3, that solves a Min-ε problem. The solution to Min-# problem is also used in step 3.

In the following, unless otherwise specified, all the curves are parameterized by the arc-length.

A. Solution to Min-# Problem

To find the minimum number of samples on a parametric curve c such that the resulting polyline has error no larger than a prescribed tolerance ε, the solution is as follows. Refer to Fig. 2(a). We start from one end of the curve and greedily add samples one by one, till we reach the other end of the curve. To add a new sample, we put a cylindrical surface with radius $\varepsilon/2$ centered at the current position, and oriented along with the tangent direction of the curve at the position. The nearest intersection point of the curve with the cylindrical surface is considered as the next sample.

The above greedy approach works almost fine. But the samples near the approaching end are frequently observed to be problematic: the last sample may (and usually appears to) be very close to the end point [see the second row of Fig. 2(a)]. To remedy this situation, starting from the last sample, we apply again the greedy method in the opposite direction with decreasing order of parameter.

Denote the polyline obtained by the first round of greedy approach by p with vertices $(v_1, v_2, \dots, v_n)$. Let u_i represents the parameter of v_i, then $u_1 < u_2 < \dots < u_n$. v_{n-1} may be very close to v_n and so is u_{n-1} to u_n. In the second round of

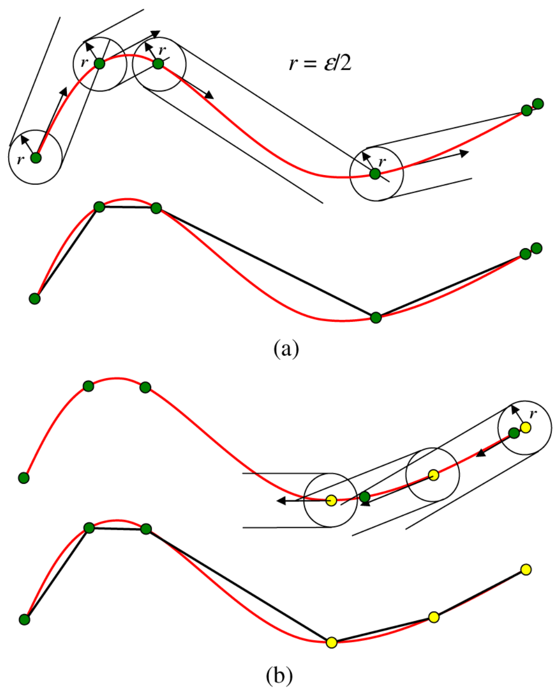


Fig. 2. Polyline approximation of a curve with a prescribed error ε. (a) Greedy polyline approximation from left end and (b) the fine tuning approach to (a).

greedy approach, starting from v_n we compute v'_{n-1} with decreasing parameters. It must be satisfied that $u'_{n-1} \leq u_{n-1}$. The new sample is then defined as v''_{n-1} with the parameter $u''_{n-1} = (u'_{n-1} + u_{n-1})/2$. Since arc-length parameterization is used, v''_{n-1} sits at the midway between v'_{n-1} and v_{n-1} on curve. It can be readily verified that v''_{n-1} satisfies the error bounds from both sides. Starting at v''_{n-1}, the fine tuning algorithm is as follows:

```
1. while $i > 0$
1.1 apply the greedy approach to $v''_i$ in the reverse direction to
    find $v'_{i-1}$ with parameter $u'_{i-1}$;
1.2 if $u'_{i-1} < u_{i-1}$
1.2.1 $u''_{i-1} = \max\{(u'_{i-1} + u_{i-1})/2, 0\}$;
1.3 else break;
1.4 $i--$;
```

The above fine tuning approach has a few advantages. First, it guarantees that the same number of samples are resulted in and the same error bound still applies. Second, the distribution of samples becomes much more uniform, as shown in the second row of Fig. 2(b).

B. Solution to Min-ε Problem

Given a fixed number n of sample points, to find the minimum error of polygonal approximation p of a parametric curve c, the objective is to optimize a $n-2$ vector $U = (u_2, \ldots, u_{n-1})^T$ in the optimization function defined by

$$\mathrm{Opt}(U) = \sum_{i=1}^{n-1} \int_{u_i}^{u_{i+1}} \left\| c(u) - \frac{(u_{i+1} - u)c(u_i) + (u - u_i)c(u_{i+1})}{u_{i+1} - u_i} \right\|^2 du \quad (1)$$

where $c(u_i), i = 1, \ldots, n$ are n optimal positions of samples with $u_1 = 0, u_n = \mathrm{length}(c)$. Minimization of function $\mathrm{Opt}(U)$ is a typical multidimensional optimization problem which can be solved by the classical gradient or simplex methods. Note that given

$$\begin{cases} f(t) = \int_t^b h(t,b,u)du \\ g(t) = \int_a^t h(a,t,u)du \end{cases}$$

we have

$$\begin{cases} \frac{df(t)}{dt} = \int_t^b \frac{\partial h(t,b,u))}{\partial t} du - h(t,b,t) \\ \frac{dg(t)}{dt} = \int_a^t \frac{\partial h(a,t,u)}{\partial t} du + h(a,t,t) \end{cases}.$$

So given any $n-2$ dimensional point $\mathbf{x}$, we can not only evaluate $\mathrm{Opt}(\mathbf{x})$, but also easily evaluate $\nabla\mathrm{Opt}(\mathbf{x})$. To minimize objective $\mathrm{Opt}(\mathbf{x})$, we use the conjugate gradient method in multidimensions [24]; this method requires only of order a few times n storage and converges quickly in practice.

C. Finding Optimal Boundary Pair

The optimal pair determined from the boundary curves e_1, e_2, e_3, e_4 of the parametric surface is used to locate the endpoints of geodesics for trimming the surface. Two strategies are used in Algorithm **DevAppr** to find the optimal pair. The first strategy is fully automatic. Two pairs (e_1, e_3) and (e_2, e_4) are, respectively, approximated by polylines (p_1, p_3) and (p_2, p_4) with a prescribed tolerance ε. p_1 and p_3 have the same minimum number n of samples. p_2 and p_4 has the same minimum number m of samples. If $m < n$, then (e_2, e_4) is the optimal pair; otherwise it is (e_1, e_3).

The first strategy is optimal in the geometric sense. However, in many CAD models, different boundary pairs have different functionalities. To take the semantics of physical functionalities into account, we set the second strategy that allows the user to interactively select the optimal pair.

IV. Compute Geodesics on Parametric Surfaces

Given two points $\mathcal{P}, \mathcal{Q}$ on a surface, we use an adaptive solution based on the Maekawa's method [15] to compute the geodesic passing $\mathcal{P}$ and $\mathcal{Q}$.

Denote the parametric surface by $S(u,v)$. Let the prescribed source points $\mathcal{P}, \mathcal{Q}$ be specified by parameters $(u_p, v_p), (u_q, v_q)$ and any curve on the surface connecting them be specified by function $(u(s), v(s))$. The geodesic differential equations [4] are

$$\begin{cases} \frac{d^2(u)}{ds^2} + \Gamma_{11}^1 \left(\frac{du}{ds}\right)^2 + 2\Gamma_{12}^1 \frac{du}{ds}\frac{dv}{ds} + \Gamma_{22}^1 \left(\frac{dv}{ds}\right)^2 = 0 \\ \frac{d^2(v)}{ds^2} + \Gamma_{11}^2 \left(\frac{du}{ds}\right)^2 + 2\Gamma_{12}^2 \frac{du}{ds}\frac{dv}{ds} + \Gamma_{22}^2 \left(\frac{dv}{ds}\right)^2 = 0 \end{cases} \quad (2)$$

where Γ_{ij}^k is the Christoffel symbol. Let $\mathbf{y} = (u, v, p, q)^T$, $\mathbf{g} = (p, q, -\Gamma_{11}^1 p^2 - 2\Gamma_{12}^1 pq - \Gamma_{22}^1 q^2, -\Gamma_{11}^2 p^2 - 2\Gamma_{12}^2 pq - \Gamma_{22}^2 q^2)^T$. Then, the second-order (2) can be reduced to a system of first-order differential equations (ODEs)

$$\frac{d\mathbf{y}}{ds} = \mathbf{g}(s, \mathbf{y}). \tag{3}$$

Relaxation method [24] is used for numerical solution. First, the parameter domain (u, v) is discretized into a set of mesh points (u_i, v_i). The system of ODEs (3) is then replaced by a system of finite difference equations (FDEs)

$$\mathbf{y}_k - \mathbf{y}_{k-1} - \frac{1}{2}(s_k - s_{k-1})(\mathbf{g}_k + \mathbf{g}_{k-1}) = 0, \quad k = 1, 2, \ldots \tag{4}$$

A line joining $\mathcal{P}$ and $\mathcal{Q}$ on parameter domain is employed as the initial guess of the solution. If the surface is near developable, this guess is pretty good. The initial guess usually only satisfies the required boundary conditions. Then, the iteration process, called *relaxation*, is invoked to adjust the values on the grid.

Let the line $\overrightarrow{\mathcal{P}\mathcal{Q}}$ be sampled by m mesh points $u_i = u_{i-1} + i\Delta u_i, v_i = v_{i-1} + i\Delta v_i$. The boundary condition of FDEs (4) is

$$(u_0, v_0) = (u_p, v_p), \quad (u_{m-1}, v_{m-1}) = (u_q, v_q). \tag{5}$$

Let $x(s)$ be a monotonically increasing function of s that satisfies $x = i$ at the mesh points $(u_i, v_i), i = 0, 1, \ldots, m-1$. Between any two successive points, $\Delta x = 1$. Since dx/ds is proportional to the density of mesh points, we define a density function as

$$\phi(s) = c\frac{dx}{ds}$$

where $\phi > 0$ and c is an overall scale constant. Given two mesh points $(u_k, v_k), (u_{k-1}, v_{k-1})$, we use the length of vector $S(u_k, v_k) - S(u_{k-1}, v_{k-1})$ to approximate $(s_k - s_{k-1})$ in FDEs (4). For a better approximation of arc length, we want to put more points in the place of high curvature and less in planar regions. So, ϕ is chosen to be the curvature function

$$\phi(s) = \frac{u'(s)v''(s) - u''(s)v'(s)}{\sqrt{(u'^2 + v'^2)^3}} = \frac{p\frac{dq}{ds} - \frac{dp}{ds}q}{\sqrt{(p^2 + q^2)^3}}.$$

The Newton–Raphson method [24] is used to solve a system of nonlinear equations, which converges quadratically when the initial guess is near the root. Examples of geodesic computation on different B-spline surfaces, using different parametric line segments as the initial guesses, are illustrated in Fig. 3.

A. Handle Non-Geodesic Boundaries

Consider a trimming geodesic g close to the boundary $r(v) = S(0, v)$. If r itself is a geodesic of S, then g and r will not cross each other. In the Appendix, it is shown that for r is a geodesic, the necessary and sufficient condition is that $(\partial S(u, v))/(\partial u)|_{u=0}$ lies in the rectifying plane of $r(v)$.

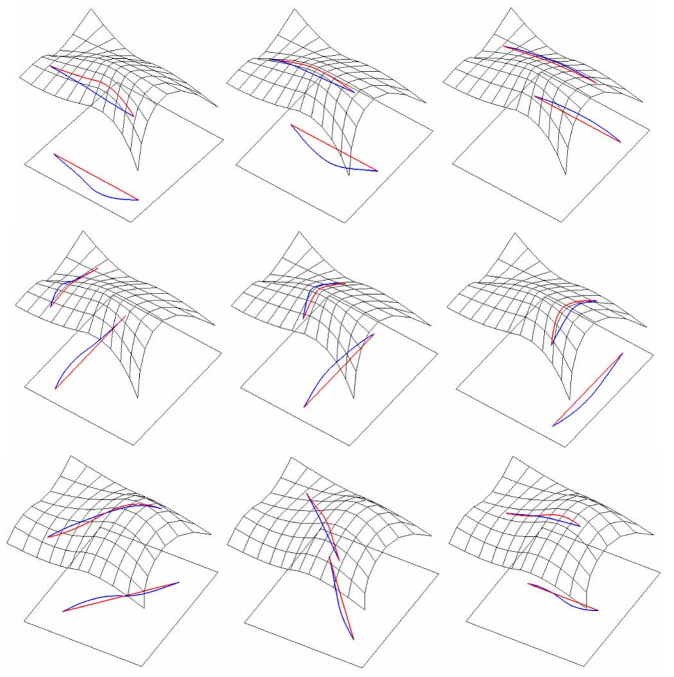


Fig. 3. Test examples on different B-spline surfaces with different source points $(\mathcal{P}, \mathcal{Q})$: the initial guess is shown in red and the solution is in blue. Both parametric domain in $\mathbb{R}^2$ and image surface in $\mathbb{R}^3$ are given in the figure.

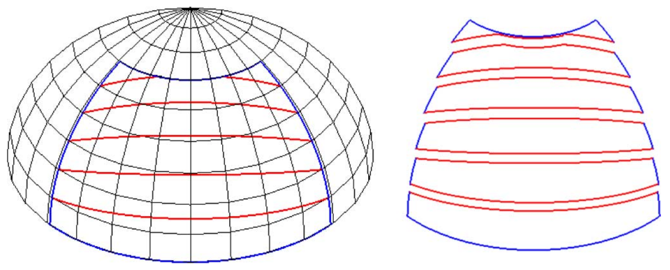


Fig. 4. Case that trimming geodesic intersects a boundary of a sperhical region (surrounding in blue color).

If Bezier or B-spline surface is used, a convenient way to construct a surface with geodesic boundary is as follows. Let r be specified by the first row of control points $P_{0,i}, i = 1, 2, \ldots$, which lie in a plane b. Construct the second row of control points such that $P_{1,i} - P_{0,i}, i = 1, 2, \ldots$, are all perpendicular to b.

If r is not a geodesic, it may intersect the closest trimming geodesic. An extreme case is illustrated in Fig. 4. If this case happens, we disturb the intersection segment a bit and symbolically separate the strip.

B. A Computational Issue

We use geodesic paths to trim the surface into pieces. Although the samples on the opposite boundary curves e_1, e_3 are ordered in the same direction, two adjacent geodesic paths may still touch each other in some extreme cases. In industrial design, the surfaces approximated by developable patches are usually close to developable. In this case, our method works pretty well. If touching is detected, we start a desperate mode in our algorithm: similar to the case shown in Fig. 4, the two

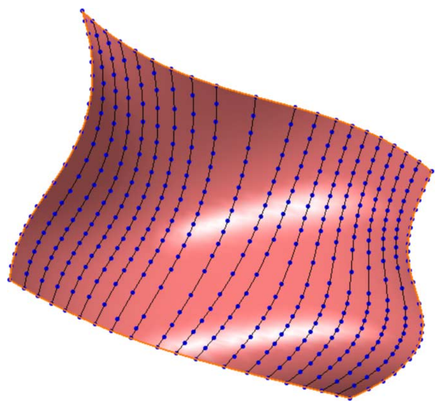


Fig. 5. Surface trimming by geodesic paths.

segments between the touching points are replaced by the one whose length is shorter.

V. Developable Strip Generation

Refer to Fig. 5. The computed shortest paths trim the parametric surface into strips $(S_1, S_2, \ldots, S_{n-1})$. Each strip S_i is bounded by two parametric curves c_i, c_{i+1} (either geodesics or boundary curves e_2, e_4) and two line segments. Any ruled surface interpolating c_i, c_{i+1} can be characterized by

$$S_i(u,v) = (1-v)c_i(u) + vc_{i+1}(f(u)) \tag{6}$$

where $f(u)$ is a monotone function mapping from $[0, \mathrm{length}(c_i)]$ to $[0, \mathrm{length}(c_{i+1})]$. If at any ruling the tangent plane to the surface is the same, then the surface is developable. However, finding such an exact solution to the mapping $f(u)$ is an extremely difficult problem. Inspired by the recent work [6], [27], our practical solution is presented as follows.

A. Initial Strip Generation

The trimming geodesics, as well as boundary curves e_2, e_4, are approximated by polylines with a given tolerance ε. Consider the strip S_i bounded by curves c_i and c_{i+1}. Denote the two approximate polylines of c_i and c_{i+1} by $P_i = \{p_1, \ldots, p_n\}$ and $Q_i = \{q_1, \ldots, q_m\}$. The *triangle strip* T_i is defined to be a constrained triangulation which satisfies the following conditions.

- T_i interpolates polylines P_i, Q_i and line segments p_1q_1, p_nq_m [refer to the first row of Fig. 1(c)].
- All the vertices of T_i belong to the set of $(p_1, \ldots, p_n, q_1, \ldots, q_m)$.

If any edge in T_i connecting a p_i and a q_j, it is called a *bridge edge*. Similar to [27], to build an objective function for optimization, either the minimal length of the sum of bridge edges (refer to as *MinDist*) or the minimal bending energy on all bridge edges (refer to as *MinBend*) can be used. The length of a bridge p_iq_j is $\mathrm{Dist}(p_iq_j) = \|p_i - q_j\|$. The bending energy related to p_iq_j is calculated by the dihedral angle between two triangles adjacent to p_iq_j; here we denote it by $\mathrm{Bend}_{xy}(p_i, q_j)$, where x denotes the sample which forms the preceding triangle with p_i, q_j and y forms the succeeding triangle, e.g., $\mathrm{Bend}_{pq}(p_i, q_j)$ means the two adjacent triangles being (p_{i-1}, p_i, q_j) and (p_i, q_j, q_{j+1}).

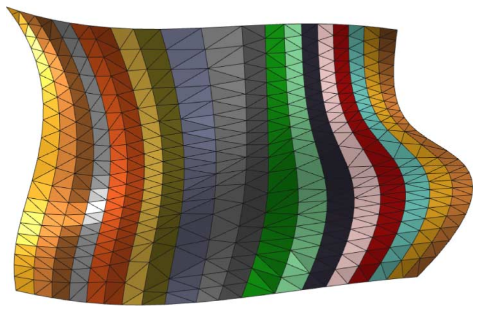


Fig. 6. Triangulation with the minimal length criterion: both the greedy approach [27] and the proposed dynamic programming approach have the same result in this case.

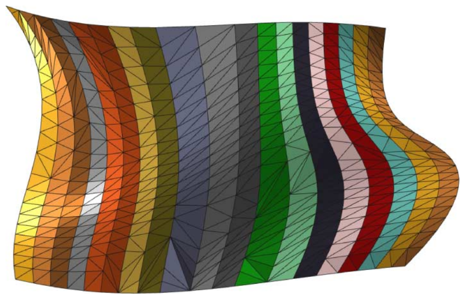


Fig. 7. Triangulation with the minimal bending energy criterion using the greedy approach [27].

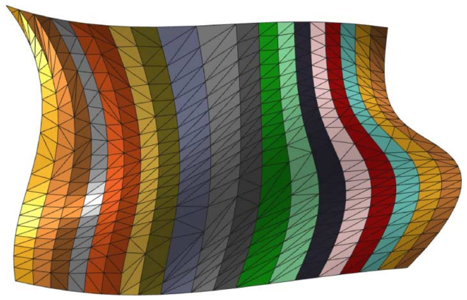


Fig. 8. Triangulation with the minimal bending energy criterion using the proposed dynamic programming approach.

A greedy approach is proposed in [27] to find a locally optimal solution to both the minimal length and the minimal bending energy problems. Contrasting with this local optimal solution, in the Algorithm **DevAppr**, we propose a dynamic-programming-based approach that guarantees output a globally optimal solution with the same constraints as in [27]. We test both greedy and dynamic programming approaches on many examples: the minimal length criterion usually leads to close or even the same results by these two approaches (as shown in Fig. 6); while the greedy approach with minimal bending energy, however, produces suboptimal results much worse than the dynamic programming approach (compare Figs. 7 and 8). This is clearly revealed in the analytical data presented in Fig. 9 and Table I.

We use the following two strategies in the dynamic programming approach.

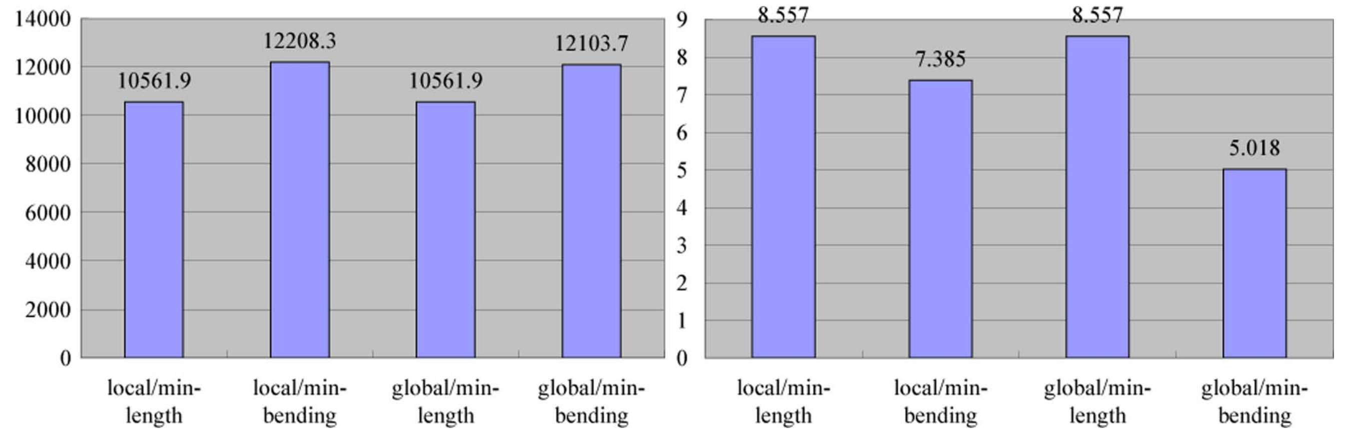


Fig. 9. Analytical data of the minimal length and bending energy of both local and global approach in the model as shown in Figs. 6–8 and in another model (right).

TABLE I
ANALYTICAL DATA OF EXPERIMENT IN FIG. 10

method	length	bend. engy	L_∞ error	L_2 error	time
lenth/local	240.333	39.8183	3.1940	0.5839	28ms
lenth/global	233.612	38.9149	1.9017	0.3844	29ms
bend/local	414.264	30.2529	4.0808	0.8843	40ms
bend/global	256.737	4.8431	3.1940	0.6660	42ms

Minimal distance strategy. Assume that $\mathrm{MinDist}(p_i, q_j)$ is the minimal length of total bridge edges from $p_1, \ldots, p_i$ and $q_1, \ldots, q_j$. It is immediately seen that

$$\mathrm{MinDist}(p_i, q_j) = \min\{\mathrm{MinDist}(p_{i-1}, q_j), \mathrm{MinDist}(p_i, q_{j-1})\} + \mathrm{Dist}(p_i, q_j)$$

with the boundary conditions

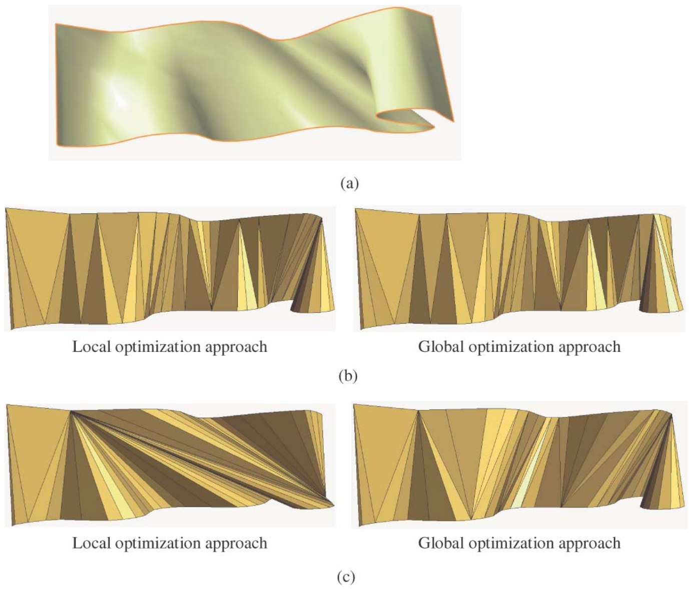
$$\begin{cases} \mathrm{MinDist}(p_1, q_1) = \|p_1 - q_1\| \\ \mathrm{MinDist}(p_i, q_1) = \mathrm{MinDist}(p_{i-1}, q_1) \\ \quad +\mathrm{Dist}(p_i, q_1), \quad 1 < i \le n \\ \mathrm{MinDist}(p_1, q_j) = \mathrm{MinDist}(p_1, q_{j-1}) \\ \quad +\mathrm{Dist}(p_1, q_j), \quad 1 < j \le m \end{cases}.$$

The global minimum solution is then derived from $\mathrm{MinDist}(p_n, q_m)$ with back tracing.

Minimal bending energy strategy. Assume that $\mathrm{MinBend}_x(p_i, q_j)$ is the minimal bending energy for sequences $p_1, \ldots, p_i$ and $q_1, \ldots, q_j$, with the last triangle having two samples on x side ($x = p$ or q). The rule to compute a particular MinBend is given as

$$\begin{aligned} &\mathrm{MinBend}_p(p_i, q_j) \\ &\quad = \min\{\mathrm{MinBend}_p(p_{i-1}, q_j) + \mathrm{Bend}_{pp}(p_{i-1}, q_j), \\ &\qquad \mathrm{MinBend}_q(p_{i-1}, q_j) + \mathrm{Bend}_{qp}(p_{i-1}, q_j)\} \\ &\mathrm{MinBend}_q(p_i, q_j) \\ &\quad = \min\{\mathrm{MinBend}_p(p_i, q_{j-1}) + \mathrm{Bend}_{pq}(p_i, q_{j-1}), \\ &\qquad \mathrm{MinBend}_q(p_i, q_{j-1}) + \mathrm{Bend}_{qq}(p_i, q_{j-1})\}. \end{aligned}$$

The global minimum solution is then derived from $\min\{\mathrm{MinBend}_p(p_n, q_m), \mathrm{MinBend}_q(p_n, q_m)\}$ with back tracing.



(a)
Local optimization approach
Global optimization approach
(b)
Local optimization approach
Global optimization approach
(c)

Fig. 10. Comparison of our global optimization approach with the local optimization in [27]. (a) The input surface patch, (b) optimization using the minimal length criterion, and (c) optimization using the minimal bending energy criterion.

The global optimality of the above formulae can be readily verified by induction. The improvement of our global optimization over the local optimization is further demonstrated in Fig. 10 with the analytical data shown in Table I. Experimental results show that the proposed global optimization approach works more robustly on the general data sets than our implementation of the local greedy approach in [27]. Our experiments were carried out on a Core2Duo 2 GHz laptop computer with 2 GB memory, using CATIA as the supporting platform. Timings are almost similar for both methods. This shows that dynamic programming optimization takes almost negligible time for the overall computation, especially considering necessary interactions with CATIA interfaces.

B. Error-Driven Strip Refinement

Quite different from the motivations in [6] and [27], our generated triangle strip T_i needs to further satisfy a prescribed toler-

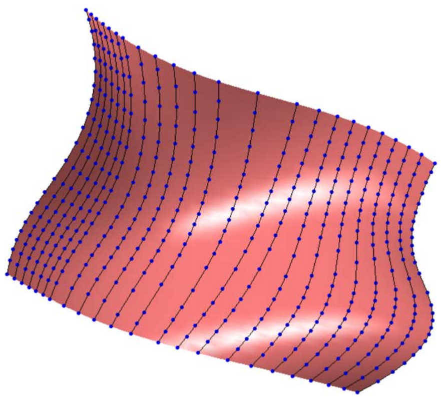


Fig. 11. Strip refinement: curve sampling (compare to Fig. 5).

TABLE II
COMPARISON OF THE VALUE OF INTEGRAL OF ABSOLUTE GAUSSIAN CURVATURE BETWEEN THE ORIGINAL SURFACE AND ITS RULED APPROXIMATION

Models	ruled approx.	original surf.	decreasing ratio
Fig. 8	1.5474	3.1623	51.1%
Fig. 13	0.2538	3.0639	91.7%

ance ε when compared to the original strip S_i. In the Algorithm **DevAppr**, adaptive triangle strip refinement is designed as follows to produce output triangle strips within the prescribed error bound ε.

Since the geodesics and boundary curves are sampled by solving Min-# problems, it guarantees that the approximation error between the polylines and their corresponding curves on the surface is bounded by ε. Then, it suffices to consider the interior of each strip. Given the initial triangulation, for a triangle t in a strip T_i, assume its three vertices are v_1, v_2 and v_3, with the parameters $\mathbf{x}_1, \mathbf{x}_2$, $\mathbf{x}_3$ respectively. For any point $t(\lambda_1, \lambda_2)$ in the triangle with barycentric coordinate $(\lambda_1, \lambda_2, 1-\lambda_1-\lambda_2), 0 \le \lambda_1, \lambda_2, \lambda_1+\lambda_2 \le 1$, its corresponding point $s(\lambda_1, \lambda_2)$ on the surface is the one with parameter $\lambda_1 \cdot \mathbf{x}_1 + \lambda_2 \cdot \mathbf{x}_2 + (1-\lambda_1-\lambda_2) \cdot \mathbf{x}_3$. The L_p error between triangle t and its image s on the surface is defined as

$$E_p(t,s) := \sqrt[p]{\frac{1}{A_t} \iint_{t(u,v)} \|s(u,v) - t(u,v)\|^p du dv}$$

and the L_p error between the triangle strip T_i and the parametric strip S_i is defined as

$$E_p(T_i, S_i) := \sqrt[p]{\frac{1}{A_{T_i}} \sum_{t \in T_i} E_p^p(t,s) A_t}$$

where A_t and A_{T_i} are the area of the triangle t and T_i, respectively. In particular, we are interested in L_∞ error

$$E_\infty(T,S) = \max_{t \in T} \max_{(\lambda_1,\lambda_2)} |t(\lambda_1, \lambda_2) - s(u(\lambda_1,\lambda_2), v(\lambda_1,\lambda_2))|.$$

The overall L_∞ error is computed for the initial triangulation. If it is above ε, the strip with the largest L_∞ error is picked out and subdivided by a new added trimming line connecting two points located at the midpoints of two boundary line segments [see Fig. 1(d)]. The two subdivided strips are then triangulated and checked again for the approximation error. The process repeats until the error is within the given tolerance ε. We select the trimming line to be the average of two boundary geodesics in the parametric domain so that the error is guaranteed to decrease.

For the data set shown in Figs. 5–8, the cylinder radius r used for point sampling is set to be 5 mm, and the prescribed approximation error in Fig. 8 is $\varepsilon = 10$ mm. The size of the illustrated shape is about 2000 mm in its largest direction. The resulting triangulation has the L_∞ error $E_\infty(T,S) = 21.85$ mm. After three iterations of strip refinement, $E_\infty(T,S)$ becomes 7.42 mm and the algorithm terminates. The curve sampling and retriangulation after refinement are illustrated in Figs. 5–8, respectively.

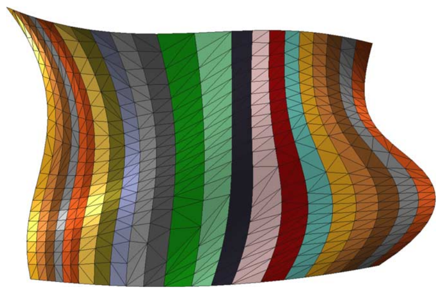


Fig. 12. Strip refinement: retriangulation (compare to Fig. 8).

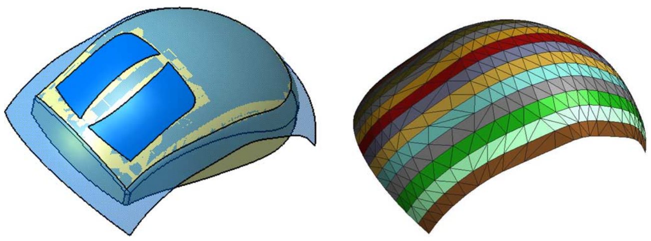


Fig. 13. Stripification of the upper surface of a mouse model. The surface is afterwards trimmed by surface intersection.

C. Triangle Strip Flattening

Given the triangle strips, flattening them is straightforward. For each strip, the first triangle is flattened at some place in the $X-Y$ plane. Adjacent triangles are then flattened one by one along the bridge edges, as long as the flattened triangles do not overlap each other in the plane. If overlapping occurs, we start a new flattening at the first overlapping triangle and continue the process. Several flatten examples are shown in Fig. 14.

VI. Experimental Results

Increasing the Developability: Our method trims a nondevelopable surface into strips. Each strip is approximated by a ruled patch [see (6)] which is further discretized into a developable triangle strip. It is expected that the ruled surface approximation could increase the developability of the surface and thus the global error controlled triangulation could be a good developable approximation. This expectation is demonstrated by our experiments. Table II summarizes the comparison of the value of integral of the absolute Gaussian curvature over the original surface and its ruled surface approximations. The triangle stripification also shows a good behavior on surface developability.

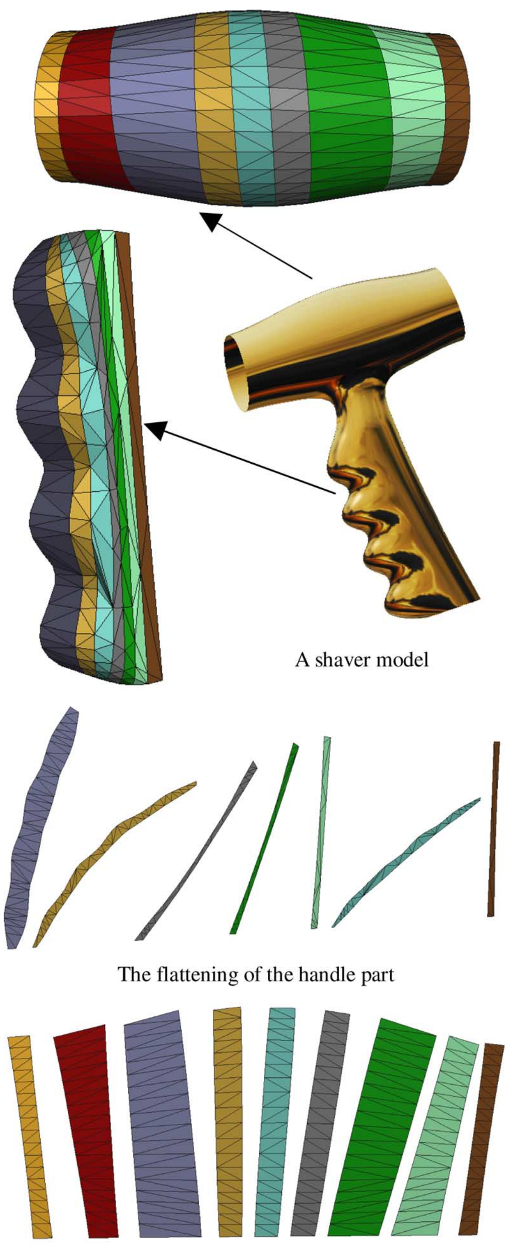

A shaver model
The flattening of the handle part
The flattening of the top part

Fig. 14. Stripification of two component surfaces of a shaver product: the top part and handle part are stripified and flattened with error 0.65 and 3.91 mm, respectively.

TABLE III
PERFORMANCE DATA OF EXPERIMENTS WITH DIFFERENT ERROR BOUNDS

model	input error ε	final error	#. strips	#. triangles
Fig. 13	$10mm$	$7.06mm$	17	751
Fig. 14(top)	$1mm$	$0.65mm$	9	314
Fig. 14(handle)	$4mm$	$3.91mm$	7	227

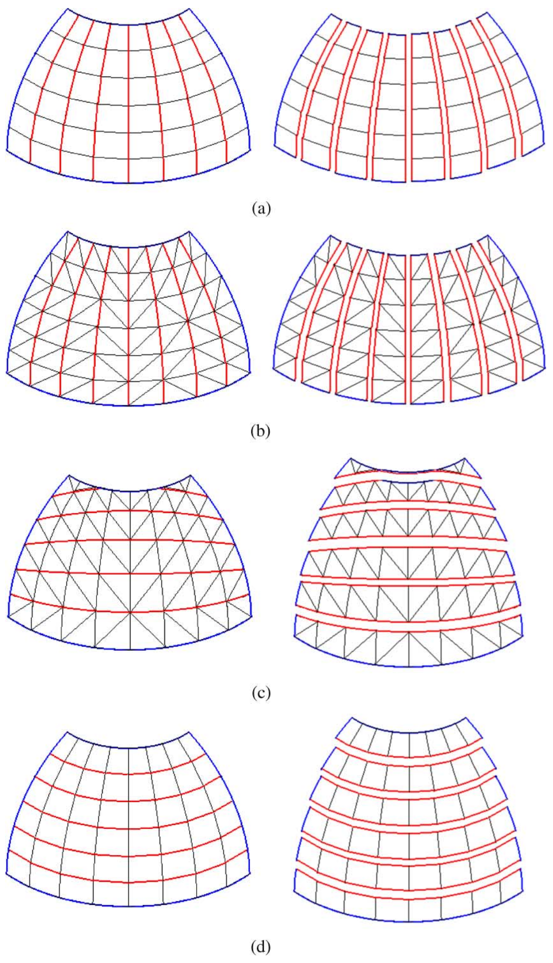


Fig. 15. Given a piece of surface of revolution (see the blue surrounding area in Fig. 4), the comparison of our method with the Hoschek's method [10] and the Elber's method [5]. (a) Developable approximation using Hoschek's method (1998). (b) Our method using upper and lower boundaries as the optimal pair. (c) Our method using left and right boundaries as the optimal pair. (d) Developable approximation using Elber's method (1995).

Industrial Experiments: Two typical industrial models are presented and performance data is summarized. The first example is the upper surface of a mouse model. The surface is about 105 mm $\times$ 46 mm. The stripification result with error 7.06 mm is shown in Fig. 13. All performance data as well as other models is summarized in Table III.

The second example is a shaver model. Its top part has size of 220.5 mm $\times$ 148.4 mm. Stripifications and flattening of the top part, with error 0.65 mm, are presented in the first and last rows in Fig. 14. The handle part of the shaver model has size of 43.9 mm $\times$ 200.9 mm. Its stripifications and flattening, with error 3.91 mm, are shown in the mid of Fig. 14.

Consideration of Geometric Continuity: Our method generates a C^0-continuous developable stripification to an input rectangular free-form surface. Experiments show that the generated strips have almost uniform widths in their planar versions (see Figs. 12–14). This is a distinct advantage of our developable strips compared to the arbitrary triangle strips used in computer graphics rendering. The C^0 continuity also has applications in industry. In some manufacturing process, due to the impulse response in step-motor, the toolpath can only be C^0 and thus can be regarded as polyline approximation of a smooth curve. Finally, since the physical surfaces have some thickness, a polishing step is used to generate a smooth thin-shell developable part; in this sense the function of global error controlling is important.

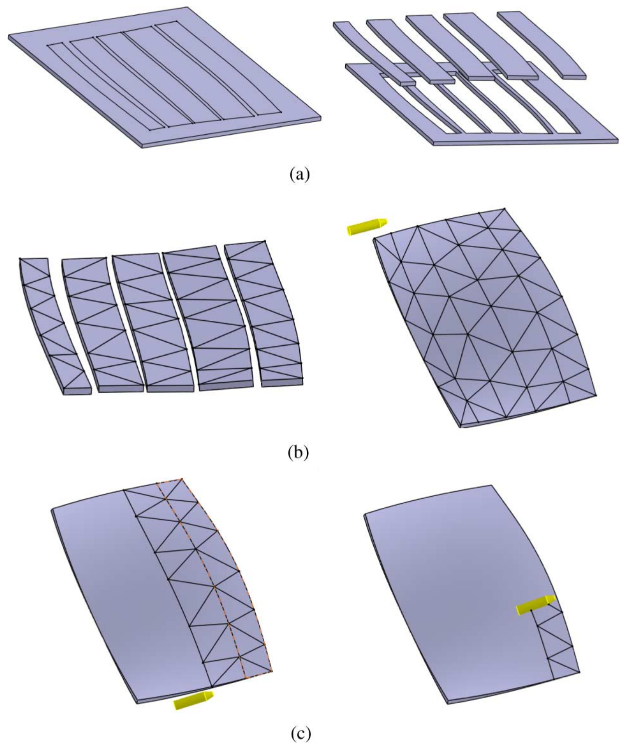


Fig. 16. Illustration of an application scenario. (a) Print the flattened pattern in a plate and cut the material: this step benefits from the minimal length of geodesics. (b) Bend and sew the pieces together: this step benefits from the stable mechanical property of geodesics and the minimal bending energy. (c) Fine polish the part to get a smooth surface: this step benefits from the global error control.

Choices of Cutting Lines: To cut surface into pieces, three types of cutting lines can be used: iso-parameter curves, geodesics and geodesic offsets of boundary curves.

- Iso-parameter curves versus geodesics. Iso-parameter curves are parameterization-dependent, while geodesics are consistent if the same surface with different parametrizations is used. Since the cutting lines are also the welding lines that sew the adjacent developable patches together and whose lengths should be minimized, using geodesics is better than iso-parameter curves. Another advantage of using geodesics is that, it is known in differential geometry that if a curve on surface only subjects to the internal surface forces, this curve can only be a geodesic; so using geodesics as the welding lines would make the overall product structure very stable.
- Geodesic offsets versus geodesics. Uniform width of each strip is a useful property in milling process. Geodesic offsets of boundary curves are better to achieve this property. However, observe that most surfaces designed in industry exhibit property of symmetries. So the surface can be approximately decomposed into pieces of generalized cylinder patches, in which the geodesic offsets of geodesic are themselves geodesics. In this case, cutting using the geodesics can also achieve almost uniform widths, as demonstrated by examples shown in Figs. 12–14. Together with the advantages of minimal cutting lengths and stable mechanical structures, we choose geodesics.

Limitations of the Method: Currently our method can only handle the rectangular free-form surfaces, especially for the tensor product surfaces. For arbitrary n-side surfaces with trimmed boundaries, one way is to use our method on the original untrimmed surface, and then trim the developable strips (see Fig. 13). However, this may not be as efficient as it could be. We will consider extension of our method along this direction in the future. Another limitation is that, if the surface of revolution is under consideration, the Elber's method [5] is superior to ours. In Fig. 15, we compare our method with the Hoschek's method [10] and the Elber's method [5]. Our method generally has the similar performance with the Hoschek's method, while the Elber's method outperforms both our method and the Hoschek's method.

Application Scenario: By applying our method, first the flattened pattern is printed in a planar material and then the material is cut. Refer to Fig. 16(a). Since geodesic is used, the lengths of cutting lines are minimized. Afterwards, the pieces of material are bent in space and sewed together along the geodesics. Since the geodesics are such curves that only subjects to the internal surface forces, using them as sewing lines can make the overall mechanical structure stable. Our method also minimizes the bending energy in the triangle strip generation, and thus, the bending step shown in the left of Fig. 16(b) is optimized. Finally, as shown in Fig. 16(c), the part is machined (typically using noncontact methods) to make the surface smooth. Since the global error of approximation is controlled, a fine polishing with small material removal is enough for this machining.

VII. Conclusion

In this paper, a simple and efficient algorithm is proposed to approximate a free-form surface with rectangular parameter domain using a small set of C^0-joint triangle strips. Different from graphics rendering, the resulting triangle strips are designed for developable approximation. We show by examples the proposed method can achieve several advantages in industrial manufacturing, including minimal cutting lengths, stable structure of sewing lines, minimal bending energy and global error control of the approximation. The proposed method has been implemented as a plugin module in the commercial CATIA CAD system.

Appendix

Let $S(u,v)$ be a tensor product surface with parameters (u,v). Denote

$$r(v)=S(u_0,v), r'(v)=a, r''(v) = b, \left.\frac{\partial S(u,v)}{\partial u}\right|_{u=u_0}=c.$$

If the iso-parameter curve $r(v)$ is a geodesic on the surface S, then the direction of the curvature vector $\kappa(v)n(v)$ of $r(v)$ must be coincided with the direction of the surface normal $N(u_0,v)$

$$\begin{aligned} n\times N=0 &\Rightarrow (a\times b\times a)\times(a\times c)\\ &=[(a\times b\times a)\cdot c]a\\ &=0\Rightarrow (a\times b\times a)\cdot c=0. \end{aligned}$$

Then, $c(v)$ is on the rectifying plane of $r(v)$. The above process is invertable. So it is a necessary and sufficient condition of $r(v)$ being a geodesic on S.

Acknowledgment

The authors would like to thank Dr. C. C. L. Wang at CUHK for providing them the data in Fig. 10(a).

References

[1] R. Bodduluri and B. Ravani, "Geometric design and fabrication of developable surfaces," *ASME Adv. Design Autom.*, vol. 2, pp. 243–250, 1992.

[2] R. Bodduluri and B. Ravani, "Design of developable surfaces using duality between plane and point geometries," *Comput.-Aided Des.*, vol. 25, pp. 621–632, 1993.

[3] H. Chen, I. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner, "On surface approximation using developable surfaces," *Graph. Models Image Process.*, vol. 61, pp. 110–124, 1999.

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

[5] G. Elber, "Model fabrication using surface layout projection," *Comput.-Aided Des.*, vol. 27, pp. 283–291, 1995.

[6] W. Frey, "Boundary triangulations approximating developable surfaces that interpolate a close space curve," *Int. J. Found. Comput. Sci.*, vol. 13, pp. 285–302, 2002.

[7] W. Frey, "Modeling buckled developable surface by triangulation," *Comput.-Aided Des.*, vol. 36, pp. 299–313, 2004.

[8] M. Gopi and D. Eppstein, "Single-strip triangulation of manifolds with arbitrary topology," *Eurographics*, pp. 371–379, 2004.

[9] J. Hoschek and H. Pottmann, "Interpolation and approximation with developable B-spline surfaces," in *Proc. Mathematical Methods for Curves and Surfaces*, M. Dahlen, T. Lyche, and L. L. Schumaker, Eds., 1995, pp. 255–264.

[10] J. Hoschek, "Approximation of surfaces of revolution by developable surfaces," *Comput.-Aided Des.*, vol. 30, pp. 757–763, 1998.

[11] H. Hoppe, "Optimization of mesh locality for transparent vertex caching," in *Proc. ACM SIGGRAPH'99*, 1999, pp. 269–276.

[12] S. Leopoldseder and H. Pottmann, "Approximation of developable surfaces with cone spline surfaces," *Comput.-Aided Des.*, vol. 30, pp. 571–582, 1998.

[13] Y. J. Liu, K. Tang, and A. Joneja, "Modeling dynamic developable meshes by Hamilton principle," *Comput.-Aided Des.*, vol. 39, pp. 719–731, 2007.

[14] G. Lukacs, A. Marshall, and R. Martin, "Faithful least squares fitting spheres, cylinders, cones, and tori for reliable segmentation," *Comput. Vis.*, pp. 671–686, 1998.

[15] T. Maekawa, "Computation of shortest paths on free-form parametric surfaces," *J. Mech. Des.*, vol. 118, pp. 499–508, 1996.

[16] F. Massarwi, C. Gotsman, and G. Elber, "Papercraft models using generalized cylinders," *Pacific Graphics*, pp. 148–157, 2007.

[17] J. Mitani and H. Suzuki, "Making papercraft toys from meshes using strip-based approximate unfolding," *ACM Trans. Graphics (SIGGRAPH'04)*, vol. 23, pp. 259–263, 2004.

[18] J. Peterson, "Tessellation of NURBS surfaces," in *Proc. Graphics Gems IV*, P. S. Heckbert, Ed., 1994, pp. 286–320.

[19] L. Peigl and M. Rchard, "Tessellating trimmed NURBS surfaces," *Comput.-Aided Des.*, vol. 27, pp. 16–26, 1995.

[20] L. Peigl and W. Tiller, "Geometry-based triangulation of trimmed NURBS surfaces," *Comput.-Aided Des.*, vol. 30, pp. 11–18, 1998.

[21] H. Pottmann and G. Farin, "Developable rational Bezier and B-spline surfaces," *Comput. Aided Geomet. Des.*, vol. 12, pp. 513–531, 1995.

[22] H. Pottmann and T. Randrup, "Rotational and helical surface approximation for reverse engineering," *Comput.*, vol. 60, pp. 307–323, 1998.

[23] H. Pottmann and J. Wallner, "Approximation algorithms for developable surfaces," *Comput. Aided Geom. Des.*, vol. 16, pp. 539–556, 1999.

[24] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C++*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[25] T. Randrup, "Approximation of surfaces by cylinders," *Comput.-Aided Des.*, vol. 30, pp. 807–812, 1998.

[26] J. Subag and G. Elber, "Piecewise developable surface approximation of general NURBS surfaces with global error bounds," *Proc. GMP2006, LNCS*, vol. 4077, pp. 143–156, 2006.

[27] K. Tang and C. Wang, "Modeling developable folds on a strip," *J. Comput. Inform. Sci. Eng.*, vol. 5, pp. 35–47, 2005.

[28] G. Weiss and P. Furtner, "Computer-aided treatment of developable surfaces," *Comput. Graph.*, vol. 12, pp. 39–51, 1988.

[29] X. Xiang, M. Held, and J. Mitchell, "Fast and effective stripification of polygonal surface models," in *Proc. Symp. Interactive 3D Graphics*, 1999, pp. 71–78.

Yong-Jin Liu received the Ph.D. degree from Hong Kong University of Science and Technology, Hong Kong, in 2003.

He is an Associate Professor with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include computer-aided design and computer graphics.

Yu-Kun Lai received the B.S. and Ph.D. degrees in computer science from Tsinghua University, Beijing, China, in 2003 and 2008, respectively.

He is currently a Research Fellow at Visual Media Research Center, Department of Computer Science and Technology, Tsinghua University. His research interests include computer graphics, geometry processing, computer-aided geometric design, and computer vision.

Shimin Hu received the Ph.D. degree from Zhejiang University, Zhejiang, China, in 1996.

He is currently a Chair Professor of computer science at Tsinghua University, Beijing, China. He is on the editorial board of *Computer-Aided Design*. His research interests include digital geometry processing, video-based rendering, rendering, computer animation, and computer aided geometric design.