Computer-Aided Design 43 (2011) 1089–1098

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Industrial design using interpolatory discrete developable surfaces

Yong-Jin Liu [a,*], Kai Tang [b], Wen-Yong Gong [c], Tie-Ru Wu [c]

[a] Department of Computer Science and Technology, Tsinghua National Lab for Information Science and Technology, Tsinghua University, Beijing, PR China
[b] Department of Mechanical Engineering, Hong Kong University of Science and Technology, Hong Kong, China
[c] School of Mathematics, Jilin University, Jilin, PR China

ARTICLE INFO

Article history:
Received 18 October 2010
Accepted 3 June 2011

Keywords:
Developable surfaces
Boundary triangulations
Industrial design

ABSTRACT

Design using free-form developable surfaces plays an important role in the manufacturing industry. Currently most commercial systems can only support converting free-form surfaces into approximate developable surfaces. Direct design using developable surfaces by interpolating descriptive curves is much desired in industry. In this paper, by enforcing a propagation scheme and observing its nesting and recursive nature, a dynamic programming method is proposed for the design task of interpolating 3D boundary curves with a discrete developable surface. By using dynamic programming, the interpolatory discrete developable surface is obtained by globally optimizing an objective that minimizes tangent plane variations over a boundary triangulation. The proposed method is simple and effective when used in industry. Experimental results are presented that demonstrate its practicality and efficiency in industrial design.

1. Introduction

A surface $\mathcal{S}$ is ruled if through every point on $\mathcal{S}$ there is a straight line (called a ruling) lying in $\mathcal{S}$. If at each ruling the tangent planes to $\mathcal{S}$ remain unchanged, $\mathcal{S}$ is a developable surface. Developable surfaces can be flattened into a plane without stretching and distortions [1]. This unique property makes developable surfaces much desired in many design tasks in the manufacturing industry, including the design of aircraft, ship hulls, automobiles and garments [2], using the inelastic material of sheet metal, plywood, cardboard and cloth, etc.

Most commonly used free-form surfaces in industry are doubly curved [3]. So special efforts have to be paid to design with developable surfaces. The first computer treatments of developable surfaces were given in [4,5], in which two classes of developable surfaces were considered. The first is constructed with a given directrix and a given generator direction, and the second is constructed by interpolating two boundary curves. Developable free-form surfaces, in terms of Bezier surfaces, were studied in [6,7] and later were extended to B-spline form in [8]. However, nonlinear characterizing equations needed to be solved for satisfying the developability conditions on these developable free-form surfaces. A novel work was presented in [9] that can generate developable Bezier surfaces through a Bezier curve of arbitrary degrees, without solving nonlinear characterizing equations. Aumann's developable Bezier surface [9] was extended to be compatible with B-spline control nets in [10].

Another distinct direction of designing with developable Bezier and B-spline surfaces was to use the dual space from the point of view of projective geometry. In this perspective, a developable surface is treated as the envelope of a one-parameter family of tangent planes, and thus can be represented by a curve in a dual projective 3-space. This direction was typified in the work [11–13]. If sufficient differentiability is assumed, developable surfaces can only be part of plane, cone, cylinder, tangent surface $\mathcal{S}(t, r) = c(t) + rc'(t)$ of a twisted curve $c(t)$ or a composition of them. Design of smooth developable surfaces using the above analytic methods often exhibits inflexibility behavior, i.e., very few degrees of freedom can be provided.

Towards a flexible and efficient tool in industrial design, mesh discretization of smooth developable surfaces has received considerable attention recently. Both quadrilateral and triangular meshes were adopted. Liu et al. [14] showed that the quadrilateral meshes with planar facets (called *PQ meshes*) are particularly suitable for free-form glass structures in architectural design. The relation between PQ meshes and conjugate curve networks was also given in [14]. For the discrete-differential-geometry properties of PQ meshes, a rigorous analysis was presented in [15]. Constrained triangular meshes are also studied in modeling and tessellating arbitrary free-form developable shapes [16–22]. The condition of a triangulation approximating a developable surface was given in [16]. A nice property of these mesh discretizations is that both smooth and buckled developable patches joined along crease lines can be modeled in a unified framework [15,23,24].

* Corresponding author. Tel.: +86 10 62780807.
E-mail address: liuyongjin@tsinghua.edu.cn (Y.-J. Liu).

0010-4485/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2011.06.001

In this paper, we adopt a constrained triangulation as a discrete developable surface. We formulate the constraints into optimization criteria and present a simple dynamic programming solution for finding such a mesh surface that achieves globally maximized developability. A distinct characteristic of our method is its compatibility with a user-friendly interactive design interface with sketching. Compared to previous strip-based triangulation methods [16,18,20], the presented method constructs developable meshes interpolating closed 3D curves which can enclose a multiple-connected region when projected back to the sketching plane.

2. Related work

Design directly using developable surfaces is desired in many industrial applications. Many different design styles have been proposed. In [7], to satisfy the nonlinear constraints, a strict and complicated process is required to set up the positions of control points of a developable Bezier surface. A better interaction process was proposed in [9], in which the user first draws a Bezier curve c_1 of arbitrary shape and degree, and then a developable Bezier surface is constructed by interpolating c_1 and some points on the other boundary curve c_2. This method however had the limitation that only five degrees of freedoms can be used to specify the point positions on the curve c_2. Design of a developable surface as a 3D curve in dual space offers a new perspective [11–13], but the user usually loses some intuitions when interactively designing in a projective dual space. A novel interactive style was presented in [25] with which the user can control a geodesic to efficiently animate a paper bending. The primitive shape represented in [25] is however restricted to a rectangle strip.

Design using discrete mesh surfaces widely enlarges the variety of surface shapes that can be represented by developable meshes. Rectangle-like developable triangle strips interpolating two arbitrary boundary curves were studied in [18,20]. The Hamilton principle was introduced in [24] to animate an inelastic disk-like sheet. With an optimized segmentation, developable triangle strips can be used to model complicated 3D graphics models [21]. An elegant optimization process was proposed in [14] for perturbing a quad mesh into a PQ mesh so that free-form shapes, especially architectural structures, can be represented. The PQ meshes were extended in [15] to model free-form developable surfaces with curved folds. The developable meshes generated by the method in this paper interpolates closed 3D curves and can model free-form shapes in an intuitive and efficient way.

Paper-and-pencil-based sketches are suited well to the traditional design habits of human beings. Many sketch-recognition systems have been proposed to support pen-input interaction concerning general 2D geometry constructions [26]. Inspired by the user-interface given in [19,27], in this paper we also develop a sketch-based interactive design interface: The user sketches arbitrary closed curves in a sketch plane and manipulates points' normals to generate 3D boundary curves. In this work, a dynamic programming solution is presented to efficiently generate a developable mesh surface interpolating the given arbitrary 3D boundary curves, whose projection into the sketch plane can bound a multi-connected region. Dynamic programming has been used in [18,22] for modeling developable meshes in rectangle-like strips. Since any triangle strip is trivially discrete developable, compared to the work in [18,22], in this paper we use an elegant measure of discrete developability and the major contribution is that by optimization using the measure of developability, the free-form developable shape that we can model is extended from simple rectangular strips [18,20,22] to branched shape (Fig. 10(a)) and shape with inner holes (Figs. 12 and 17).

3. Overview of the design system

The goal of the proposed system is to provide a simple and easy-to-use software tool with which common users can design free-form developable surfaces for diverse applications. The system interface (Fig. 1 left) explores a point in the tradeoff between expressiveness and naturalness. The user sketches (Fig. 1 right) arbitrary closed curves that are shown in the left panel of the interface. The planar sketching curves are obtained by tracing pen/mouse moving. The system reports the positions of pen/mouse motion using a polyline with a large number of vertices. The polyline is simplified using the classic Douglas–Peucker algorithm [28]. The simplified polyline is the initial draft capturing the design intention of the user. As it is often desired that the initial draft can be modified and refined by continuous interaction [29], the polyline is approximated by a B-spline curve of degree three. The approximation error is globally controlled within a tolerance $L/100$, where L is the length of the polyline. Non-professional users often prefer to modify the curve shape by directly moving curve-points (called *handle points* below). Advanced B-spline techniques [30] are provided in the system so that the user can modify the positions of handle points on the curve, add or delete some handle points on the curve, all through simple sketching operations [27]. One or more non-intersection closed curves representing a multi-connected region can be sketched in this phase.

The right panel of the system (left in Fig. 1) contains an OpenGL window that renders the closed planar curves in $\mathbb{R}^3$. The user can modify the positions of handle points in this window. But the movement of handle points is restricted in the normal direction of the sketching plane in the left panel. In this way the projection of 3D curves (in the right panel) onto the sketching plane is guaranteed to be the sketched 2D planar curves (in the left panel). The interactive design of closed 3D curves is continuous in both left and right panels.

Given a set of closed 3D curves as boundaries, an interpolatory discrete developable surface is generated using the method proposed in the next section. The user can browse the results in different rendering modes (shading, wireframe and others) also in the right panel. Since the proposed developable surface interpolation method is fast, the user can interactively modify the shape of boundary curves and re-generate the interpolating surface in real-time. User experiences in Section 5 show that great productivity is achieved by this interactive manner and diverse free-form shapes can be designed directly using discrete developable surfaces.

4. Interpolatory discrete developable surface

In this section we present a special type of triangulation as a discrete developable surface and impose a measure of discrete developability on it. Firstly the 3D curves sketched in the system are discretized into polylines P using the solution to a $Min - \#$ problem [18]. Given polylines P with edges E_P and vertices V_P, a *boundary triangulation* of P is defined as a triangulation T of V_P that satisfies the following conditions:

1. $E_P \subseteq E_T$ and $V_P = V_T$, where E_T and V_T are the edge set and vertex set of T;
2. For each $v \in V_T$, the local surrounding area of v in T is homeomorphic to a two-dimensional half space $H^2 = \{(x_1, x_2) \in \mathbb{R}^2, x_1 \geq 0\}$;
3. If an edge $e \in T$ is incident to only one face, then $e \in E_P$.

Some notations used in this paper are summarized in Table 1.

Observing the fact that along each ruling of a developable surface, the tangent plane does not change, we define the following objective function with which the optimal boundary triangulation maximizes a discrete developability measure.

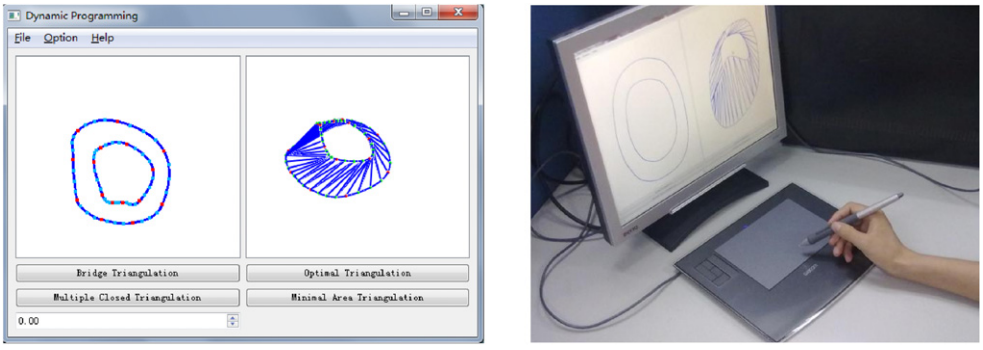

Fig. 1. The interface of the working system. Left: A screen snapshot. Right: A typical working environment of the sketching system.

Table 1
Some notations used in this paper.

Notation	Meaning
$T(P)$	A boundary triangulation of P
$OT(P)$	The optimal boundary triangulation of P That maximizes the measure $M(T)$.
π	The sketch plane
P_{2D}	The projection of P onto π
Supporting line segment (SLS)	A line segment inside P that starts at some Vertex $p_i \in P$, goes through some concave Vertex $p_j \in P$ and ends in an edge of P.
A rung $\langle m, n\rangle$	A line segment connecting p_m and p_n that is Entirely inside P except its two ends.
A gate rung	A rung that is a part of an SLS
A regular rung	A rung that is not a gate rung
Optimal rung	A rung that is an edge in $OT(P)$
Bridge triangulation $BT(P)$	A boundary triangulation in which every triangle has at most two rungs.
$OBT(P)$	The optimal bridge triangulation of P that Maximizes the measure $M(BT)$.
$L(i)$	The set of valid rungs emanating from p_i

Let P discretize smooth curves C. The vertices in P are oriented so that when one walks along the P, the interior region bounded by P (P possibly has several disjoint closed polylines) is always on the right-hand side. For any boundary triangulation T of P, each edge $e \in T$ consists of two vertices $p_i, p_j \in V_P$. Denote the tangent vectors in the smooth curves C at the positions of p_i, p_j by $\mathbf{t}_i, \mathbf{t}_j$, respectively. The discrete normal vector to the discrete surface T at position p_i, with respect to p_j, is defined by $\mathbf{n}_i^j = \frac{(p_j - p_i)\times \mathbf{t}_i}{\|(p_j - p_i)\times \mathbf{t}_i\|}$. A twist measure of edge $e = (p_i, p_j)$ is defined by

$$m(e) = \begin{cases} \mathbf{n}_i^j \cdot \mathbf{n}_j^i - 1, & e \notin P \\ 0 & e \in P. \end{cases}$$

The *discrete developability* of T is measured by the total twist $M(T)$:

$$M(T) = \sum_{e\in T, e\notin P} m(e). \tag{1}$$

Since the measure $m(e)$ is always less than or equal to zero, and $m(e) = 0$ only if perfect developability is achieved, given an T, the larger $M(T)$ is, the better the developability obtained. Note that a similar measure of developability for semi-discrete surfaces is used in [31] to define developable surface strips. Among all the possible boundary triangulations of P, let OT be the triangulation that maximizes the measure $M(T)$. If the sampling density of P is increased to the infinate, the boundary triangulation $OT(P)$ maximizing the developablity measure (1) converges to a smooth developable surface or a composition of several developable surfaces joined along crease lines, while other boundary triangulations may converge only to a ruled surface. In this paper, $OT(P)$ is regarded as a quasi-developable surface of P.

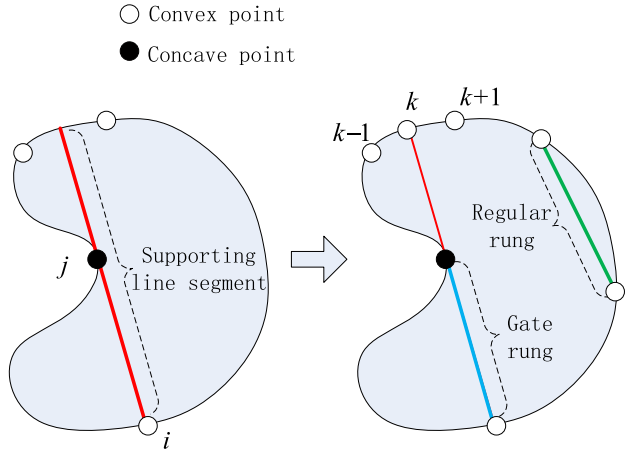

Fig. 2. Polyline resampling: Supporting line segment (SLS), gate and regular rungs.

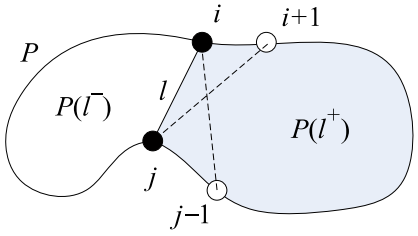

Fig. 3. Polyline P partitioning by a rung l: Two halves $P(l^+)$ and $P(l^-)$, and two bridge triangles from l in $P(l^+)$.

4.1. Optimal bridge triangulation of a single closed curve

Let π be the sketch plane specified in Section 3. Denote the projection of P onto π by P_{2D}. Let P_{2D} be indexed using the right-hand-side rule specified at the beginning of Section 4. With this order, we make no distinction between sampling P and its polyline. A point in P is said to be concave in $\mathbb{R}^2$ if its interior angle is larger than π, otherwise it is convex in $\mathbb{R}^2$. Refer to Fig. 2. A *supporting line segment* (SLS) of P is a line segment inside P that starts at some vertex $p_i \in P$, supports (goes through) some concave vertex $p_j \in P$ and ends in an edge of P. Assume that the original polyline P can be resampled so that for any SLS, both its ends are vertices in P. A line segment connecting p_m and p_n, denoted as $\langle m, n\rangle$, is called a *rung* if it is entirely inside P except its two ends. In particular, it is called a *gate rung* if it is a part of an SLS, e.g., $\langle i, j\rangle$ in Fig. 2; otherwise, it is a *regular rung*. When a rung $\langle i, j\rangle$ is referred to, it is assumed that $i < j$.

As P is simple, any rung $l = \langle i, j\rangle$ divides P into two halves. Denote by $P(l^+) = P(\langle i, j\rangle^+)$ the half that has the vertex p_{i+1} and by $P(l^-) = P(\langle i, j\rangle^-)$ the other half (Fig. 3). If a rung l is an edge

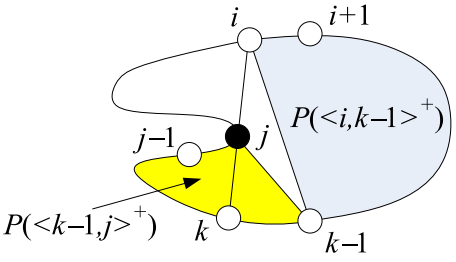

Fig. 4. Topological structure of P around a gate rung $\langle i, j\rangle$.

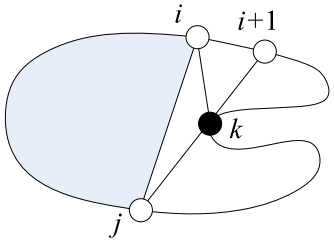

Fig. 5. Change of topological structure when a gate rung is encountered.

in an OT of P (referred to as an *optimal rung*), it is obvious that the following equations hold:

$$M(OT(P)) = M(OT(P(l^+))) + M(OT(P(l^-))) + m(l). \tag{2}$$

Recall that $OT(P)$ is the optimal triangulation of sampling P under the twist measure $M(\cdot)$. We first study a particular type of triangulation called *bridge triangulation* (BT): BT is a boundary triangulation in which every triangle has at most two rungs.

Refer to Fig. 3. For any optimal bridge triangulation (OBT) of $P(l^+)$, it must have either rung $\langle i, j-1\rangle$ or $\langle i+1, j\rangle$ as an edge. The following recursive rule holds for a regular optimal rung $l = \langle i, j\rangle$:

$$\begin{aligned} &M(OBT(P(\langle i,j\rangle^+))) \\ &\quad = \max\{M(OBT(P(\langle i+1,j\rangle^+))) + m(\langle i+1,j\rangle), \\ &\qquad M(OBT(P(\langle i,j-1\rangle^+))) + m(\langle i,j-1\rangle)\} \end{aligned} \tag{3}$$

with the boundary condition that this recursion terminates at $j = (i+2) \bmod n$. The recursive rule for $M(OBT(P(\langle i, j\rangle^-)))$ is defined similarly. Given rule (3), the dynamic programming method [32] can be applied to find an OBT of P if the projection of P onto the sketch plane is convex.

The dynamic programming method has been used in [18,22] to obtain an optimal bridge triangulation of a rectangular strip. However, concave vertices are not considered in [18,22]. In general, the projection of P could enclose a concave region in the sketch plane π. Refer to Fig. 4. Consider a gate rung $l = \langle i, j\rangle$ with j being the supported concave vertex and k the vertex on the other end of the associated SLS. At a concave vertex p_j, $P(\langle i+1, j\rangle^+)$ is well defined, but not $P(\langle i, j-1\rangle^+)$ since $\langle i, j-1\rangle$ is not a rung (it intersects P). To extend rule (3) for including concave vertices, the following recursive rule is established for a gate rung $\langle i, j\rangle$:

$$\begin{aligned} &M(OBT(P(\langle i,j\rangle^+))) \\ &\quad = \max\{M(OBT(P(\langle i+1,j\rangle^+))) + m(\langle i+1,j\rangle), \\ &\qquad M(OBT(P(\langle i,k-1\rangle^+))) + m(\langle i,k-1\rangle) \\ &\qquad + M(OBT(P(\langle k-1,j\rangle^+))) + m(\langle k-1,j\rangle)\}. \end{aligned} \tag{4}$$

In rule (4), we assume $i < k < j$ after a cyclic reordering of P. Rule (4) considers the case that the gate rung $\langle i, j\rangle$ is moving out of a pocket. The opposite case is that the rung propagation enters a pocket, as shown in Fig. 5. The corresponding rule of this opposite case is as follows:

$$\begin{aligned} M(OBT(P(\langle i,j\rangle^+))) &= M(OBT(P(\langle i,k\rangle^+))) + m(\langle i,k\rangle) \\ &\quad + M(OBT(P(\langle k,j\rangle^+))) + m(\langle k,j\rangle). \end{aligned} \tag{5}$$

Generally in each SLS there is only one concave vertex. In rare scenarios, degenerate cases may occur that several concave vertices lie in an SLS. Refer to Fig. 6: Two concave vertices p_j and p_k lie in an SLS. In this case, we duplicate p_k with a dummy convex vertex p'_k. Then we treat (i, j, k') and (i, k, l) as two different SLSs. Applying dynamic programming on this treatment will produce a dummy triangle $\triangle(i, k, k')$ in an optimal triangulation OT. To process degeneracies, we find and delete all the dummy triangles in OT. Our split-dummy-vertex method falls into the symbolic perturbation strategy in [33] in which elegant analysis on this strategy is discussed in a much broader scope.

Given rules (2–5), the standard top–down dynamic-programming algorithm [32] is applied to find an OBT of P. To start up the algorithm, consider an arbitrary convex vertex p_i. Let $L(i)$ be the set of valid rungs emanating from p_i. Any valid triangulation of P must contain either a rung in $L(i)$ or the rung $\langle i-1, i+1\rangle$. Therefore by applying rule (2) to every rung in $L(i) \cup \{\langle i-1, i+1\rangle\}$, the triangulation with the maximum twist measure (1) is an OBT of P.

A memorization technique [32] is used to improve the efficiency of the dynamic programming method while maintaining the top–down strategy. Explicitly, two $n \times n$ scalar matrices M^+ and M^- are utilized to record the twist values $M(OBT(P\langle i, j\rangle^+))$ and $M(OBT(P\langle i, j\rangle^-))$. i.e., $M^+[i, j] = M(OBT(P\langle i, j\rangle^+))$ and $M^-[i, j] = M(OBT(P\langle i, j\rangle^-))$. In addition to obtain the maximum twist value of the OBT, another $n \times n$ structural matrix N is built to record the local advancement of the rungs returned from rules (2–5), so that the final OBT can be reconstructed. The total running time of the memorized dynamic-programming algorithm is $O(n^3)$, where n is the number of sample points in P.

4.2. Optimal triangulation of a single closed curve

The optimal triangulation obtained by applying the rules (2–5) is limited to bridge triangulation only, i.e., every triangle must have an edge in P. Thus the shape shown in Fig. 7 cannot be found. Next we develop a recursive rule for the general optimal triangulation OT.

Let $l = \langle i, j\rangle$ be a rung, either regular or gate. A vertex p_k is said to be *rung-visible* from l if both $\langle i, k\rangle$ and $\langle k, j\rangle$ are rungs (again we assume $i < k < j$ after a cyclic reordering). Since P is simple, if p_k is rung-visible, the triangle (p_i, p_j, p_k) lies entirely inside P except the three vertices. Denote by $V(i)$ the set of indices of the vertices rung-visible from p_i, the following rule for the general optimal triangulation is established:

$$\begin{aligned} M(OT(P(\langle i,j\rangle^+))) = \max\{&M(OT(P(\langle i,x\rangle^+))) + m(\langle i,x\rangle) \\ &+ M(OT(P(\langle x,j\rangle^+))) \\ &+ m(\langle x,j\rangle) : x \in V(i) \cap V(j) \cap P(\langle i,j\rangle^+)\}. \end{aligned}$$

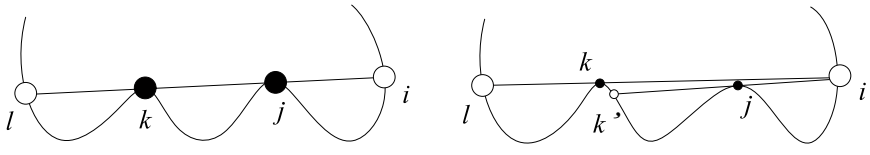

Fig. 6. Handling degenerate cases when multiple concave vertices lie in a SLS.

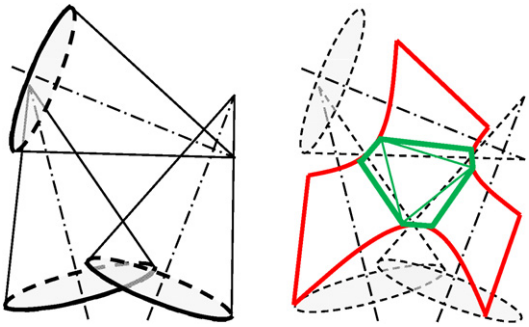

Fig. 7. A composite developable surface: Three cone patches joined with a plane, and the boundary curve is shown in red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Recall that if $e \in P, m(e) = 0$, i.e., $M(T)$ does not count the edges in P, we can slightly modify the above rule to include the OBT case:

$$\begin{aligned} M(OT(P(\langle i,j\rangle^+))) = \max\{&M(OT(P(\langle i,x\rangle^+))) + m(\langle i,x\rangle) \\ &+ M(OT(P(\langle x,j\rangle^+))) \\ &+ m(\langle x,j\rangle) : x \in (V(i) \cup i+1) \\ &\cap (V(j) \cup j-1) \cap P(\langle i,j\rangle^+)\}. \end{aligned} \tag{6}$$

By using rules (2) and (6) with the boundary conditions specified by rules (2–5), the general OT is found after the recursion of the top–down dynamic-programming algorithm. Note that the rung-visibility is locally updated after each marching step, so that $V(i)$ and $V(j)$ only contain non-visited vertices.

To implement the algorithm, we need to compute the rung-visibility set $V(i)$ for all vertices in P. This equals building a visibility graph $\mathcal{VG}(P)$ of a simple polygon P and removing the edges in P from $\mathcal{VG}(P)$. We use the optimal output-sensitive algorithm in [34] to build $\mathcal{VG}(P)$ in $O(m + n \log\log n)$ time, where m is the number of edges in $\mathcal{VG}(P)$.

4.3. Optimal triangulation of multiple closed curves

Generally a developable surface can have multiple closed curves as its boundary. Our sketching interface provides an flexible tool for constructing this kind of boundary with continuous interactive editing: First, multiple planar closed curves that enclose a multiple-connected region are specified in the sketch plane; then the handle points of B-spline curves are modified along the normal direction to make 3D boundaries.

Consider a non-simple polygon P in the sketch plane which is bounded by the curve set $\Omega = \{Q, H_1, H_2, \ldots, H_m\}$, where Q is the external closed boundary curve and $H_1, H_2, \ldots, H_m$ are m non-intersecting holes forming the internal boundary of P. The vertices on holes are ordered counter-clockwise, so that when one walks along the boundary Ω, the interior region is always on the right-hand side. Refer to Fig. 8. Our strategy is to cut P along some rungs and then convert P into a simple polygon P', for which the rules (2–6) can be applied.

To guarantee $OT(P) = OT(P')$, the choice of cutting rungs is important. Let v_c^i be a vertex on the convex hull of a hole H_i. v_c^i is called a *sink* if every rung in $L(v_c)$ connects H_i to Q. We have the following two properties:

Property 4.1. *If three sequential vertices v_{j-1}^i, v_j^i, v_{j+1}^i of H_i lie on the convex hull of H_i, then any boundary triangulation of P must contain at least one rung in $L(v_j^i)$ that connects H_i to another boundary curve in $\Omega \setminus H_i$.*

Proof. Denote the three sequential vertices of H_i that lie on the convex hull of H_i, in counter-clockwise order, by v_{j-1}^i, v_j^i and v_{j+1}^i. If Property 4.1 does not hold, then all the rungs in $L(v_j^i)$ connect v_j^i to some other vertex v_k^i of H_i. Since (1) any boundary triangulation T of P contains v_j^i as a vertex which has a degree no smaller than 3, and (2) two edges (v_{j-1}^i, v_j^i) and v_j^i, v_{j+1}^i lie on the convex hull of H_i, all remaining edges (a nonempty set) in T which are incident to v_j^i must cross the interior of hole H_i. That is a contradiction to the definition of boundary triangulation. So Property 4.1 holds. □

Property 4.2. *If three sequential vertices v_{j-1}^i, v_j^i, v_{j+1}^i of H_i lie on the convex hull of $\{H_1, H_2, \ldots, H_m\}$, then v_j^i is a sink and any boundary triangulation of P must contain at least one rung in $L(v_j^i)$ that connects H_i to Q.*

Proof. Let Ω be the convex hull of inner boundaries $(H_1, H_2, \ldots, H_m)$, and v_{j-1}^i, v_j^i, v_{j+1}^i be three sequential vertices of H_i, in counter-clockwise order, lying on Ω. If v_j^i is not a sink, then all the rungs in $L(v_j^i)$ connect v_j^i to some other vertex v_k^j of H_j. Since v_{j-1}^i, v_j^i, v_{j+1}^i lies on Ω, all the rungs in $L(v_j^i)$ must cross the interior of hole H_i, a contradiction. So Property 4.2 holds. □

Since boundary Ω consists of smooth curves, if the sampling of Ω is sufficiently dense, a sink vertex always exists. Based on the Properties 4.1 and 4.2, we use the following procedure to find an optimal set of cutting rungs.

Step 1. Let external boundary be Q and internal boundary $\mathcal{H} = \{H_1, H_2, \ldots, H_m\}$.
Step 2. Find the convex hull of $\mathcal{H}$ and identify a sink vertex v_c.
Step 3. For every rung l in $L(v_c)$.
Step 3.1. Cut the region using l which connects a H_k to Q;
Step 3.2. Set $Q = Q \cup H_k$ and $H = H \setminus H_k$;
Step 3.3. If $H \neq \emptyset$.
Step 3.3.1. Go to Step 2.
Step 3.4. Else stop.

The convex hull of a simple polygon can be efficiently found in linear time [35]. To find the convex hull of several polygons, we apply the marriage-before-conquest method in [36] that runs in $O(n \log h)$ time, where n is the number of vertices of input polygons and h is the number of vertices on the output hull. Finally, for helping identify sink vertices, we use the output-sensitive scheme in [37] to build the visibility graph of simple polygon Q

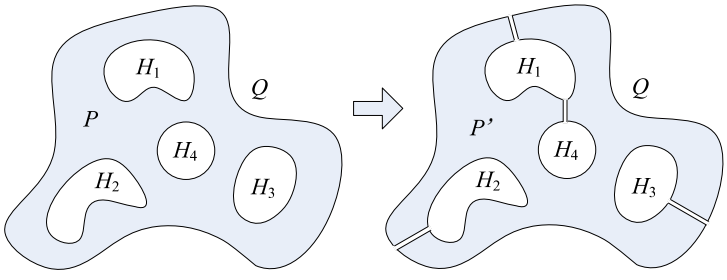

Fig. 8. Cutting a non-simple polygon into a simple one along some rungs.

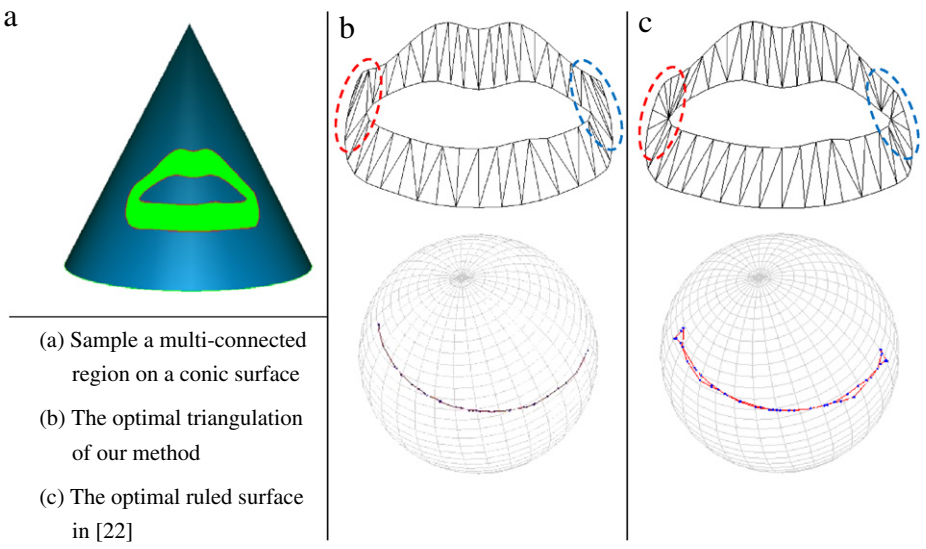

Fig. 9. Reconstruction of a multi-connected region in a conic section, using the proposed optimal triangulation method and the optimal ruled surface method [22], respectively. Dash circles outline the major difference in triangulations. The developability is visualized using Gaussian images.

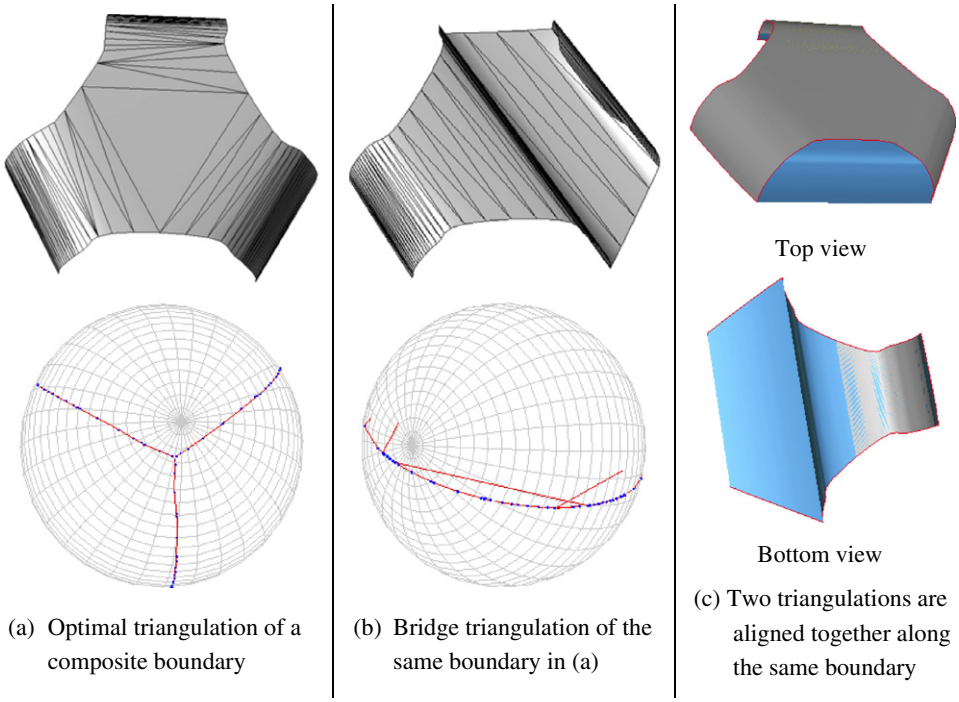

Fig. 10. Developable surface reconstruction by interpolating a composite boundary: (a) the optimal triangulation; (b) the optimal bridge triangulation; (c) two triangulations have the exact same boundary.

with respect to m polygonal obstacles $\{H_1, H_2, \ldots, H_m\}$, in optimal $O(E + T + m \log n)$ time, where E is the size of the visibility graph and T is the time required to triangulate the simple polygon with obstacles. The overall dynamic-programming algorithm for multiple close curves has the time complexity of $O(n^{m+3})$ in the worst case. Since we use several output-sensitive schemes in the algorithm, in all our experiments, our algorithm is fast: If there are a few hundred sample points, the running time is less than one second. In our experiments, all the tests are performed on a Core2Duo 2.4 GHz laptop computer with 2GB of memory.

5. Experiments

The proposed discrete developable surface design system (Fig. 1) is built on the platform of Visual C++.net with Qt 4 GUI. The distinct features of the system include supporting sketch input and supporting free-form shape design using discrete developable surfaces. Below we present a series of experiments to demonstrate these features. Since along each ruling of a developable surface $\mathcal{S}$, the tangent plane does not change, the Gaussian image of $\mathcal{S}$ is a spherical arc. Based on this property, we use the Gaussian image to visualize the quality of discrete developability.

Developable surface design with a multi-connected region. As a simple and demonstrative example, we sample a multi-connected region on a (developable) conic surface (Fig. 9(a)). Given the sampled boundary P, we apply our algorithm to reconstruct the developable patch interpolating P (Fig. 9(b)). Fig. 9(c) shows the comparison to the optimal ruled surface using the method in [22]. Since our algorithm minimizes the tangent plane variation and

Table 2
Statistic data of the models presented in Section 5. The interactive design time includes the creative idea generation.

Model name	Pieces of dev. patches	Triangle number	Interactive design time (min)
Tent (Fig. 12 bottom)	4	407	15
Lantern (Fig. 13)	12	1152	5
Flower (Fig. 14)	36	2304	75
Ship hull (Fig. 15)	4	160	20
Car hood (Fig. 16)	8	5928	60
Architectural base (Fig. 17 bottom)	1	264	10

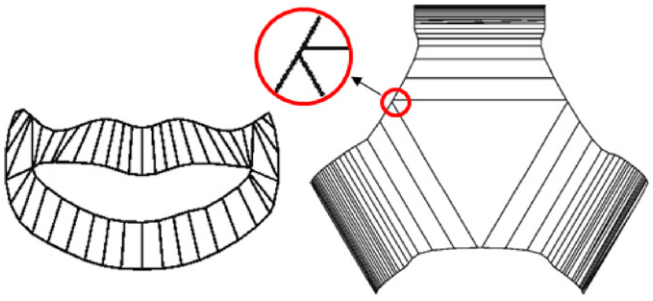

Fig. 11. Quadrilateral meshes converted from the optimal triangulations shown in Figs. 9(b) and 10(a), respectively.

the optimal triangulation is not limited to bridge triangles, it is clearly shown in the companion Gaussian images that our optimal triangulation of P better approximates the developable surface than the optimal ruled surface method in [22]. Other examples of free-form surface design with a multi-connected region are presented in Figs. 12 and 17. Compared to the previous work [18,20,22] that builds bridge triangulations in rectangular strips, the shape that encloses multiple inner holes in the sketch plane π (e.g., the ones shown in Figs. 12 and 17) can only be modeled by the method proposed in this paper.

Composite developable surface design. Our algorithm designs developable surfaces by interpolating given boundaries. In this manner composite developable surfaces can be efficiently built as follows. Given two pieces of patch boundaries with consecutive portions $P_1 = \{b_1^1, \ldots, b_1^n\}$ and $P_2 = \{b_2^1, \ldots, b_2^m\}$, if some portions in two patches satisfy $b_1^i = b_2^j$, then the two patches can be joined together and we denote this by $P_1 \oplus P_2 = \{\ldots, b_1^{i-1}, b_2^{j+1}, \ldots, b_2^{j-1}, b_1^{i+1}, \ldots\}$. We can further impose continuity conditions such as G^1 or G^2 between boundary segments b_1^{i-1}, b_2^{j+1} and b_2^{j-1}, b_1^{i+1}, so that smooth composite boundary $P_1 \oplus P_2$ can be achieved. Fig. 10 shows developable surfaces interpolating smoothly joined composite boundary P which is akin to the composite example shown in Fig. 7. Fig. 10(a) shows the optimal triangulation $OT(P)$ and Fig. 10(b) shows the special bridge optimal triangulation $OBT(P)$ from [22]. The reconstruction quality of these two triangulations is shown in the Gaussian images. This example clearly demonstrates that using only bridge triangles can find limited success in a few special cases [20,22] while the proposed optimal triangulation is a generalization suitable in a wider range of applications.

Connection to quadrilateral meshes. Both quadrilateral and triangular meshes are widely used for a discrete surface representation in industry. An optimal triangulation obtained from the proposed algorithm can be post-processed into a quadrilateral mesh, which in turn can serve as an initial input towards an optimal PQ mesh [14,15]. Let the optimal triangulation $OT(P)$ of boundary P be denoted by a graph $G(V, E)$, where $V \subseteq P$. For each vertex $v \in V$, if the degree of vertex $\deg(v) > 3$, then v is called a candidate vertex. For each edge $e \in E$, if both its vertices are candidate vertices, then e is called a candidate edge. For each candidate edge ce, denote the dihedral angle of two triangles incident to ce by $Dih(ce)$. The following algorithm converts an optimal triangulation $OT(P) = G(V, E)$ into a quadrilateral mesh.

1. Compute the degrees of all vertices in G and identify all the candidate vertices.
2. Identify all the candidate edges and put them into a heap $\mathcal{H}$ indexed by $Dih(ce)$.
3. While $\mathcal{H}$ is not empty
   3.1. Extract a candidate edge ce with minimum index from $\mathcal{H}$.
   3.2. Delete ce from E and locally update $\mathcal{H}$.
4. Update G and identify all the candidate vertices in G.
5. For each candidate vertex v

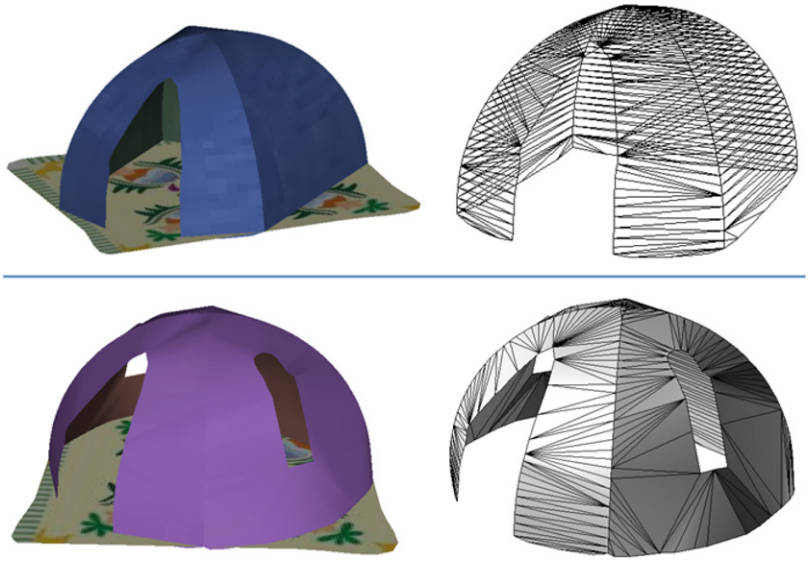

Fig. 12. Two tent models. Top: Composite quasi-developable surface design without holes. Bottom: Windows are modeled by trimming two patches with inner holes.

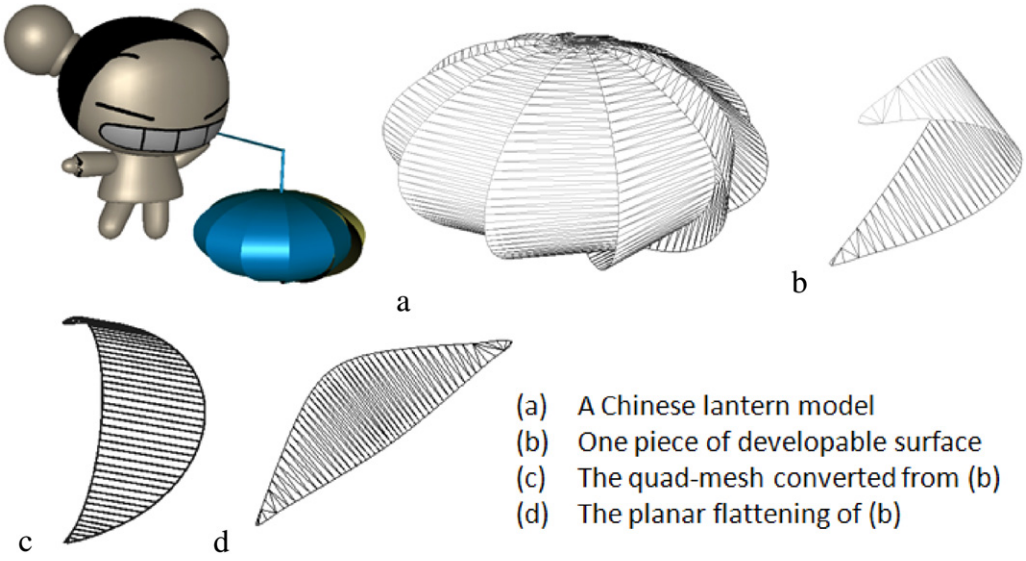

Fig. 13. A Chinese lantern model made up of 12 quasi-developable patches with 1152 triangles.

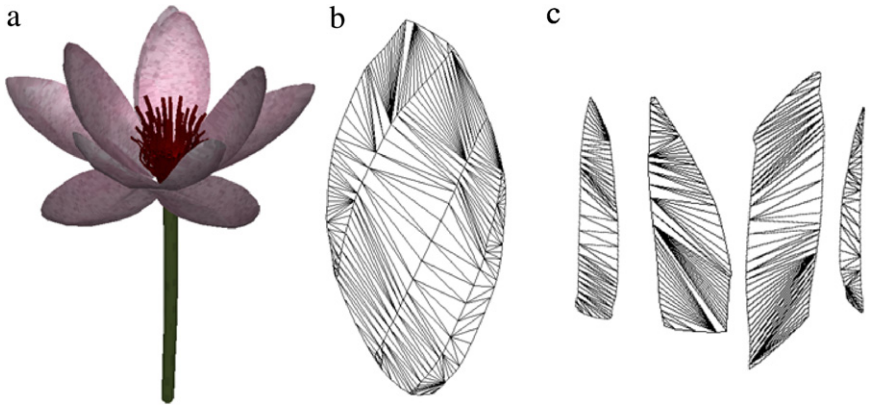

Fig. 14. A flower model made up of 9 petals with 2304 triangles: (a) the rendering; (b) one petal; (c) the planar flattening of (b).

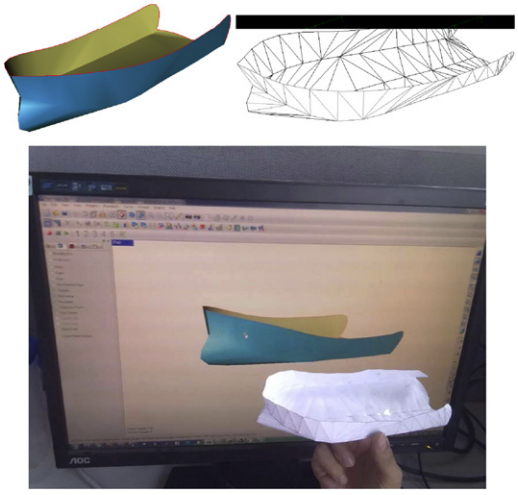

Fig. 15. A ship hull model. Top row: The ship hull model in Gouraud shading and wireframe representations. Bottom row: a papercraft model made from planar flattening of discrete developable patches.

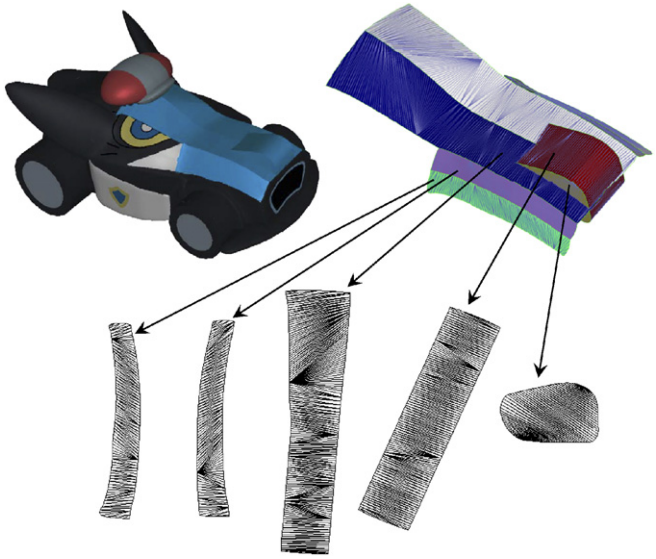

Fig. 16. A car hood model made up of 8 quasi-developable patches.

5.1. Duplicate the vertex with $\deg(v) - 2$ copies and distribute the copies in a predefined small interval $(-\varepsilon, \varepsilon)$ centered at position of v on P.

5.2. Locally update the connectivity of v in the subgraph incident to v, such that each copy of v has degree 3.

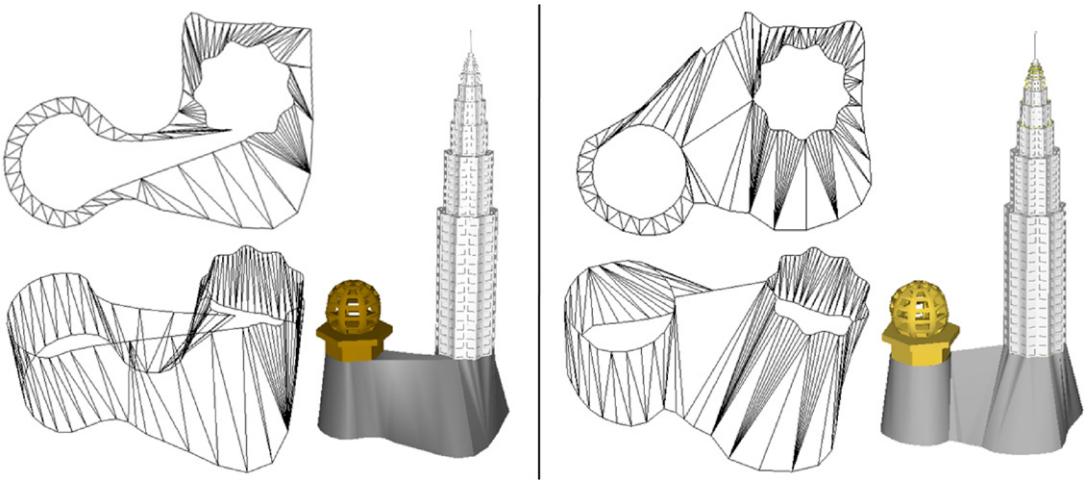

Fig. 17. Architectural base design with a projected multi-connected region. Left: One inner hole. Right: Two inner holes.

Fig. 11 shows two examples of quadrilateral meshes converted from the examples shown in Figs. 9(b) and 10(a). As shown in Fig. 11, few planar polygons other than quadrangles may be output from the above algorithm and they can be further converted into quadrilaterals using the methods in [38].

Evaluation of sketch-based interface. To the authors' best knowledge, there is no commercial system that offers the function of design directly using free-form developable surfaces. In [18] a plug-in module of approximating free-form B-spline surface by developable strips with a controllable global error bound is developed in the CATIA V5 R16 system. We present both the CATIA V5 R16 system with the developable approximation module and the proposed sketching system to nine college students (4 females and 5 males) aged between 17 ad 23. The same design task to be fulfilled by both systems was assigned to the nine users. Then we asked the users to comment on both systems. All the users commented that CATIA is powerful, but it is not easy to learn at first glance. All the users also commented that the sketching system is simple and easy to use, and the design intent of users can be addressed fluently. Table 2 summarizes the interactive design time (including the creative idea generation) of six sophisticated models using the sketching interface. With this data, we conclude that the sketching interface explores a point in the tradeoff between expressiveness and naturalness, with which users can focus on the task without excessively switching among menus, buttons and keyboard hits.

Aesthetic and industrial design. The sketching system provides a promising tool for using developable surfaces in aesthetic and industrial design. Figs. 12 and 13 show two tents and a Chinese lantern, which are made of 4 and 12 quasi-developable patches, respectively. Fig. 14 shows a composite flower model that uses 36 quasi-developable patches: One graduate student took 75 min to sketch all the boundaries and a total of 2304 triangles are generated by the optimal triangulations. A ship hull consisting of 4 quasi-developable patches, in a symmetric configuration, is shown in Fig. 15 in which the papercraft model constructed from planar development is also presented. A sophisticated car model is presented in Fig. 16 in which the hood is designed using a composite quasi-developable surface. An example in architectural design is presented in Fig. 17. Table 2 summarizes the statistics data of all the models (developable patches, number of triangles, and the interactive design time using the sketching interface). Once the boundary curves are specified, the models in all the examples are generated in time between one to one hundred milliseconds, on a Core2Duo 2.4 GHz laptop computer with 2GB memory, indicating that real-time shape manipulation and modification are achieved by the proposed system.

6. Conclusions

In this paper, we have presented a dynamic programming method for generating optimal discrete developable surfaces interpolating given 3D boundary curves. Compared to previous works, a distinct feature of the proposed method is that our design system separates developable surface modeling into three phases: (1) interactive design of boundary curves in a sketching plane, (2) interactive modification along normal directions, and (3) automatic generation of discrete developable surfaces that interpolate the given boundaries. Unlike previous strip-based methods, on the sketching plane, the boundaries in our system can enclose several inner holes and then form a multi-connect region. To facilitate the proposed system, a sketching interface is utilized that offers a design tool to allow designers to construct sophisticated free-form shapes naturally and efficiently. Experimental results are presented, showing that sophisticated free-form developable surfaces can be designed by users in a low cognitive load, by using the three-phase design system with a sketching interface.

Acknowledgments

The authors thank the reviewers for their comments that help improve this paper. This work was supported by the National Science Foundation of China (Project 60970099, 60736019), the National Basic Research Program of China 2011CB302202 and Tsinghua University Initiative Scientific Research Program 20101081863.

References

[1] do Carmo M. Differential geometry of curves and surfaces. Prentice-Hall, Inc.; 1976.
[2] Liu Y, Zhang D, Yuen M. A survey on CAD methods in garment design. Computers in Industry 2010;61(6):576–93.
[3] Farin G. Curves and surfaces for CAGD. 5th ed. Morgan Kaufmann Pub.; 2002.
[4] Gurunathan B, Dhande S. Algorithms for development of certain classes of ruled surfaces. Computers & Graphics 1987;11(2):105–12.
[5] Weiss G, Furtner P. Computer-aided treatment of developable surfaces. Computers & Graphics 1988;12(1):39–51.
[6] Aumann G. Interpolation with developable Bezier patches. Computer Aided Geometric Design 1991;8(5):409–20.
[7] Lang J, Roschel O. Developable $(1, n)$-Bezier surfaces. Computer Aided Geometric Design 1992;9(4):291–8.
[8] Maekawa T, Chalfant J. Design and tessellation of b-spline developable surfaces. Transactions of the ASME. Journal of Mechanical Design 1998;120(3):453–61.
[9] Aumann G. A simple algorithm for designing developable Bezier surfaces. Computer Aided Geometric Design 2003;20(8–9):601–19.

[10] Fernandez-Jambrina L. B-spline control nets for developable surfaces. Computer Aided Geometric Design 2007;24(4):189–99.
[11] Bodduluri R, Ravani B. Design of developable surfaces using duality between plane and point geometries. Computer-Aided Design 1993;25(10): 621–32.
[12] Bodduluri R, Ravani B. Geometric design and fabrication of developable Bezier and b-spline surfaces. Transactions of the ASME. Journal of Mechanical Design 1994;116(4):1042–8.
[13] Pottmann H, Farin G. Developable rational Bezier and b-spline surfaces. Computer Aided Geometric Design 1995;12(5):513–31.
[14] Liu Y, Pottman H, Wallner J, Yang Y, Wang W. Geometric modeling with conical meshes and developable surfaces. In: ACM SIGGRAPH 2006. 2006. p. 681–9.
[15] Kilian M, Flory S, Chen Z, Mitra N, Sheffer A, Pottmann H. Curved folding. In: ACM SIGGRAPH 2008. 2008. p. Article No. 75.
[16] Frey W. Boundary triangulations approximating developable surfaces that interpolate a closed space curve. International Journal of Foundations of Computer Science 2002;13(2):285–302.
[17] Julius D, Kraevoy V, Sheffer A. *D*-charts: quasi-developable mesh segmentation. In: Eurographics 2005. 2005. p. 581–90.
[18] Liu Y, Lai Y, Hu S. Stripification of free-form surfaces with global error bounds for developable approximation. IEEE Transactions on Automation Science and Engineering 2009;6(4):700–9.
[19] Rose K, Sheffer A, Wither J, Cani M, Thibert B. Developable surfaces from arbitrary sketched boundaries. In: Eurographics symposium on geometry processing. 2007. p. 163–72.
[20] Tang K, Wang C. Modeling developable folds on a strip. Journal of Computing and Information Science in Engineering 2005;5(1):35–47.
[21] Wang C. Computing length-preserved free boundary for quasi-developable mesh segmentation. IEEE Transactions on Visualization and Computer Graphics 2008;14(1):25–36.
[22] Wang C, Tang K. Optimal boundary triangulations of an interpolating ruled surface. Journal of Computing and Information Science in Engineering 2005; 5(4):291–301.
[23] Frey W. Modeling buckled developable surfaces by triangulation. Computer-Aided Design 2004;36(4):299–313.
[24] Liu Y, Tang K, Joneja A. Modeling dynamic developable meshes by the Hamilton principle. Computer-Aided Design 2007;39(9):719–31.
[25] Bo P, Wang W. Geodesic-controlled developable surfaces for modeling paper bending. In: Eurographics 2007. 2007. p. 365–74.
[26] LaViola J. Sketch-based interfaces: techniques and applications. In: ACM SIGGRAPH. Course notes 3. 2007.
[27] Ma C, Liu Y, Yang H, Teng D, Wang H, Dai G. Knitsketch: A sketch pad for conceptual design of 2D garment patterns. IEEE Transactions on Automation Science and Engineering 2011;8(2):431–7.
[28] Hershberger J, Snoeyink J. Speeding up the Douglas–Peucker line-simplification algorithm. In: Proc. 5th symp. on data handling. 1992. p. 134–43.
[29] Liu Y, Ma C, Zhang D. Easytoy: A plush toy design system using editable sketch curves. IEEE Computer Graphics & Applications 2011;31(2):49–57.
[30] Piegl L, Tiller W. The NURBS book. 2nd ed. Springer-Verlag; 1997.
[31] Pottmann H, Schiftner A, Bo P, Schmiedhofer H, Wang W, Baldassini N, Wallner J. Freeform surfaces from single curved panels. In: ACM SIGGRAPH 2008. 2008. p. Article No. 76.
[32] Cormen T, Leiserson C, Rivest R. Introduction to algorithms. The MIT Press; 1990.
[33] Edelsbrunner H, Mucke E. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. ACM Transactions on Graphics 1990;9(1):66–104.
[34] Hershberger J. Finding the visibility graph of a simple polygon in time proportional to its size. In: Proc. 3rd annual symposium on computational geometry. 1987. p. 11–20.
[35] McCallum D, Avis D. A linear algorithm for finding the convex hull of a simple polygon. Information Processing Letters 1979;9(5):201–6.
[36] Kirkpatrick D, Seidel R. The ultimate planar convex hull algorithm. SIAM Journal on Computing 1986;15(1):287–99.
[37] Kapoor S, Maheshwari S. Efficiently constructing the visibility graph of a simple polygon with obstacles. SIAM Journal on Computing 2000;30(3): 847–71.
[38] Toussaint G. Quadrangulations of planar sets. In: Algorithms and data structures. LNCS, vol. 955. 1995. p. 218–27.