Quasi-Developable B-Spline Surface Design with Control Rulings

Zi-Xuan Hu\(^1\) (胡梓轩), Peng-Bo Bo\(^1,\*)\ (伯彭波), Senior Member, CCF, and Cai-Ming Zhang\(^2,3\) (张彩明), Senior Member, CCF

\(^1\)School of Computer Science and Technology, Harbin Institute of Technology, Weihai 264209, China
\(^2\)Shandong Co-Innovation Center of Future Intelligent Computing, Yantai 264005, China
\(^3\)Shandong Province Key Laboratory of Digital Media Technology, Shandong University of Finance and Economics Jinan 250061, China

E-mail: 170400110@stu.hit.edu.cn; pbbo@hit.edu.cn; czhang@sdu.edu.cn

Received June 4, 2020; accepted February 10, 2022.

Abstract We propose a method for generating a ruled B-spline surface fitting to a sequence of pre-defined ruling lines and the generated surface is required to be as developable as possible. Specifically, the terminal ruling lines are treated as hard constraints. Different from existing methods that compute a quasi-developable surface from two boundary curves and cannot achieve explicit ruling control, our method controls ruling lines in an intuitive way and serves as an effective tool for computing quasi-developable surfaces from freely-designed rulings. We treat this problem from the point of view of numerical optimization and solve for surfaces meeting the distance error tolerance allowed in applications. The performance and the efficacy of the proposed method are demonstrated by the experiments on a variety of models including an application of the method for the path planning in 5-axis computer numerical control (CNC) flank milling.

Keywords developable surface, surface modeling, B-spline surface, numerical optimization

1 Introduction

Sheet metal and wooden panels are widely-used materials for building ship-hulls and freeform architecture, which possess the physical property of high bendability and low stretchability. The developable surface is a suitable mathematical model for these materials and thus has important industrial applications\(^{[1-3]}\). Developable surfaces have also been investigated in computer numerical control (CNC) flank milling since they possess zero twist at the ruling lines and can be accurately machined with conical cutting tools\(^{[4,5]}\). However, current commercial CAD software does not have flexible and effective capabilities for modeling developable surfaces or quasi-developable surfaces. The study of design methods for developable surfaces has been an active research topic in recent years and a variety of approaches have been proposed\(^{[6-9]}\).

To be compatible with the surface representation in commercial CAD software, the designed surface is required to be represented by a B-spline surface. The most popular method starts with two curves serving as surface boundaries and a (quasi-)developable surface bounded by the curves (or their perturbation curves) is constructed. However, the line connecting the start (or end) points of the two curves is generally not a ruling line on the resulting surface. Therefore, the input curves need to be adjusted through curve extension and the generated surface needs to be trimmed\(^{[10]}\). Fig. 1 shows a simple example of the developable surface design based on boundary curves, which illustrates the necessity of curve extension and surface trimming.

This extension-trimming method has some drawbacks. 1) The developability of the resulting surface largely depends on the shape of the extension curves and it is not clear how to extend the original curves in...
Fig. 1. Developable surface modeling based on given design curves. (a) and (b) show two different perspectives of two initial design curves (thick curves) and the extension line of one design curve (thin curves). (c) Developable surface bounded by two given curves. The red line is the trimming line, and the blue dotted line is one of the rulings to be trimmed.

In this paper, we propose a quasi-developable surface design method based on a sequence of control rulings which are desired to lie on the resulting surface as much as possible. Moreover, we fix the first and the last ruling lines and control the interior shape explicitly using the control rulings. Different from the existing methods which design a surface in the dual space by specifying control planes tangent to the surface [11], our method works directly in the design space and is thus more geometrically intuitive. The main contributions of our work are summarized as follows.

- Our method generates a quasi-developable surface through the direct control of some lines which express the surface shape by serving as approximation targets. This manner provides a geometrically intuitive design tool which cannot be realized with existing methods.

- Our method provides the explicit control of the terminal ruling lines and thus avoids the tedious extension-trimming manner of traditional methods.
• Our method reveals the fact that there is a considerably large space of quasi-developable surfaces taking a given sequence of line segments as approximate rulings.

The rest of this paper is organized as follows. In Section 2, we discuss existing work on developable surface modeling. In Section 3, we propose a method for the developable surface design with control rulings. In Section 4, we propose a method to deal with an additional constraint of fixing one boundary. Section 5 presents some experimental results and discussions. Section 6 discusses the future research directions and concludes the paper.

2 Related Work

Developable surface modeling has been extensively studied and various representations have been considered\cite{1,3,12–18}. Although exact developable surface computation is an interesting mathematical problem, in real applications, only nearly developable surfaces exist for specific design input. In computer animation and simulation, the triangular and quadrilateral meshes are most frequently used (for the work on developable surface modeling based on meshes, please refer to [12–15]). In many CAD applications such as architectural surface design and CNC flank milling, it is desired that the surface can be approximated with a set of developable strips and the surface strips are represented by B-spline/NURBS surfaces\cite{3,16}. Subag and Elber approximated a general NURBS surface using piecewise developable surfaces with a global error bound\cite{17}. Pottmann et al. used the composite developable strip model to obtain free-form surfaces in architecture and manufacturing\cite{1}. Gavrilí et al. proposed a method to improve the developability of a B-spline surface using the property of its Gauss image\cite{18}. In this paper, we concern on the fundamental problem of designing a quasi-developable strip as a B-spline surface, which has wide applications in industry. For example, in the ship-hull design, the hull shape is described by some feature curves, and then the developable surface is constructed between two adjacent feature curves\cite{4}. Pottmann et al. demonstrated the possibility of approximating a freeform surface with a set of developable strips\cite{5}.

Discrete Developable Strips. Tang and Wang simulated the folding process of elastic sheets to approximate a developable surface via the operation of boundary triangulation\cite{19}. Wang et al. formulated the developable triangulation problem as a graph problem and used the Dijkstra algorithm to solve it\cite{20,21}. A local-global method was proposed to improve surface developability by optimizing mesh vertices\cite{7}. The surfaces obtained by these methods are discrete mesh surfaces.

Analytic Computation of Smooth Developable Strips. The smooth developable ruled surfaces are often represented as the Bézier or the B-spline surfaces. It is required that the surface satisfies the nonlinear developability constraint and particular boundary conditions (such as fixing one boundary curve). Some work studies the analytical computation of developable surfaces from the nonlinear developability equations\cite{8,16–26}. Chu studied the degrees of freedom in a developable B-spline surface bounded by two curves\cite{27}. The main issue with analytical methods is that the given inputs are required to lie on exact developable surfaces and this cannot be guaranteed in real applications such as the ship-hull design and the free-form architectural design. A developable surface can also be modelled by assembling a set of smooth surface patches. Triangular Bézier patches are used for creating smooth surfaces with $G^1$ continuity\cite{28} while quadrilateral Bézier patches are used for constructing smooth surfaces with $G^2$ continuity\cite{29}. However, these methods are not capable of constructing a developable surface strictly bounded by given curves. Moreover, the resulting surfaces do not possess explicit ruling lines which provide important guiding information in the real manufacturing process. Bodduuri and Ravani\cite{11} made use of the duality between plane and point geometries for developable surface modeling. However, this method is weak in terms of geometric intuition\cite{11}. Although analytical derivation of exact developable surfaces is mathematically interesting, this method cannot handle freely-designed inputs from which only approximate results can be obtained.

Numerical Computation of Smooth Quasi-Developable Strips. Numerical methods for the quasi-developable surface design have been widely studied in recent years due to their flexibility in handling freely-designed inputs. In this discipline, a quasi-developable surface which meets application error tolerance is computed. Tang et al. proposed an interactive design method and decreased the degree of the constraint equations by introducing auxiliary variables\cite{6}. Pérez and Suárez studied the application of quasi-developable B-spline surfaces in the ship-hull design and used the multi-conic method to modify the given curves to im-
prove surface developability \[3\]. A method has been proposed for computing a developable surface bounded by curves perturbed from original design curves, but the resulting surface is not a B-spline surface \[10\]. Bo et al. proposed a method for constructing a quasi-developable B-spline surface between given B-spline curves, and the resulting surface is strictly interpolated to given boundary curves \[30\]. As we have explained, the design method based on boundary curves is not a reasonable way to control the terminal rulings of the surface.

**Developable Strip Computation Considering Ruling Lines.** Ruling lines are considered as design guidance in some work. Chalfant and Maekawa proposed a quasi-developable design method based on given boundary curves and boundary rulings, and discussed its applications in ship-hull surface modeling \[2\]. Park et al. gave the direction of a set of ruling lines and two endpoints of the boundary curve, and obtained developable surfaces by optimal control \[31\]. Fernandez-Jambrina designed a developable surface by giving one boundary curve and two boundary rulings, but the endpoints of the boundary rulings cannot be specified by the designers \[32\]. Caton and Fernández-Jambrina improved the method in \[32\] by the degree elevation operation and provided the ability of choosing both endpoints of the rulings \[9\]. These methods generate exact developable surfaces and have strict requirements on boundary conditions. Therefore, these methods are not suitable for solving practical problems where the input curves or ruling lines may not contribute to a precise developable surface. In this paper, we propose an intuitive design method based on numerical computation for constructing a quasi-developable surface by specifying some ruling lines freely in the space as control tools.

The proposed method is distinguished from existing methods in that it works with freely-designed control rulings. This provides the designer a tool to define a quasi-developable surface by explicitly specifying and controlling some lines which are desired to be on the design surface. This explicit design manner, i.e., designing a surface by explicit control over some points or curves on the shape, has the benefit of geometric intuitiveness and is a popular way in the CAD community. However, to the best of our knowledge, similar techniques are currently not available for the developable surface design and our method is the first one with explicit control over the surface.

### 3 Developable Surface Design Through Control Rulings

#### 3.1 Problem Formulation

Given a sequence of line segments in space \(L_i, i = 0, ..., K\), with \(Q_i\) and \(P_i\) being the two endpoints of \(L_i\), and \(K + 1\) being the number of given line segments, we aim to generate an as-developable-as-possible surface \(S\) approximating \(L_i, i = 0, ..., K\). \(S\) is in the representation of a B-spline ruled surface

\[
S(t, s) = C_0(t)(1 - s) + C_1(t)s, \quad t, s \in [0, 1],
\]

where \(C_k(t)\), \(k = 0, 1\) is B-spline curves of degree \(d\) whose control points are denoted by \(P_{k, i}, i = 0, ..., N\) with \(N + 1\) being the number of control points. The clamped knot vector \(\Omega = (t_0, ..., t_{N+d+1})\) is used, which has duplicated terminal knots \((t_0 = ... = t_d = 0, t_{N+1} = ... = t_{N+d+1} = 1)\) and uniform internal knots. The ruled surface \(S\) is required to satisfy the developability constraints which are nonlinear functions. Approximating the given line segments with \(S\) requires each line segment \(L_i\) to be close to some ruling line on \(S\). Especially, the first and the last line segments \((L_0, L_K)\) are treated as hard constraints demanding \(L_0\) and \(L_K\) to be identical to \(S(0, s)\) and \(S(1, s)\), respectively. In this way, the input line segments serve as tools for controlling the surface shape and are thus called the control rulings.

Fitting \(S\) to the sequence of line segments \((L_i, i = 0, ..., K)\) can be solved by fitting the curves \(C_0(t)\) and \(C_1(t)\) to the corresponding endpoints of the line segments, i.e., to satisfy \(S(t_i) = Q_i\) and \(S(t_i) = P_i\), \(i = 0, ..., K\), where \(t_i\) is the parametrization of \(L_i\). It is clear that if we have \(C_0(t_i) = Q_i\) and \(C_1(t_i) = P_i\), the ruling line \(S(t_i, s)\) is identical to the input control ruling connecting \(Q_i\) and \(P_i\). Therefore, the problem of developable surface generation can be transformed into the problem of curve fitting, namely, finding two B-spline curves \(C_0(t)\) and \(C_1(t)\) fitting the data points \(Q_i\) and \(P_i\), \(i = 0, ..., K\), respectively. However, different from a general curve fitting problem, the curves \(C_0(t)\) and \(C_1(t)\) are interrelated in developable surface modeling by the following constraints:

- constraint 1: the data points \(Q_i\) and \(P_i\) correspond to the same parameter value \(t_i\);  
- constraint 2: the surface \(S\) bounded by \(C_0(t)\) and \(C_1(t)\), defined by (1), achieves a high degree of developability.
3.2 Algorithm

Because there are generally no exact developable surfaces interpolating arbitrary line sequences, we have to make some relaxations and compute developable surfaces numerically. In our method, we treat the first and the last control rulings as strict interpolation constraints and the interior control rulings as soft constraints. We formulate this problem as an optimization one and obtain the resulting curves $C_0(t)$, $C_1(t)$ and the developable surface $S$ by minimizing an objective function evaluating surface developability.

In order to formulate a function for evaluating developability and other useful properties of the surface, we consider the followings.

**Prime-Dual Formulation of Developability.** The surface $S$ bounded by $C_0(t)$ and $C_1(t)$ being a developable surface requires $S$ to meet the developability constraints which are usually nonlinear functions. A widely-used formulation of developability is

$$\|C_0(t), C_1(t), C_0(t) - C_1(t)\|^2 = 0.$$  

However, this highly non-linear function makes the optimization of surface developability difficult since the solution space contains many local minimizers. In order to simplify the constraint function, Tang et al. introduced additional variables by incorporating the normal function $N(t)$ and obtained

$$\begin{cases} 
C_0'(t) \times N(t) = 0, \\
C_1'(t) \times N(t) = 0, \\
(C_0(t) - C_1(t)) \times N(t) = 0, 
\end{cases} \quad (2)$$

where $N(t)$ is a B-spline function representing the normal field\cite{33}. This form combines the prime and the dual form of the developable surface and leads to stable convergence of optimization. Moreover, owing to the inherent smoothness of a B-spline curve, the normal function $N(t)$ has effect on surface regularization.

However, using a B-spline function for the normal field also means that all normals are linear combinations of the same set of control points and this reduces the degrees of freedom for improving surface developability. Therefore, we adopt independent normals $N_k$ at a set of samplings instead of a smooth B-spline function for the normal field. We thus introduce the following term to evaluate surface developability.

$$F_{\text{Dev}} = \sum_{k=0}^{M} \left( (C_0'(t_k) \times N_k)^2 + (C_1'(t_k) \times N_k)^2 + ((C_0(t_k) - C_1(t_k)) \times N_k)^2 \right), \quad (3)$$

where $t_k$ is a sampling parameter and $M + 1$ is the number of the sampling parameters. $N_k$ is the normal corresponding to $t_k$ which is also a variable in optimization in addition to the interior control points of $C_0(t)$ and $C_1(t)$. Although the number of variables in optimization becomes larger by using (3) than by using (2), the degree of functions becomes much smaller which makes the optimization easier and more stable\cite{10}.

**Surface Regularization.** The number of variables in optimization is the sum of the number of sampling normals ($M + 1$) on surface and the number of interior control points of $C_0(t)$ and $C_1(t)$. A large number of variables in optimization may lead to an unstable optimization process and even an unsatisfactory solution in some cases, which have been observed in previous work\cite{33} on curve and surface fitting. An effective method to deal with this problem is to introduce regularization terms for stabilizing the optimization process in favour of surface fairness. We introduce the following shape-control terms.

$$F_{\text{Energy0}} = \int_0^1 \|C_0''(t)\|^2 dt,$$

$$F_{\text{Energy1}} = \int_0^1 \|C_1''(t)\|^2 dt,$$

$$F_{\text{Width}} = \sum_{j=0}^{M-1} \left( \|C_0(t_j) - C_1(t_j)\|^2 - \|C_0(t_{j+1}) - C_1(t_{j+1})\|^2 \right)^2,$$

where $F_{\text{Energy0}}$ and $F_{\text{Energy1}}$ control the smoothness of $C_0(t)$ and $C_1(t)$, respectively, and $F_{\text{Width}}$ controls the width variation of the surface.

**Interior Control by Control Rulings.** In order to control the interior shape of the surface, the internal control rulings are taken into account for controlling the shape of surface and we define the following closeness term.

$$F_{\text{Closeness}} = \sum_{i=1}^{K-1} \left( \|C_1(t_i) - P_i\|^2 + \|C_0(t_i) - Q_i\|^2 \right)^2,$$

where $t_i$ is the parameter corresponding to $P_i$ and $Q_i$ which is also an unknown variable to solve for. Note that each end control point of both curves is fixed and is not treated as a variable in optimization, which confirms hard interpolation to the specified terminal rulings.

Viewing the interior control points of two curves $C_0(t)$ and $C_1(t)$ as variables in optimization provides a large number of degrees of freedom and enables a
high possibility of finding a surface with a high degree of developability. As we have seen, fitting the developable surface to a set of line segments is equivalent to fitting two curves to the vectors of data points \(Q = (Q_0, ..., Q_K)\) and \(P = (P_0, ..., P_K)\), respectively, with certain constraints. It is well-known that the parametrization of the data points is a key issue in curve fitting. In our specific problem, data parametrization is even more tricky than in ordinary curve fitting because the data points \(Q_i\) and \(P_i\) are required to have the same parameter value as stated in constraint 1. Therefore, the key issue is how to find the optimal parameter value \(t_i\) for \(Q_i\) and \(P_i\). We propose an algorithm which solves for \(C_0(t)\) and \(C_1(t)\) and parameter \(t_i\) through an iterative process. The main steps of this algorithm are described as follows.

1) Step 1: parametrize the rulings to make \(Q_i\) and \(P_i\) have the same parameter value \(t_i\) for \(i = 0, ..., K\), respectively. We employ the centroidal parametrization and obtain the parameter values \(u = (u_0, ..., u_K)\), and \(v = (v_0, ..., v_K)\) for \(Q\) and \(P\), respectively. Then the average value of the parameter values at two endpoints of a ruling is used for the parameter value of both endpoints. Precisely, \(t_i = \frac{1}{2}(u_i + v_i)\) is used as the parameter value for both \(Q_i\) and \(P_i\).

2) Step 2: construct the interpolation B-spline curves \(\tilde{C}_0(t)\) and \(\tilde{C}_1(t)\) satisfying \(\tilde{C}_0(t_j) = Q_j\) and \(\tilde{C}_1(t_j) = P_j\), \(j = 0, ..., K\). Generally, the surface \(S\) bounded by \(C_0(t)\) and \(C_1(t)\) as defined by (1) is not a developable surface and its developability needs to be improved by relaxing the constraint of curve interpolation.

3) Step 3: solve the minimization problem \(F_1 \rightarrow \min\), where \(F_1\) is defined in (4). We obtain two updated curves \(\tilde{C}_0(t)\) and \(\tilde{C}_1(t)\) (the same symbols are used for the curves before and after updating).

\[
F_1 = F_{\text{Dev}} + \lambda_{\text{Closeness}}F_{\text{Closeness}} + \lambda_{\text{Energy}}(F_{\text{Energy0}} + F_{\text{Energy1}}) + \lambda_{\text{Width}}FW_{\text{Width}}.
\]  

In order to provide a termination criterion of the iterative process, we evaluate the developability of the surface \(S\) bounded by the updated curves \(\tilde{C}_0(t)\) and \(\tilde{C}_1(t)\). If the developability requirements are met, or the improvement of developability by this iteration is ignorable, the algorithm can be stopped. Otherwise, we go to step 4.

4) Step 4: update the parameter values of the data points. For the data point \(Q_j\), a parameter value \(u_j\) is obtained, which is the parameter value of its nearest point (foot point) on \(\tilde{C}_0(t)\). Similarly, for the data point \(P_j\), the parameter value \(v_j\) of the nearest point (foot point) on \(\tilde{C}_1(t)\) is computed. For foot point computation, a Newton-like iteration method is used [14]. Then the value \(t_j = \frac{1}{2}(u_j + v_j)\) is used as the updated parameter value for both \(Q_j\) and \(P_j\). Then we go to step 3.

The variables in the iterative optimization process include the vector of all sample normals \(N = (N_0, ..., N_M)\) and the interior control points of \(C_0(t)\) and \(C_1(t)\) (the first and the last control points remain fixed). The algorithm is summarized in Algorithm 1. Note that the auxiliary variables in \(N\) are not returned since they are not used in the definition of the resulting surface. The outputs of the algorithm together with the fixed end points give us the total set of control points, i.e., \((C_{P0}^0, ..., C_{P0}^K)\) and \((C_{P1}^0, ..., C_{P1}^K)\) for \(C_0(t)\) and \(C_1(t)\), respectively. The resulting surface \(S\) is thus obtained by the definition of (1).

An experiment is shown in Fig.3. The surface developability is evaluated on a set of ruling lines by the warp angle which measures the angle between the normals at the end points of a ruling line. Specifically, the warp angle \(\Gamma_i\) associated with parameter \(t_i\) is defined by

\[
\Gamma_i = V(N(t_i, s = 0), N(t_i, s = 1)),
\]

where \(N(t_i, s)\) is the normal at \(S(t_i, s)\) and \(V(\cdot, \cdot)\) is the function calculating the angle between two vectors.

Algorithm 1. Developable Surface Computation from a Sequence of Control Rulings

**Input:** the sequence of developable rulings \(L_i = (Q_i, P_i), i = 0, ..., K\), the weights \(\lambda_{\text{Closeness}}, \lambda_{\text{Energy}}\) and \(\lambda_{\text{Width}}\), the number of the sampling parameters \(M + 1\)

**Output:** the interior control points of \(C_0(t) = CP^0_0, ..., CP^0_K\), the interior control points of \(C_1(t) = CP^1_0, ..., CP^1_K\)

1: Compute the parameter values \(t_i, i = 0, ..., K\) of the data points \(Q_i\) and \(P_i, i = 0, ..., K\), respectively
2: Two B-spline curves are computed as the initialization by solving the linear systems of equations \(C_0(t_i) = Q_i, i = 0, ..., K\) and \(C_1(t_i) = P_i, i = 0, ..., K\), respectively
3: repeat
4: Solve the minimization of (4) for the control points of \(C_0(t)\) and \(C_1(t)\), and the normals \(N_j, j = 0, ..., M\)
5: Update the parameter values \(t_i, i = 0, ..., K\) of the data points \(Q_i\) and \(P_i, i = 0, ..., K\) by projecting the data points to the corresponding curves \(\tilde{C}_0(t)\) and \(\tilde{C}_1(t)\), respectively
6: until the improvement of the surface is smaller than a predefined tolerance
7: return \(CP^0_0, ..., CP^0_K\) and \(CP^1_0, ..., CP^1_K\)
Fig. 3. Developable surface design through control rulings with both curves relaxed. \( \lambda_{\text{Energy}} = 0.000001, \lambda_{\text{Width}} = 0.000001, \) and \( \lambda_{\text{Closeness}} = 1. \) (a) Input control rulings forming a quad strip. (b) Initial interpolation surface. \( (\beta_{\text{Max}}, \beta_{\text{Ave}}) = (28.68, 15.88) \). (c) Resulting surface. \( (\hat{\beta}_{\text{Max}}, \hat{\beta}_{\text{Ave}}) = (3.91, 1.27) \), and \( (D_{\text{Max}}, D_{\text{Ave}}) = (0.025, 0.009) \). (d) The initial boundary curves (dotted curves) and the resulting boundary curves (solid curves) are shown together for comparison. (e) Color coding of warp angles of the initial interpolation surface in (b). (f) Color coding of warp angles of the resulting surface in (c). The convergence behaviour of Algorithm 1 is shown in (g) and (h). (g) Maximum warp angle vs iteration. (h) Average warp angle vs iteration.

Given a set of rulings (Fig. 3(a)), we first construct two initial interpolation B-spline curves \( \hat{C}_0(t), \hat{C}_1(t) \) and the initial interpolation surface \( S \) (Fig. 3(b)). Fig. 3(c) shows the resulting boundary curves and the resulting surface. \( \beta_{\text{Max}} \) and \( \beta_{\text{Ave}} \) represent the maximum and the average warp angle among a set of sampling rulings, respectively. \( D_{\text{Max}} \) and \( D_{\text{Ave}} \) represent the maximum and the average distance, respectively. Within the range of unit 1, the maximum distance from given data points to the resulting curves \( C_0(t) \) and \( C_1(t) \) is 0.025, and the average distance is 0.009, indicating that the surface achieves high degrees of approximation to input line segments. Fig. 3(d) shows the difference between initial interpolation curves (dotted curves) and result boundary curves (solid curves). Fig. 3(e) and Fig. 3(f) show the color coding of warp angles of the surfaces in Fig. 3(b) and Fig. 3(c), respectively. The color coding of each surface is based on the range of its own maximum and minimum warp angles. The convergence behaviour of Algorithm 1 is shown in Fig. 3(g) and Fig. 3(h), which demonstrates the super-linear convergence rate and high stability of the algorithm.

3.3 Designing the Control Rulings

Algorithm 1 depends on a sequence of line segments serving as control rulings of the surface. Basically, the line segments can be specified in the space freely by the designer. However, with arbitrary line segments in space, the distance between the control lines and the resulting developable surface may be large, which is not
desired in the surface design.

To reduce the difference between the original design rulings and the resulting surface, while still achieving a high degree of developability, additional constraints in the design of input rulings should be considered. It has been shown that planar quadrilateral strips can be viewed as the discretization of smooth developable surfaces\cite{14}. Therefore, in the design process of the control rulings, we require that every pair of adjacent rulings should be coplanar. Precisely, if the given rulings are denoted by \(L_i = P_iQ_i, i = 0, ..., K\), we require the quadrilateral \(P_iQ_iQ_{i+1}P_{i+1}\) bounded by two adjacent rulings \(L_i, L_{i+1}\) to be coplanar.

In Figs. 4(a)–4(c), the control rulings are designed arbitrarily and we observe that it is not always guaranteed that a resulting surface with a high degree of developability can meet the original design intention tightly. That is, a compromise between the original design intention of shape and a high degree of developability should be made.

Figs. 4(d)–4(f) give the experimental results with the original control rulings similar to those in Fig. 4(a) which however satisfy the coplanarity constraint. From the experimental results in Figs. 4(d)–4(f), we observe that we can obtain a developable surface better meeting the original design intention with input line segments forming a planar quadrilateral strip. The developability of the resulting surface based on coplanar lines is about an order of magnitude better than that based on non-planar lines and the resulting surface based on coplanar lines achieves good approximation quality to the control rulings.

From these experiments, we can suggest the following procedural way of the control ruling design. Once the \(i\)-th control ruling \(L_i\) is ready, the \((i + 1)\)-th ruling should be defined on the plane containing \(L_i\). Once we have a sequence of line segments, it is easy to adjust the length of each line segment so that the sequence of end points of the line segments forms two fair curves. Note that although this process depends on a sequence of planes, this method is essentially different from the existing methods working with planes tangent to the developable surface. Our method works directly in the design space which has more geometric intuition than

![Fig. 4](image-url)
those working in the dual space. This algorithm of the control ruling design is presented in Algorithm 2. Another option of designing a sequence of coplanar lines is applying the quadrilateral mesh optimization method with nonlinear coplanarity constraints [14].

4 Developable Surface Design with One Fixed Boundary

In some applications, one of the boundary curves of the developable surface is specified and fixed. Given additionally a set of straight line segments serving as ruling lines, we need to construct the other boundary curve to obtain the developable surface $S$. It is interesting to reveal the capability of our surface design method with such an additional constraint which obviously decreases the degrees of freedom in surface design. Without loss of generality, we assume the boundary curve $C_0(t), t \in [0, 1]$ is given and fixed. The control rulings are a sequence of line segments $L_i, i = 0, ..., K$ emanating from sample points on $C_0(t)$. The developable surface to compute is required to satisfy the following constraints: the line segments $L_i$ are identical to $\overline{C_0(t_i)P_i}, i = 0, ..., K$, with $t_0 = 0, t_K = 1$ indicating the terminal interpolation constraints and $P_i$ being data points in the space.

Since $C_0(t)$ is fixed and the endpoints of $C_1(t)$ are fixed to be $P_0$ and $P_K$, the unknown variables are the interior control points of $C_1(t)$, i.e., $P_i, i = 1, ..., K - 1$, in addition to the variable normals. The objective is to make $C_1(t)$ approximate $P_i, i = 1, ..., K - 1$ such that the surface $S$ defined in (1) achieves a high degree of developability. Basically, this problem comes from the problem in Subsection 3.2 with an additional constraint of fixing one boundary curve. This additional constraint greatly simplifies the problem due to the requirement that two endpoints of one ruling should have the same parameter (constraint 1), which means that the parameter of the data point $P_i$ is known to be the parameter $t_i$ of the other point of the ruling line on $C_0(t)$. Therefore, the interior closeness term becomes

$$F_{\text{Interior}} = \sum_{i=1}^{K-1} \| C_1(t_i) - P_i \|^2,$$

where the parameter value $t_i$ is constant, which is equal to the parameter value of the other endpoint of the control ruling line $L_i = \overline{C_0(t_i)P_i}$.

With the parameters $t_i$ being known values, the designed surface depends on the curve $C_1(t)$ which can be computed by minimizing an objective function, without resorting to an iterative procedure for computing the parameter values. The objective function is defined by

$$F_2 = F_{\text{Dev}} + \lambda_{\text{Energy}}F_{\text{Energy}} + \lambda_{\text{Width}}F_{\text{Width}} + \lambda_{\text{Interior}}F_{\text{Interior}}. \quad (5)$$

The algorithm with fixed parameters is presented in Algorithm 3. The resulting B-spline surface defined by (1) is obtained using the outputs of Algorithm 3 and the fixed end points $CP_0$ and $CP_K$.

It is interesting to know whether there is enough degrees of freedom with the additional boundary fixing constraint. We illustrate this by several experiments in Fig.5–Fig.7. Firstly, we assume there are no more control rulings other than the first and the last ones. This case is characterized by $K = 1$ in our definition. The first and the last ruling line can be defined by two additional data points $P_0$ and $P_1$ in space and are denoted by $\overline{C_0(0)P_0}$ and $\overline{C_0(1)P_1}$, respectively. This design model is conceptually similar to the design of a Hermite curve by specifying the end data points and associated tangent vectors.

In this case, the energy term $F_{\text{Energy}}$ and the width variation control term $F_{\text{Width}}$ have an important effect on the interior shape of the resulting surface which is illustrated by the experimental results in Fig.5 and Fig.6, respectively. The boundary curve $C_1(t)$ is initialized by transforming the control points of $C_0(t)$ by rotation, translation and scaling operations such that $C_0(0)$ and

---

**Algorithm 2.** Design Process of Sequence of Control Rulings

**Input:** a target shape in the designer’s mind

**Output:** the sequence of straight line segments in space $L_i = (Q_i, P_i), i = 0, ..., K$

1: The first line segment $L_0$ is defined by specifying two data points $Q_0$ and $P_0$ in the space
2: $i = 0$
3: repeat
4: A plane $\Omega_i$ containing $L_i$ is defined by specifying a line segment $W_i = (A_i, B_i)$ in the space where $A_i$ is a data point on $L_i$
5: The $(i + 1)$-th line segment is defined by specifying two data points on the plane $\Omega_i$
6: $i++$
7: until The current line segment is specified as the last line segment
8: return $L_i = (Q_i, P_i), i = 0, ..., K$
Algorithm 3. Developable Surface Computation from a Sequence of Control Rulings and a Fixed Boundary Curve

**Input:** the sequence of control rulings $L_i = (Q_i, P_i), i = 0, ..., K$, the curve $C_0(t)$ which interpolates the data points $Q_i, i = 0, ..., K$, the weights $\lambda_{\text{Energy}}, \lambda_{\text{Width}}$ and $\lambda_{\text{Interior}}$, the number of the sampling normals $M + 1$

**Output:** the interior control points of $C_1(t)$: $CP^1_0, ..., CP^1_K$.

1. For each data point $P_i, i = 0, ..., K$, the parameter value of the data point $Q_i$ on $C_0(t)$ is used for its parametrization
2. A B-spline curve $C_1(t)$ satisfying $C_1(t_i) = P_i, i = 0, ..., K$ is computed by solving the linear system of equations
3. A set of normals $N_k, k = 0, ..., M$ are computed on the B-spline surface $S$ bounded by $C_0(t)$ and $C_1(t)$, defined by (1)
4. Solve the minimization of (5) for the control points of $C_1(t)$ and the normals $N_j, j = 0, ..., M$
5. return $CP^1_0, ..., CP^1_K$.

$C_0(1)$ coincide with $P_0$ and $P_1$, respectively. The developability of the surface is evaluated by the warps between the normals at both ends of the ruling lines.

Observing the results in Fig.5, we find that the energy term $F_{\text{Energy}}$ can efficiently control the shape of the boundary curve and avoid self-overlapping of the surface, without sacrificing the developability of the surface. In fact, the resulting surface achieves a higher degree of developability than the resulting surfaces without the energy term. This is because the energy term $F_{\text{Energy}}$ leads to a finer shape of the solution space and consequently a stable optimization process which avoids unsatisfactory local minimums. Observing the results shown in Fig.6, the width variation control term $F_{\text{Width}}$ makes the resulting surface have a smooth variation of the width of ruling lines with improved surface developability. The reason for better developability is the same as that of the energy term $F_{\text{Energy}}$.

When $K > 1$, the interior rulings other than the end ones can be used to control the interior shape of the resulting surface, which also introduce more constraints and further reduce the number of the degrees of freedom. The experiments in Fig.7 show the difference between the results with and without the constraint of interior rulings. The upper boundary curve $C_0(t)$ in Fig.7(a) is the given curve which is fixed. Fig.7(b) and Fig.7(c) show the developable surfaces without and with the constraint of interior rulings, respectively. The color coding in Fig.7 indicates the magnitude of the warp angle. We also measure the distances from the endpoints of rulings to the final boundary curve.

We conclude that in the case of fixing one boundary curve, surface developability has to be given up to
some extent in order to comply with the interior control rulings. This implies that only a limited design space exists and the boundary interpolation constraint should be relaxed to increase the possibility of finding a surface with a high degree of developability.

5 Method Evaluation and Discussions

In this section, we show more experiments and discuss the capability and flexibility of our proposed method. Surface developability is evaluated by measuring the warp angles at a set of ruling lines. Existing work has shown that the practical requirements of the ship-hull design can be met when the maximum warp angle is less than $6^\circ$ [3].

5.1 Implementation Details

The proposed algorithms have been implemented with C++ on the Microsoft Visual Studio platform. All experiments are performed on a laptop with 2.6 GHz Intel Core i7 CPU and 16 GB 2400 MHz DDR4 memory.

In our experiments, the boundary curves are cubic B-spline curves. The number of samplings $(M + 1)$ on the surface is 100 which is used to evaluate surface developability during optimization. Theoretically, $M + 1$ should depend on the number of control points of the B-spline surface and a very small value of $M + 1$ possibly makes some control points of the surface unconstrained and leads to a bad result. Practically, as long as $M + 1$ is large enough, our method works very well. The minimization of the nonlinear functions (4) and (5) is solved with the L-BFGS algorithm [33]. The running time ranges from 16 s to 60 s for the experiments in this paper.

The weight of each term in the objective function is determined empirically. To unify the weight settings in the objective function, all models are scaled into a unit box. We use the same weights in one algorithm for all experiments and obtain satisfactory results. The weight settings $(\lambda_{\text{Energy}}, \lambda_{\text{Width}}, \lambda_{\text{Closeness}})$, the maximum and the average warp angles $(\beta_{\text{Max}}, \beta_{\text{Ave}})$ of the resulting surfaces and the distance errors $(D_{\text{Max}}, D_{\text{Ave}})$ of the experiments in this paper are given in Table 1.

5.2 Experiments on Relaxing Both Boundary Curves

In Fig. 8, we give a comparison of the performance of fixing one boundary curve (Algorithm 3 in Section 4) and relaxing both curves (Algorithm 1 in Subsection 3.2), with the same input control rulings, the
Table 1. Hyperparameter Settings and Results of the Experiments in this Paper

| Figure | $\lambda_{\text{Energy}}$ | $\lambda_{\text{Width}}$ | $\lambda_{\text{Closeness}}$ | ($\beta^\text{pre}_{\text{Max}}, \beta^\text{pre}_{\text{Ave}}$) | ($\beta^\text{ref}_{\text{Max}}, \beta^\text{ref}_{\text{Ave}}$) | ($D^\text{Max}, D^\text{Ave}$) | Algorithm |
|--------|--------------------------|--------------------------|-------------------------------|----------------------------------|----------------------------------|-----------------------------|-----------|
| Fig.3  | 0.00001                  | 0.00001                  | 1.00000                       | (28.68, 15.88)                  | (3.91, 1.27)                     | (0.025, 0.009)               | Algorithm 1 |
| Fig.8(d)| 0.00001                  | 0.00001                  | 1.00000                       | (67.52, 16.81)                  | (1.28, 0.28)                     | (0.012, 0.005)               | Algorithm 1 |
| Fig.9  | 0.00001                  | 0.00001                  | 1.00000                       | (71.78, 16.56)                  | (1.93, 0.48)                     | (0.016, 0.006)               | Algorithm 1 |
| Fig.10(a)| 0.00100                  | 0.00001                  | NA                            | (20.63, 4.86)                   | (2.88, 0.70)                     | NA                          | Algorithm 3 |
| Fig.10(c)| 0.00100                  | 0.00001                  | NA                            | (90.00, 16.37)                  | (3.95, 0.89)                     | NA                          | Algorithm 3 |
| Fig.10(e)| 0.00100                  | 0.00001                  | NA                            | (25.23, 5.36)                   | (3.92, 0.98)                     | NA                          | Algorithm 3 |
| Fig.10(g)| 0.00100                  | 0.00001                  | NA                            | (28.14, 5.90)                   | (3.91, 0.87)                     | NA                          | Algorithm 3 |
| Fig.10(i)| 0.00100                  | 0.00001                  | NA                            | (39.69, 20.59)                  | (4.00, 1.06)                     | NA                          | Algorithm 3 |
| Fig.10(k)| 0.00100                  | 0.00001                  | NA                            | (20.50, 4.92)                   | (2.45, 0.59)                     | NA                          | Algorithm 3 |
| Fig.11(d)| 0.00001                  | 0.00001                  | 1.00000                       | (38.68, 7.52)                   | (0.61, 0.18)                     | (0.029, 0.011)               | Algorithm 1 |
| Fig.11(e)| NA                      | NA                      | NA                            | (38.68, 7.52)                   | (12.18, 1.11)                    | (0.019, 0.006)               | Algorithm 1 |
| Fig.11(f)| NA                      | NA                      | NA                            | (38.68, 7.52)                   | (1.39, 0.22)                     | (0.043, 0.015)               | Algorithm 1 |
| Fig.12(c)| 0.00001                  | 0.10000                  | 0.10000                       | (1.70, 1.32)                    | (0.03, 0.01)                     | (0.020, 0.010)               | Algorithm 1 |
| Fig.12(e)| 0.00001                  | 0.10000                  | 0.10000                       | (1.29, 1.00)                    | (0.02, 0.004)                    | (0.030, 0.010)               | Algorithm 1 |
| Fig.12(g)| 0.00001                  | 0.10000                  | 0.10000                       | (2.72, 2.22)                    | (0.09, 0.02)                     | (0.050, 0.030)               | Algorithm 1 |

Note: ($\beta^\text{pre}_{\text{Max}}, \beta^\text{pre}_{\text{Ave}}$) $\rightarrow$ ($\beta^\text{ref}_{\text{Max}}, \beta^\text{ref}_{\text{Ave}}$) indicates the change of the maximum and the average warp angles. NA: not available.

Fig.8. Comparison between Algorithm 3 and Algorithm 1. (d)-(f) show the results with the upper curve fixed. (g)-(i) show the results with both curves relaxed. The same weights $\lambda_{\text{Energy}} = 0.00001$, $\lambda_{\text{Width}} = 0.00001$, $\lambda_{\text{Interior}}/\lambda_{\text{Closeness}} = 1$ are used for all experiments. (a) Input control rulings. (b) Initial surface ($\beta^\text{ref}_{\text{Max}}, \beta^\text{ref}_{\text{Ave}}$) = (67.52, 16.81). (c) Comparison between resulting boundary curves with both curves relaxed (dotted curves) and one curve fixed (solid curves). (d) Resulting surface using the algorithm of fixing one curve with the surface in (b) as an initialization; ($\beta^\text{ref}_{\text{Max}}, \beta^\text{ref}_{\text{Ave}}$) = (53.74, 9.54), ($D^\text{Max}, D^\text{Ave}$) = (0.014, 0.006). (e) Color coding of warp angles on the surface in (d). (f) Comparison between the initial boundary curves (dotted curves below) and the resulting boundary curves (solid curves below). (g) Resulting surface using the method relaxing both curves with the surface in (b) as the initialization; ($\beta^\text{ref}_{\text{Max}}, \beta^\text{ref}_{\text{Ave}}$) = (1.28, 0.28), ($D^\text{Max}, D^\text{Ave}$) = (0.012, 0.005). (h) Color coding of warp angles on the surface in (g). (i) Comparison between initial boundary curves (dotted curves) and result boundary curves (solid curves).
same weight settings and the same termination conditions of the optimization. The results are shown in Fig. 8. For this experiment, adjacent lines in the input are coplanar as shown in Fig. 8(a). Using the given weight setting, the algorithm fixing one boundary curve makes only a small improvement on surface developability. The surface developability can be further improved by decreasing $\lambda_{\text{interior}}$, but it will also decrease the fitting quality. For the same input and the same initial curves, the method relaxing both boundary curves enables the resulting surface to achieve a much higher degree of developability. Moreover, the distance errors between the resulting surface and the given rulings are also smaller.

This experiment demonstrates that a tiny perturbation of control rulings provides a larger solution space of developable surfaces than fixing boundary curves. The optimization algorithm is thus able to find a much higher developability of the resulting surface without sacrificing the fidelity to original design intention. Since in real applications, a particular level of distance errors of the surface is always reasonable, the proposed method of curve perturbation is a practically useful method for generating developable surfaces with a high degree of developability from control rulings.

Fig. 9 shows a highly-curved shape where the input control rulings obey the coplanarity property as described in Subsection 3.3. We observe that even the initial control rulings form a discrete developable surface, the B-spline surface bounded by the direct interpolation curves using the averaged centripedal parametrization is far from being developable. The warp angles are usually $30^\circ$--$70^\circ$. Note that the method in [3] for constructing quasi-developable surfaces is similar to our initialization method in that both methods generate a B-spline surface by parametrizing the sequence of ruling lines forming a discrete developable surface. The bad developability observed in our experiments is a demonstration of the necessity of computing a B-spline surface directly, instead of computing a sequence of developable lines.

With the initial B-spline surfaces interpolating the control rulings, Algorithm 1 gives a surface whose maximum warp angle is less than $1^\circ$ in most cases, which is
fairly good considering application requirements. This is achieved by introducing a small fitting error which is also acceptable in industrial applications such as ship hull design and architecture surface design.

5.3 Experiments on Fixing One Boundary Curve

Algorithm 3 in Section 4 is useful when fixing one boundary of the surface is a hard rule, especially when there are no interior control rulings. This has been demonstrated by the results in Figs.5–7.

To fully investigate the flexibility of our method with one fixed boundary curve, we evaluate the resulting surfaces computed from various terminal rulings and the same fixed boundary curve \( C_0(t) \). The experimental results in Fig.10 demonstrate that by fixing one boundary curve and two terminal rulings, there are still enough degrees of freedom for generating a high degree of developability, and the interior shape can be nicely controlled with our shape regularization terms. This provides the users with a developable surface design tool by modifying the terminal ruling lines which act as hard interpolation constraints.

5.4 Comparisons with Existing Methods

Existing methods generate quasi-developable surfaces from boundary curves and cannot be directly used for fitting a sequence of control ruling lines. Fortu-

Fig.10. Results from various terminal rulings and the same fixed boundary curve \( C_0(t) \). (a)(c)(e)(g)(i)(k) show the initial surfaces and (b)(d)(f)(h)(j)(l) show corresponding resulting surfaces. The same weights to function terms are used: \( \lambda_{\text{Energy}} = 0.001 \), \( \lambda_{\text{Width}} = 0.00001 \). (a) Initial surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (20.63, 4.86)\). (b) Resulting surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (2.88, 0.70)\). (c) Initial surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (90.00, 16.37)\). (d) Resulting surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (3.95, 0.89)\). (e) Initial surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (25.23, 5.36)\). (f) Resulting surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (3.92, 0.98)\). (g) Initial surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (28.14, 5.90)\). (h) Resulting surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (3.91, 0.87)\). (i) Initial surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (39.69, 20.59)\). (j) Resulting surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (4.00, 1.06)\). (k) Initial surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (20.50, 4.92)\). (l) Resulting surface \((\beta_{\text{Max}}, \beta_{\text{Avg}}) = (2.45, 0.59)\). The color coding of each surface is based on the range of its own maximum and minimum warp angles.
nately, curve interpolation to a sequence of data points is a well-studied problem and smooth curves interpolating the end points of the control rulings are easily computed\cite{35}. Then existing developable surface modelling methods working with boundary curves can be employed\cite{3,10}. We compare our method with two existing methods which compute quasi-developable surfaces from boundary curves and demonstrate the advantages of our method. The first method (method 1) proposed in \cite{3} computes a discrete representation by searching for a sequence of lines connecting the input boundary curves which are locally developable. A smooth surface is then computed by the curve interpolation method\cite{35}. The second method (method 2) proposed in \cite{10} solves for a smooth surface which maximizes the developability of the surface with its boundaries approximating the input curves. The results are given in Fig.11. The results by all methods have similar distance errors and the values can be found in Table 1. From the color coding of warp angles of the resulting surfaces, we observe that our method can generate a surface with a high degree of developability.

The resulting surface via method 1 has a small region with bad developability, although most parts of the surface are highly developable. This is caused by the discrete nature of this method and a larger sampling density may improve the result which is however highly time-consuming. Moreover, even if the discrete
lines are developable, the developability of the resulting continuous surface is not guaranteed because the results also depend on the parametrization of the discrete lines. We also notice the unfairness of the interpolation curves because curve fairness is not considered in this method. The resulting surface via method 2 is good in terms of developability, which, however, does not preserve the terminal control rulings. Note that although method 1 computes interpolation curves, it has zero distance only at the interpolation positions and approximation errors still exist for the remaining parts.

5.5 More Examples

In this subsection, we show more experimental results. Fig. 9 shows a curved model for which the proposed method also works well. To test the performance of our method with real industrial data, we apply our method to compute quasi-developable strips for path planning in 5-axis CNC flank milling. It is well-known that a model can be accurately machined with conical tools if the surface is developable [4]. Because most models in industry are non-developable surfaces, it is highly demanding that a surface can be approximated with developable strips with a high precision. As we have discussed in Section 1, it is often much desired that the terminal rulings of the developable surface can be explicitly specified and our method provides a useful tool for that purpose. Fig. 12 shows three quasi-developable strips on the blisk surface computed using our method for which the control rulings are interactively specified by the user. The experiments show that our method is capable of generating surface strips which meet the manufacturing tolerance in CNC milling. This experiment shows the potential application of our method in the path planning for CNC flank milling.

6 Conclusions

An intuitive design method was proposed to construct a quasi-developable B-spline surface through a sequence of specified rulings. Our method is geometrically intuitive because the surface is directly controlled by the control rulings. This design process is similar to the curve design through control points. Our approach has a distinctive advantage over existing algorithms in that the terminal ruling lines of the resulting surface are specifically defined. The disadvantage of our method is that the weight settings of each term in the objective

![Fig. 12. Results on a blisk surface. (a) Part of a blisk model. (b) Surface for test. (c) Resulting strip 1. (d) Color coding of warp angles of strip 1. (e) Resulting strip 2. (f) Color coding of warp angles of strip 2. (g) Resulting strip 3. (h) Color coding of warp angles of strip 3.](image-url)
function cannot be done automatically. Another limitation of our method is that it is not efficient enough for interactive design and efficiency improvement should be further investigated in the future. We are also interested in the automatic generation of control rulings on a target surface which can lead to quasi-developable surfaces with a high approximation accuracy. This method will make our approach a useful tool for path planning in CNC flank milling.

References

[1] Pottmann H, Schiftner A, Bo P B et al. Freeform surfaces from single curved panels. ACM Trans. Graph, 2008, 27(3): Article No. 76. DOI: 10.1145/1360612.1360675.

[2] Chalfant J S, Maekawa T. Design for manufacturing using B-spline developable surfaces. Journal of Ship Research, 1998, 42(03): 207-215. DOI: 10.5057/jsr.1998.42.3.207.

[3] Pérez F, Suárez J A. Quasi-developable B-spline surfaces in ship hull design. Computer-Aided Design, 2007, 39(10): 853-862. DOI: 10.1016/j.cad.2007.04.004.

[4] Chu C H, Chen J T. Tool path planning for five-axis flank milling with developable surface approximation. The International Journal of Advanced Manufacturing Technology, 2006, 29(7-8): 707-713. DOI: 10.1007/s00170-005-2564-6.

[5] Calleja A, Bo P B, González H, Bartoň M, de L Wallace L N L. Highly accurate 5-axis flank CNC machining with conical tools. The International Journal of Advanced Manufacturing Technology, 2018, 97(5/6/7/8): 1605-1615. DOI: 10.1007/s00170-018-2033-7.

[6] Tang C C, Bo P B, Wallner J, Pottmann H. Interactive design of developable surfaces. ACM Trans. Graph, 2016, 35(2): Article No. 12. DOI: 10.1145/2832906.

[7] Chen M, Tang K, Janeja A. Design of developable interpolating strips. Computer-Aided Design and Applications, 2011, 8(4): 557-570. DOI: 10.3722/CADAPS.2011.557-570.

[8] Li C, Zhu C. Designing developable C-Bézier surface with shape parameters. Mathematics, 2020, 8(3): Article No. 402. DOI: 10.3390/math8030402.

[9] Cantón A, Fernández-Jambrina L. Interpolation of a spline developable surface between a curve and two rulings. Frontiers of Information Technology & Electronic Engineering, 2015, 16(3): 173-190. DOI: 10.1631/FITEE.140210.

[10] Bo P B, Zheng Y J, Jia X H, Zhang C C. Multi-strip smooth developable surfaces from sparse design curves. Computer-Aided Design, 2019, 114: 1-12. DOI: 10.1016/j.cad.2019.05.001.

[11] Boddu P M C, Ravani B. Design of developable surfaces using duality between plane and point geometries. Computer-Aided Design, 1993, 25(10): 621-632. DOI: 10.1016/0010-4485(93)90017-I.

[12] English E, Bridson R. Animating developable surfaces using nonconforming elements. ACM Trans. Graph, 2008, 27(3): Article No. 66. DOI: 10.1145/1360612.1360665.

[13] Solomon J, Vouga E, Wardetzky M, Grinspun E. Flexible developable surfaces. Computer Graphics Forum, 2012, 31(5): 1567-1576. DOI: 10.1111/j.1467-8659.2012.03162.x.

[14] Liu Y, Pottmann H, Wallner J, Yang Y L, Wang W. Geometric modeling with conical meshes and developable surfaces. ACM Trans. Graph, 2006, 25(3): 681-689. DOI: 10.1145/1141911.1141941.

[15] Liu Y J, Tang K, Gong W Y, Wu T R. Industrial design using interpolatory discrete developable surfaces. Computer-Aided Design, 2011, 43(9): 1089-1098. DOI: 10.1016/j.cad.2011.06.001.

[16] Oetter R, Barry C D, Duffy B, Welter J. Block construction of small ships and boats through use of developable panels. Journal of Ship Production, 2002, 18(02): 65-72. DOI: 10.5057/jsap.2002.18.2.65.

[17] Subag J, Elber G. Piecewise developable surface approximation of general NRURBS surfaces with global error bounds. In Proc. the 4th International Conference on Geometric Modeling and Processing, July 2006, pp.143-156. DOI: 10.1007/11802914_11.

[18] Gavriil K, Schiftner A, Pottmann H. Optimizing B-spline surfaces for developability and paneling architectural freeform surfaces. Computer-Aided Design, 2019, 111: 29-43. DOI: 10.1016/j.cad.2019.01.006.

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

[20] Wang C L, Tang K. Optimal boundary triangulations of an interpolating ruled surface. Journal of Computing and Information Science in Engineering, 2005, 5(4): 291-301. DOI: 10.1115/1.2032580.

[21] Tang K, Chen M. Quasi-developable mesh surface interpolation via mesh deformation. IEEE Trans. Visual Comput. Graphics, 2009, 15(3): 518-528. DOI: 10.1109/TVCG.2008.192.

[22] Pottmann H, Farin G. Developable rational Bézier and B-spline surfaces. Computer Aided Geometric Design, 1995, 12(5): 513-531. DOI: 10.1016/0167-8396(94)00031-M.

[23] Chen H Y, Lee I K, Leopoldseder S, Pottmann H, Rundrup T, Wallner J. On surface approximation using developable surfaces. Graphical Models and Image Processing, 1999, 61(2): 110-124. DOI: 10.1006/gmip.1999.0487.

[24] Aumann G. A simple algorithm for designing developable Bézier surfaces. Computer Aided Geometric Design, 2003, 20(8/9): 601-619. DOI: 10.1016/j.cagd.2003.07.001.

[25] Aumann G. Degree elevation and developable Bézier surfaces. Computer Aided Geometric Design, 2004, 21(7): 661-670. DOI: 10.1016/j.cagd.2004.04.007.

[26] Hu G, Wu J. Generalized quartic H-Bézier curves: Construction and application to developable surfaces. Advances in Engineering Software, 2019, 138: Article No. 102723. DOI: 10.1016/j.advengsoft.2019.102723.

[27] Chu C H. Geometric design of uniform developable B-spline surfaces: Constraints and degrees of freedom. Journal of Industrial and Production Engineering, 2013, 30(5): 291-295. DOI: 10.1080/21681015.2013.826788.

[28] Chen M, Tang K. Quasi-developable surface modeling of contours with curved triangular patches. Computers & Graphics, 2013, 37(7): 851-861. DOI: 10.1016/j.cag.2013.05.002.

[29] Chen M, Tang K. $\mathcal{G}^2$ quasi-developable Bezier surface interpolation of two space curves. Computer-Aided Design, 2013, 45(11): 1365-1377. DOI: 10.1016/j.cad.2013.06.009.
Bo P B, Zheng Y J, Chu D H, Zhang C C. As-developable-as-possible B-spline surface interpolation to B-spline curves. Computer Aided Geometric Design, 2020, 79: Article No. 101863. DOI: 10.1016/j.cagd.2020.101863.

Park F C, Yu J, Chun C, Ravani B. Design of developable surfaces using optimal control. Journal of Mechanical Design, 2002, 124(4): 602-608. DOI: 10.1115/1.1515790.

Fernández-Jambrina L. B-spline control nets for developable surfaces. Computer Aided Geometric Design, 2007, 24(4): 189-199. DOI: 10.1016/j.cagd.2007.03.001.

Zheng W N, Bo P B, Liu Y, Wang W W. Fast B-spline curve fitting by L-BFGS. Computer Aided Geometric Design, 2012, 29(7): 448-462. DOI: 10.1016/j.cagd.2012.03.004.

Hoschek J, Lasser D. Fundamentals of Computer Aided Geometric Design. Peters, 1993.

Lim C G. A universal parametrization in B-spline curve and surface interpolation. Computer Aided Geometric Design, 1999, 16(5): 407-422. DOI: 10.1016/S0167-8396(99)00019-2.

Zi-Xuan Hu is an undergraduate student at the School of Computer Science and Technology, Harbin Institute of Technology, Weihai. His main research interests include geometric modeling and computer graphics.

Peng-Bo Bo is a professor and doctoral supervisor at the School of Computer Science and Technology, Harbin Institute of Technology, Weihai. He received his B.S. and M.E. degrees in computer science at the School of Computer Science, Shandong University, Jinan, in 2001 and 2004, respectively, and his Ph.D. degree in computer science from the University of Hong Kong, Hong Kong, in 2010. His research interests include computer graphics, 3D visual computing and CNC machining.

Cai-Ming Zhang is a professor and doctoral supervisor of the School of the Software at Shandong University, Jinan. He received his B.S. and M.E. degrees in computer science at the School of Computer Science, Shandong University, Jinan, in 1982 and 1984, respectively, and his Dr. Eng. degree in computer science from the Tokyo Institute of Technology, Tokyo, in 1994. From 1997 to 2000, Dr. Zhang had held a visiting position at the University of Kentucky, Lexington. His research interests include CAGD (computer aided geometric design), CG (computer graphics), information visualization and medical image processing.