Design of developable surface via CSA-based modification of boundary curves

Huanxin CAO*, Hongchan ZHENG* and Gang HU**

*Northwestern Polytechnical University
1 Dongxiang Road, Changan District, Xi’an 710129, China
E-mail: huanxincao@sina.cn
**Xi’an University of Technology
5 South Jinhua Road, Xincheng District, Xi’an 710054, China

Received: 22 May 2020; Revised: 17 August 2020; Accepted: 6 October 2020

Abstract
In this paper, we present a new method for generating high-accuracy developable surface from two design curves via crow search algorithm (CSA) based modification of design curves. For achieving high-accuracy developability, we allow perturbation of the control points of design curves within a allowed bound to expand the solution space of developable surfaces, and search within the allowed perturbation bound for the set of optimal control points with which the resultant surface has the highest degree of developability via CSA. As a result, design curves are automatically and intelligently modified when that is necessary for achieving a high-accuracy developability. Modeling examples demonstrate that the method is effective and easy to implement to obtain high-accuracy developables.

Keywords: Developable surface, Shape optimization, B-spline curve, Metaheuristic algorithm, CSA

1. Introduction

Developable surface can be unfolded or developed onto a complete plane without tearing or stretching. Conversely, a sheet material (e.g. metal plate or plywood) can be shaped into a developable surface by simply bending the material uniaxially along the rulings (Do Carmo, 1976). Owing to this significant property, developable surfaces find extensive application in the appearance design of products that use inextensible materials like leather, sheet metal and cloths. Thus how to effectively design developables to satisfy practical needs has become an important research subject in CAD/CAM.

There are many works on the design of developable surfaces. These methods include point geometric representation (Lang et al., 1992; Chu et al., 2002), line and plane geometric representation (Bodduluri et al., 1993; Zhou et al., 2013; Hu et al., 2017a; Hu et al., 2017b) and the methods for constructing developables through a given characteristic curve, e.g., geodesic curve (Bo et al., 2007), line of curvature (Li et al., 2013), asymptotes (Wang et al., 2019).

Due to the complexity of developability conditions, there are very limited shape control and design freedom left for users in the previous methods and these methods are hard to use for the average designer, which greatly limits user creativity. More significantly, in practical engineering, it is usually impossible to find a surface which not only meets engineering requirements but also has perfect developability, while small developability error is acceptable in most fields. Therefore methods for designing quasi-developable surfaces are drawing more and more attention.

There are extensive works on the design of quasi-developable surfaces. Among these efforts, the construction of quasi-developable strips between two design curves is an important part, because designers (such as in the field of ship hull, shoe and garment, etc.) usually start with sparse design curves which generally depict their aimed shapes, and create quasi-developable strip to interpolate the design curves (Jung et al., 2015; Pérez et al., 2007). Some scholars concentrated on finding bridge lines between design curves which are assembled together in terms of developability (Wang et al., 2005; Tang et al., 2005; Liu et al., 2011; Chen et al., 2011a). These discrete surfaces are usually not smooth enough and are not applicable to applications where smoothness are required. Then (Chen et al., 2011b; Chen et al., 2013) presented methods to generate quasi-developable strips with $G^1$ or $G^2$ continuity from two design curves. The generated quasi-developable strips are composed of many Bézier patches between two design curves and thus are usually obtained through complicated computation for achieving the continuity conditions between these patches. More
importantly, because these methods do not have the ability of improving surface developability automatically through modifying original design curves when that is necessary, so if the original design curves are not appropriate, it is impossible to generate a quasi-developable strip by these methods.

Owing to the rapid increase of computing power in recent years, nature-inspired metaheuristic algorithms which have the characteristics of self-organization, self-adaptation and self-learning have long been used to tackle difficult problems for the advantages of high robustness, high efficiency and convenience. As a new population-based metaheuristic optimizer, crow search algorithm (CSA) is user-friendly and easy to implement because it has simple concept and has only two parameters that need to be tuned by users (Askarzadeh, 2016). It has been shown that CSA usually has better performance in search efficiency and in finding promising results compared with many conventional search methods.

In this paper, we present a new method for generating quasi-developable strip from two design curves in B-spline form. For achieving high-accuracy developability, we allow perturbation of the control points of design curves within a allowed perturbation bound to expand the solution space of developable surfaces. We formulate the generation of quasi-developable surface as a geometric optimization problem where the objective function is defined as the developability degree of the surface and search within the allowed perturbation bound for the set of optimal control points which minimizes the objective function. Thus the original design curves can be modified automatically when that is necessary for achieving a high-accuracy developability.

The article is organized as follows. Section 2 gives some basic knowledge of B-spline curves and developable surfaces. Crow search algorithm (CSA) is introduced in Section 3. Sections 4 proposes the method for generating quasi-developable strip by using CSA to search within the allowed perturbation bound for the set of control points with which the resultant surface has the highest degree of developability. Some simulation results are given in Sections 5, followed a conclusion in Section 6.

2. Preliminaries

2.1 B-spline curves

A \(p\)-degree B-spline curve is defined as (Piegl et al., 1997)

\[
p(t) = \sum_{i=0}^{n} N_{i,p}(t)P_i, \quad t_0 \leq t \leq t_{n+p+1},
\]

(1)

where \(\{P_i\}_{i=0}^{n}\) are control points and \(\{N_{i,p}(t)\}_{i=0}^{n}\) are the B-spline basis functions defined on a knot list \(t = \{t_0, t_1, \ldots, t_{n+p+1}\}\).

The derivative of the B-spline curve is

\[
p'(t) = \sum_{i=0}^{n} N'_{i,p}(t)P_i, \quad t_0 \leq t \leq t_{n+p+1},
\]

(2)

where \(N'_{i,p}(t)\) is the derivative of the basis function \(N_{i,p}(t)\) with respect to \(t\).

For fixed \(t\), the point \(p(t)\) is obtained by a linear combination of the control points \(\{P_i\}_{i=0}^{n}\). Likewise, for fixed \(t\), the derivative \(p'(t)\) of the B-spline curve is also obtained by a linear combination of the control points \(\{P_i\}_{i=0}^{n}\).

2.2 Developable surfaces

Developable surfaces are a subset of ruled surfaces with vanishing Gauss curvature. In practical application, a ruled surface is usually expressed by its two boundary curves \(a(t)\) and \(b(t)\) as follows

\[
R(t,v) = (1-v)a(t) + vb(t), \quad v \in [0, 1],
\]

(3)

Where the line segment \(L_t(v) = (1-v)a(t) + vb(t)\) for fixed \(t\) is called a ruling of the ruled surface.

A ruled surface is developable if it has constant tangent plane along any of its ruling. Thus the following theorem can be derived.

**Theorem 1.** The necessary and sufficient condition that the ruled surface in Eqn.(3) is developable is
\[(\mathbf{a}'(t), \mathbf{a}(t) - \mathbf{b}(t), \mathbf{b}'(t)) = 0\]  
\hspace{1cm} (4)

for all \( t \), where \( (\cdot, \cdot, \cdot) \) means the mixed product of three vectors.

3. Crow search algorithm

Crows are known as the smartest birds and their intelligence is embodied in many aspects. For example, they can use tools, memorize the positions where they had hidden their food up to several months later, observe the action of other birds and approach other birds’ hiding places to steal food.

Crow search algorithm (CSA) is a new user-friendly (simple concept and easy implementation) global optimization algorithm based on the social behavior of crows flocking. In comparison with many classic search algorithms involved in (Askarzadeh, 2016), CSA enables us to obtain more promising results in solving complicated multidimensional optimization problems with less computation time.

The principles of CSA are listed as follows:
- Crows live in a flock.
- Crows memorize their food’s hiding place.
- Crows follow each other to do thievery.
- Crows protect their food from being found by a probability, which is called awareness probability here.

CSA implementation for optimization

Step 1. Initialize Problem and Adjustable Parameters: (1) Define the optimization problem in which the objective function is expressed as

\[ f(x_1, x_2, \ldots, x_d), \]

where \( x_1, x_2, \ldots, x_d \) are decision variables, \( d \) is the number of decision variables and \( f \) is the fitness (or objective) function which can accurately express the goodness of a solution in a single number.

(2) Provide a reasonable value range for each decision variable in which we search for the optimal solution. Here we denote the range as \( X_{\text{min} k} \) and \( X_{\text{max} k} \) for \( k \). Then the adjustable parameters of CSA (flock size (\( N \)), maximum number of iterations (\( \text{iter}_{\text{max}} \)), flight length (\( f^{\text{iter}} \)) and awareness probability (\( AP^{\text{iter}} \)) are valued.

Step 2. Initialize Positions and Memory of Crows: In order to make uniform search in the initial stages, each crow \( x^i \) begins at a random position in the solution space denoting a feasible solution of the problem. The positions of the crows are expressed as

\[
\text{Crows} = \begin{bmatrix}
x_1^1 & x_2^1 & x_3^1 & \cdots & x_d^1 \\
x_1^2 & x_2^2 & x_3^2 & \cdots & x_d^2 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
x_1^N & x_2^N & x_3^N & \cdots & x_d^N
\end{bmatrix}
\]

The memory of each crow is used to record the position where the crow has hidden its food, that is, the best place the crow has gone through. At the initial iteration, it is assumed that the crows have hidden their foods at their initial positions. The memory of the crows are expressed as

\[
\text{Memory} = \begin{bmatrix}
m_1^1 & m_2^1 & m_3^1 & \cdots & m_d^1 \\
m_1^2 & m_2^2 & m_3^2 & \cdots & m_d^2 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
m_1^N & m_2^N & m_3^N & \cdots & m_d^N
\end{bmatrix}
\]

Step 3. Evaluate Fitness Function: Compute the fitness of the position of each crow \( x^i \) by inserting the decision variable values \( (x_1^i, x_2^i, \ldots, x_d^i) \) into the fitness function \( f(x_1, x_2, \ldots, x_d) \).

Step 4. Generate New Positions: Perform the steps below on each crow individually to generate new positions:
Suppose it is the turn of crow $i$ to generate a new position. The crow selects a crow in the flock (for example crow $j$) randomly and try to follow crow $j$ to discover the food's hiding place $\mathbf{m}^j$ of crow $j$.

In this case, two states may happen:

State 1: Crow $j$ is not aware of being followed by crow $i$. As a result, crow $i$ will approach to the hiding place of crow $j$.

State 2: Crow $j$ realizes that it is being followed by crow $i$. As a result, for protecting its food from being stolen, crow $j$ will fly to another position in the search space to fool crow $i$.

Totally, states 1 and 2 can be expressed as follows:

$$
\mathbf{x}^i = \begin{cases} 
\mathbf{x}^i + r_j \cdot f_l^{j,\text{iter}} \cdot (\mathbf{m}^j - \mathbf{x}^i), & r_j \geq AP^{j,\text{iter}}, \\
\text{a random position in the solution space}, & \text{otherwise}.
\end{cases}
$$

where $\mathbf{x}^i$ denotes the generated new position for crow $i$, $r_j$ and $r_j$ are random numbers between 0 and 1, $AP^{j,\text{iter}}$ and $f_l^{j,\text{iter}}$ are the awareness probability of crow $j$ and the flight length of crow $i$ at iteration $\text{iter}$ respectively.

**Step 5. Check the Feasibility of New Positions:** If the new position is within the solution space, the crow moves to the new position. Otherwise, the crow stays in its current position. That is

$$
\mathbf{x} = \begin{cases} 
\mathbf{x}^i, & X_{\min} \leq x_k^i \leq X_{\max} \quad \text{for } k = 1, 2, \ldots, d, \\
\mathbf{x}^i, & \text{otherwise},
\end{cases}
$$

where $x_k^i$ is the $k$th element of the vector $\mathbf{x}^i$.

**Step 6. Evaluate the Fitness of New Positions:** The fitness values of the new positions are computed.

**Step 7. Update Memory:** Crows update their memory as follows:

$$
\mathbf{m}^i = \begin{cases} 
\mathbf{x}^i, & f(\mathbf{x}^i) \text{ is better than } f(\mathbf{m}^i), \\
\mathbf{m}^i, & \text{Otherwise}.
\end{cases}
$$

**Step 8. Check Termination Criterion:** If the termination criterion are not met, go back to Step 4. Otherwise, output the set of optimal parameters, which is the optimal solution to the problem in the given solution space. Apart from flock size and maximum number of iterations, there are only two parameters (flight length and awareness probability) that need to be tuned by users. In comparison with genetic algorithm (GA) associated with 6 parameters and particle swarm optimization (PSO) associated with 4 parameters that need to be tuned, CSA is more easy to implement and thus is more attractive for applications.

4. Generation of developable surface from two design curves

In appearance design, 3D shapes are often depicted by sparse curves. Given two design curves, which are treated as the boundary curves of a ruled surface here, if the developability error of the resultant ruled surface is not within the tolerance of the material used in manufacturing, the developability error would result in failure in actual manufacturing. If so, original design curves need to be modified to achieve a high-accuracy surface developability. On the other hand, in order to preserve user's design intentions, new boundary curves should not deviate too much from original ones to keep the basic 3D shape of the original ruled surface. The relationship between the two sides is contradictory, so it is necessary to use a certain tradeoff between them.

To solve this problem, we allow perturbation of the control points of design curves within a allowed bound to expand the solution space of developable surfaces to achieve a high-accuracy developability. The allowed perturbation bound is defined by users in terms of the tolerance for developability error and the tolerance for the error of curve modification, which enables users to control the tradeoff between a high-accuracy developability and shape preservation according to practical demand in engineering. We can formulate the construction of quasi-developable surface as a geometric optimization problem where the objective function is defined as the developability degree of the surface and search within the allowed perturbation bound for the set of control points, with which the resultant surface has the highest degree of developability. In order to measure how close a surface is to being developable, a criterion — developability degree — should be defined.
4.1 Developability degree

For a general ruled surface \( R \) in Eqn.(3), the tangent lines to the boundary curves through two corresponding points \( a(t) \) and \( b(t) \) are usually not coplanar. If the boundary curves \( a(t) \) and \( b(t) \) can be selected such that for any \( t \) the tangent lines \( a'(t) \) and \( b'(t) \) and the corresponding ruling \( L_t(v) = (1-v)a(t) + vb(t) \) lie in the same plane, the surface possesses perfect developability. Coplanarity of the three lines can be expressed by the normals at the ends of the ruling

\[
\mathbf{n}_1(t) \times \mathbf{n}_2(t) = 0,
\]

where \( \mathbf{n}_1(t) = \frac{(a(t) - b(t)) \times a'(t)}{\|a(t) - b(t)\|} \) denotes the unit normal vector of the surface at the point \( a(t) \).

\[
\mathbf{n}_2(t) = \frac{(a(t) - b(t)) \times b'(t)}{\|a(t) - b(t)\|} \text{ denotes the unit normal vector at the point } b(t).
\]

In general, on a ruling of a ruled surface, there often exists a non-zero angle between the normals at the ends of the ruling, which is referred to as the warp angle (or normal twist) of the ruling. If the warp angle of any ruling equals to zero, namely the surface satisfies the common tangent plane condition along any ruling, the ruled surface becomes developable. Therefore, warp angles of rulings are usually used to measure the developability of ruled surfaces (Chen et al., 2011a).

To avoid the cross product of normal vectors above, a twist measure of a ruling \( v_tL \) for fixed \( t \) can be defined by (Liu et al., 2011)

\[
m(t) = (\mathbf{n}_1(t), \mathbf{n}_2(t)) - 1,
\]

where \((\cdot , \cdot )\) means the inner product of two vectors.

It is obvious that for fixed \( t \), the smaller the warp angle of the ruling \( L_t(v) \) is, the larger the value of \( m(t) \) is. Besides, \( m(t) \) is always less than or equal to 0, and \( m(t) = 0 \) only if accurate developability is achieved. Thus the developability degree of the ruled surface can be measured by

\[
E_{delp} = \int \left((n_1(t), n_2(t)) - 1\right) dt.
\]

For the sake of simplicity, we take sufficient sample rulings on \( R \), each of which corresponds to a unique parameter value \( t_k \) in the value range of \( t \). The developability degree of the ruled surface is measured by

\[
E_{delp} = \sum_{k=1}^{K} (n_1(t_k), n_2(t_k)) - 1). \tag{5}
\]

Given a ruled surface \( R \), the larger \( E_{delp}(R) \) is, the better developability the ruled surface has.

4.2 Generating quasi-developable surface from two design curves

In our method, since every control point of boundary curves is bounded in a box, the rulings of the resulting surface will not intersect as long as the initial design curves do not intersect. Therefore the smoothness of the resulting surfaces can be guaranteed naturally.

To prevent unnecessary modification to design curves, such as modification leading to the shrinking or extending of the ruled surface along its rulings, another objective function need to be added. We denote \( \{\Delta A_i\}_{i=0}^{m} \) and \( \{\Delta B_j\}_{j=0}^{n} \) the perturbation vectors of the control points \( \{\mathbf{A}_j\}_{i=0}^{m} \) and \( \{\mathbf{B}_j\}_{j=0}^{n} \) respectively.

If there is no specific requirement for the boundaries of the output surface, we allow perturbation of all the control points of both design curves within allowed bound. In this case, the total objective function is defined as

\[
E_{total} = E_{delp} + \lambda \left( \sum_{i=0}^{m} \|\Delta A_i\|^2 + \sum_{j=0}^{n} \|\Delta B_j\|^2 \right), \tag{6}
\]
where $E_{\text{delp}}$ is the developability objective function defined in Eqn.(5), $\lambda_i$ is a weighting coefficient.

If a design curve $a(t)$ is specified as the directrix of the ruled surface $R$, we allow perturbation of the control points $\{b_j\}_{j=0}^n$ of the other design curve $b(t)$ within allowed bound. In this case, the total objective function is defined as

$$E_{\text{total}} = E_{\text{delp}} + \lambda_2 \sum_{j=0}^n \|\Delta b_j\|^2,$$

where $\lambda_2$ is a weighting coefficient.

If the two end rulings of the original ruled surfaces need to be retained, we keep the end control points of each boundary curve unchanged and allow perturbation of the remaining control points within allowed perturbation bound. In this case, the total objective function is defined as

$$E_{\text{total}} = E_{\text{delp}} + \lambda_3 \left(\sum_{i=1}^{m+1} \|\Delta a_i\|^2 + \sum_{j=1}^n \|\Delta b_j\|^2\right),$$

where $\lambda_3$ is a weighting coefficient.

In Eqn.(5), the developability objective function $E_{\text{delp}}$ involves unitization of normal vectors and cross-product of vectors in the computation of normal vectors, which would lead to some high order expressions in the differentiation of $E_{\text{delp}}$. Moreover, the control points must lie within their allowed perturbation bound. Therefore using traditional optimization methods to solve this problem would result in complex calculation. In order to obtain the set of optimal control points within their allowed perturbation bound for achieving a high-accuracy developability, we resort to crow search algorithm (CSA) for its many advantages mentioned previously.

Here the algorithm of generating quasi-developables by modifying a design curve while retaining a directrix is described below as an example to show the process of generating quasi-developables by this method.

**The algorithm of generating quasi-developables**

**Step 1.** Input a B-spline directrix $a(t)$ and a design curve $b(t)$ in B-spline form whose control points are denoted by $\{b_j\}_{j=0}^n$.

**Step 2.** If the developability of the resultant ruled surface is unsatisfactory, according to practical engineering requirements, provide the set of control points $\{b_j\}_{j=0}^n$ a reasonable allowed perturbation bound in which we search for the set of optimal control points to achieve a high degree of developability.

**Step 3.** Sampling sufficient parameter values $\{t_k\}_{k=0}^K$ in the value range of $t$ to determine sampling rulings. Precompute and store the points $a(t_k)$ and derivatives $a'(t_k)$ for $k = 1, 2, \cdots, K$ by Eqn.(1) and Eqn.(2) respectively to accelerate future computations. Likewise, for the design curve $b(t)$, precompute and store the values of its basis functions $\{N_{j,p}(t_k)\}_{j=0}^n$ in Eqn.(1) and the derivatives $\{N'_{j,p}(t_k)\}_{j=0}^n$ in Eqn.(2) for $k = 1, 2, \cdots, K$.

**Step 4.** By using crow search algorithm where the decision variables are $\{\Delta b_j\}_{j=0}^n$ and the fitness function is the total objective function $E_0(B_0 + \Delta B_0, B_1 + \Delta B_1, \cdots, B_n + \Delta B_n)$ in Eqn.(7), we search within the allowed perturbation bound for the set of optimal perturbation vector $(\Delta B_0, \Delta B_1, \cdots, \Delta B_n)$. In the fitness evaluation of a set of control points $E(B_0 + \Delta B_0, B_1 + \Delta B_1, \cdots, B_n + \Delta B_n)$ in Eqn.(7), point $b(t)$ and the derivative $b'(t_k)$ are computed by Eqn.(1) and Eqn.(2) respectively with the precomputed values $\{N_{j,p}(t_k)\}_{j=0}^n$ and $\{N'_{j,p}(t_k)\}_{j=0}^n$ for $k = 1, 2, \cdots, K$.

**Step 5.** Compute the modified boundary curve $b(t)$ using the set of optimal control points $\{b_j + \Delta b_j\}_{j=0}^n$ and output a ruled surface with a high degree of developability by Eqn.(3).

### 4.3. Comparisons to existing works

Compared with these methods which find discrete rulings between two boundary curves in terms of developability (Wang et al., 2005; Tang et al., 2005; Liu et al., 2011; Chen et al., 2011a), our method can generate quasi-developable
surfaces with natural smoothness. Though a quasi-developable strip with $G^1$ or $G^2$ continuity can be constructed from two design curves by (Chen et al., 2011b; Chen et al., 2013), the construction process is usually complex because adjacent quasi-developable Bézier patches are required to fulfill $G^1$ or $G^2$ continuity conditions to achieve the smoothness of the resultant strip.

In (Gavriil et al., 2019), developability optimization of tensor product B-spline surfaces is presented. However, this method does not apply to our case where only sparse design curves are available. In (Chalfant et al., 1998), the method for generating quasi-developable B-spline surfaces based on optimization techniques is presented. However, the method can only generate quasi-developable Bézier surfaces with two boundary curves being of the same degree. If a boundary curve is in B-spline form, users need to extract Bézier segments from the B-spline curve to generate quasi-developable Bézier patches and merge these patches together for smoothness, which makes the design procedure complex. Besides, the method results in overdetermined nonlinear equations and the optimal control points are usually obtained through complicated calculation. In comparison with the method, our method is simpler.

Compared with the design of developable surfaces form control rulings (Park et al., 2002; Hu et al., 2020), the design from boundary curves can better depict designer’s aimed shapes. Besides, as stated in (Hu et al., 2020), there are many variables in the optimization problem, i.e., control points and a set of normal vectors, the shape of the solution space is often not good, which makes the optimization process unstable and even leads to an unsatisfactory solution in some cases. Therefore some regularization terms have to be added to control the shape of the solution space in (Hu et al., 2020). Besides, there are 3 weighting coefficients that need to be tuned by users. These weighting coefficients need to be selected carefully, or the balance of those different energy terms will be broken and as a result the shape of the resultant surface will be unsatisfactory. While our method is associated with only one weighting coefficient, which makes the design of quasi-developable surfaces easier for users.

5. Results and discussion
5.1 Results

In this section, 7 examples are given to verify the presented method. In our examples, we select 21 values in the value range of $t$ evenly to determine the sampling rulings on each surface, i.e., $K = 21$ in Eqn.(6) and Eqn.(7). Weighting coefficients $\lambda_1, \lambda_2, \lambda_3$ are set to 0.001. In regard to the values of adjustable parameters of CSA in these examples, the flock sizes in Example 1, Example 2, Example 3, Example 4, Example 5, Example 6 and Example 7 are 30, 50, 50, 80 and 80 respectively, and in each example, maximum number of iterations $\text{iter}_{\text{max}}$ is 60, flight length $fl$ decreases from 2 to 0.2 with iterations and awareness probability $AP$ is set to 0.05.

In order to show the effects of the developability optimization clearly, in each example, we display both original ruled surface and resultant surface with color coding of the Gauss-curvature. Moreover, the surfaces are displayed from the same viewing angle and in the same size coordinate system. In addition, in each example, we display the average warp angle of the optimal ruled surface in each iteration to show the decreasing of developability error with iterations, where the initial average warp angle is that of the original ruled surface.

Example 1. Fig. 1 shows the generation of quasi-developable surface by modifying a design curve while retaining a directrix. In Fig. 1, given a directrix (in red) and a design curve (a cubic Bézier curve in blue) whose control points are $\{B_j\}_{j=0}^n$, the original ruled surface is displayed in Fig. 1 (a). To improve the developability of the ruled surface, we keep its directrix unchanged and allow perturbation of the control points of the blue design curve with every coordinate component of $\Delta B_j$ being between $[-0.2,0.2]$ for $j = 0, 1, 2, 3$. By performing the proposed algorithm, we obtain the set of optimal control points and thus the resultant quasi-developable. Fig. 1 (b) and Fig. 1 (c) show the generated surface after the first iteration of CSA and 60 iteration of CSA respectively. Fig. 1 (d) shows the decreasing of average warp angle with iterations.

Example 2. Fig. 2 shows the generation of quasi-developable by modifying both design curves. Here the original design curves are those in Example 1 and the original surface can reference Fig. 1(a). Fig. 2(a) shows the generated quasi-developable surface with every coordinate component of $\Delta A_j$ and $\Delta B_j$ being between $[-0.1,0.1]$ for $j = 0, 1, 2, 3$. Fig. 2 (b) shows the decreasing of average warp angle with iterations.
Fig. 1 Generation of quasi-developable by modifying a design curve (a cubic Bézier curve in blue) with every coordinate component of $\Delta B_j$ being between $[-0.2, 0.2]$ for $j = 0, 1, 2, 3$ while retaining the red directrix.

Example 3. Fig. 3 shows the generation of quasi-developable with another two design curves, where the red directrix is kept unchanged. The blue design curve, a cubic Bézier curve whose control points are $\{B_i\}$, is modified for achieving a high degree of developability.

Example 4. Fig. 4 shows the generation of quasi-developable by modifying both design curves while retaining the two end rulings of the original ruled surface. The original design curves are those in Example 3 and the original surface can reference Fig. 3(a). We allow perturbation of the control points $A_1, A_2, B_1, B_2$ here. Fig. 4 (a) shows the generated quasi-developable surface with every coordinate component of $\Delta A_1, \Delta A_2, \Delta B_1, \Delta B_2$ being between $[-0.1, 0.1]$. Fig. 4 (b) shows the decreasing of average warp angle with iterations.
Fig. 3 Generation of quasi-developable by modifying a boundary curve (a cubic Bézier curve in blue) with every coordinate component of $\Delta \mathbf{B}_j$ being between $[-0.2,0.2]$ for $j = 0,1,2,3$ while retaining the red directrix.

Fig. 4 Generation of quasi-developable by modifying both boundary curves (two cubic Bézier curves in blue) with every coordinate component of $\Delta \mathbf{A}_1, \Delta \mathbf{A}_2, \Delta \mathbf{B}_1, \Delta \mathbf{B}_2$ being between $[-0.1,0.1]$ while retaining end rulings of the original surface.

**Example 5.** Fig. 5 shows the generation of quasi-developable with another two design curves (quartic Bézier curves), where the red directrix is kept unchanged and the blue boundary curve whose control points are $\left\{ \mathbf{B}_j \right\}_{j=0}^4$ is modified under every coordinate component of $\Delta \mathbf{B}_j$ being between $[-0.2,0.2]$ for $j = 0,1,\cdots,4$. 

(a) The original ruled surface.  
(b) The generated surface after the first iteration.
Example 6. Fig. 6 shows the generation of quasi-developable with two design curves in B-spline form, where the red directrix and two end rulings are kept unchanged. The blue boundary curve whose control points are $B_j$ is modified under every coordinate component of $\Delta B_j$ being between $[-0.1,0.1]$ for $j = 1, 2, \cdots, 6$.

Example 7. Fig. 7 shows the generation of quasi-developable with another two design curves in B-spline form, where the red directrix and two end rulings are kept unchanged. The blue boundary curve whose control points are $B_j$ is modified under every coordinate component of $\Delta B_j$ being between $[-0.1,0.1]$ for $j = 1, 2, \cdots, 6$. The red directrix and end rulings of the original surface are kept unchanged.
modified under every coordinate component of $\Delta \mathbf{B}_j$ being between $[-0.1,0.1]$ for $j = 1, 2, \cdots, 6$.

(a) The original ruled surface.  
(b) The generated surface after the first iteration.  
(c) The generated surface after 60 iterations.  
(d) The decreasing of average warp angle with iterations.

Fig. 7 Generation of quasi-developable by modifying a design curve (a B-spline curve with 8 control points in blue) with every coordinate component of $\Delta \mathbf{B}_j$ being between $[-0.1,0.1]$ for $j = 1, 2, \cdots, 6$. The red directrix and end rulings of the original surface are kept unchanged.

5.2 Discussion

Compared with the original ruled surfaces, we can see that the developability of the output surfaces has improved remarkably from their average warp angels and color coding of the Gauss-curvature, while keeping the error of curve modification within allowed bounds. The experiments demonstrate that our method works well to generate high-accuracy developables and offers users high design flexibility to control the overall shape of output quasi-developables. When the allowed perturbation bound is sufficiently big for a pair of design curves, a surface with a perfect developability can be obtained by our method. In these examples, the algorithm takes only a few seconds for most cases.

6. Conclusion

In this paper a method for generating quasi-developable surfaces from two design curves in B-spline form is presented. Through original design curves, users can control the basic shape of the surface and the design curves can be modified automatically when that is necessary to achieve a high-accuracy developability by our method. Simulation results show that our method is effective and user-friendly, which can be used by casual users with little modeling experience and can improve the problem-solving abilities in engineering appearance design significantly. A promising and important direction for future work is to incorporate practical engineering needs and specific material behaviors such as tensile properties to model required products. In addition, more high-efficient intelligent optimizers can be used in the developability optimization of surfaces.

References

Askarzadeh, A., A novel metaheuristic method for solving constrained engineering optimization problems: Crow
search algorithm, Computers and Structures, Vol.169 (2016), pp.1–12.

Bodduluri, R. and Ravani, B., Design of developable surfaces using duality between plane and point geometries, Computer-Aided Design, Vol.25, No.10 (1993), pp.621–632.

Bo, P. and Wang, W., Geodesic-controlled developable surfaces for modeling paper bending, Computer Graphics Forum, Vol.26, No.3 (2007), pp.365–374.

Chalfant, J. and Maekawa, T., Design for manufacturing using B-spline developable surfaces, Journal of Ship Research, Vol.42, No.3 (1998), pp.207–215.

Chen, M., Tang, K. and Joneja, A., Design of developable interpolating strips, Computer-Aided Design and Application, Vol.8, No.4 (2011a), pp.557–570.

Chen, M. and Tang, K., Quasi-developable surface modeling of contours with curved triangular patches, Computers & Graphics, Vol.37, No.7 (2011b), pp.851–861.

Chen, M. and Tang, K., G² quasi-developable Bézier surface interpolation of two space curves, Computer-Aided Design, Vol.45, No.11 (2013), pp.1365–1377.

Chu, C.-H. and Séquin, C.H., Developable Bézier patches: properties and design, Computer-Aided Design, Vol.34, No.7 (2002), pp.511–527.

Do Carmo, M., Differential Geometry of Curves and Surfaces, Prentice Hall, Englewood Cliffs, 1976.

Gavriil, K., Schiffner, A., Pottmann, H., Optimizing B-spline surfaces for developability and paneling architectural freeform surfaces, Computer-Aided Design, Vol.111 (2019), pp.29–43.

Hu, G., Cao, H., Qin, X. and Wang X., Geometric design and continuity conditions of developable λ-Bézier surfaces, Advances in Engineering Software, Vol.114, No.1 (2017a), pp.235–245.

Hu, G., Cao, H., Zhang, S., et al., Developable Bézier-like surfaces with multiple shape parameters and its continuity conditions, Applied Mathematics Modelling, Vol.45, No.5 (2017b), pp.728–747.

Hu, Z., Bo, P., Developable B-spline surface generation from control rulings, Computer Science - Graphics, 2020, https://arxiv.org/abs/2004.09038

Lang, J. and RÖschel, O., Developable (1, n)-Bézier surfaces, Computer Aided Geometric Design, Vol.9, No.4 (1992), pp.291–298.

Liu, Y., Tang, K., Gong, W., et al., Industrial design using interpolatory discrete developable surfaces, Computer-Aided Design, Vol.43, No.9 (2011), pp.1089–1098.

Li, C., Wang, R. and Zhu, C., An approach for designing a developable surface through a given line of curvature, Computer-Aided Design, Vol.45, No.3 (2013), pp.621–627.

Jung, A., Hahmann, S., Rohmer, D., et al., Sketching folds: Developable surfaces from non-planar silhouettes, ACM Transactions on Graphics, 2015.

Park, F., Yu, J., Chun, C., et al., Design of developable surfaces using optimal control, Journal of Mechanical Design, Vol.124, No.4 (2002), pp.1199–1204.

Pérez, F. and Suárez, J.A., Quasi-developable B-spline surfaces in ship hull design, Computer-Aided Design, Vol.39, No.10 (2007), pp.853–862.

Piegl, L. and Tiller, W., The NURBS Book (Monographs in Visual Communication) (Second ed.), Springer-Verlag, New York, 1997.

Tang, K. and Wang, C., Modeling developable folds on a strip, Journal of Computing and Information Science in Engineering, Vol.5, No.1 (2005), pp.35–47.

Wang, J., Jiang, P., Guo, Y., et al., Developable surface pencil pairs with special pairs as common asymptotes, Applied Mathematics and Computation, Vol.362 (2019). https://doi.org/10.1016/j.amc.2019.124583.

Wang, C. and Tang, K., Developable triangulations of a strip, Computer Aided Design & Applications, Vol.2, No.1–4 (2005), pp.233–242.

Zhou, M., Yang, J., Zheng, H. and Song, W., Design and shape adjustment of developable surfaces, Applied Mathematics Modelling, Vol.37, No.6 (2013), pp.3789–3801.