Interactive design of developable surfaces

CHENGCHENG TANG
Kind Abdullah University of Science and Technology
and
PENGBO BO
KAUST and Harbin Institute of Technology
and
JOHANNES WALLNER
Graz University of Technology
and
HELMUT POTTMANN
Vienna University of Technology and KAUST

We present a new approach to geometric modeling with developable surfaces and the design of curved-creased origami. We represent developables as splines and express the nonlinear conditions relating to developability and curved folds as quadratic equations. This allows us to utilize a constraint solver which may be described as energy-guided projection onto the constraint manifold, and which is fast enough for interactive modeling. Further, a combined primal-dual surface representation enables us to robustly and quickly solve approximation problems.

Additional Key Words and Phrases: interactive design, computational differential geometry, developable surface, spline surface, origami, curved folding, isometric deformation, digital reconstruction, constraint solving

ACM Reference Format:

Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. Interactive design of developable surfaces. *ACM Trans. Graph.* VOL, NO, Article NO (MONTH 2015), 12 pages.
DOI: 10.1145/2832906

1. INTRODUCTION

Developable surfaces are very important in both theory and practice. Being defined as surfaces which locally can be mapped to a planar domain without stretching or tearing, they represent the shapes obtainable with thin materials like sheet metal or paper which do not stretch. Thus developables are relevant to the manufacturing industry, for example for ship hulls [Pérez and Suárez 2007] and clothing [Chen and Tang 2010]. Freeform developables occur also in architecture and art, such as F. Gehry's designs which are piecewise-developable, and the curved-fold example of Figure 2 which is entirely developable. A topic with many contributions to developables is papercraft and origami.

We should emphasize that not only smooth developables but also crumpled/buckled surfaces are relevant [Kergosien et al. 1994]. Developables also occur in other places, for instance in connection with folding maps in augmented reality [Martedi and Saito 2011], 3D reconstruction [Perriollat and Bartoli 2012], or mesh segmentation [Julius et al. 2005; Yamauchi et al. 2005].

Despite their obvious importance, developable surfaces still present difficulties to today's CAD systems, and geometric modeling with developables in its full generality is not available, apart from *lofting*, i.e., defining a developable by its boundary.

The reason for that undoubtedly is the highly nonlinear nature of developability. Differential geometry provides us with a detailed mathematical description of developables: Assuming piecewise curvature-continuity, they consist of ruled surfaces with the additional property that the tangent plane along a ruling is constant (see Figure 3).

Statement of the problem. In order to make high quality developable surfaces available to industrial design, we have to model all degrees of freedom of composite surfaces whose individual parts

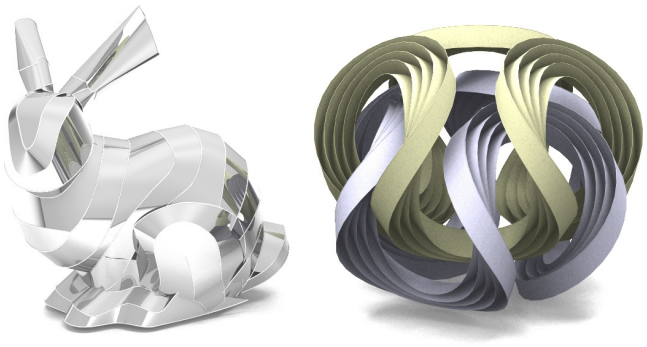


Fig. 1. We present methods for geometric modeling with developable surfaces. This includes composite piecewise-developables which approximate reference shapes (see bunny at left) as well as interactive design of curved origami. The right hand example is inspired by paper sculptures obtained by folding a sheet of paper along concentric rings, which go back to 1927 Bauhaus classes.

DOI: 10.1145/2832906

enjoy C^2 smoothness and which are constrained by developability. It is precisely this task which this paper is concerned with.

We propose an approach to developability based on splines, which is the standard surface representation of computer-aided design. A main contribution of this paper is the ability to quickly solve the constraints which express developability of such surfaces, and other nonlinear constraints related to isometric deformations, to curved origami, to approximation, and others.

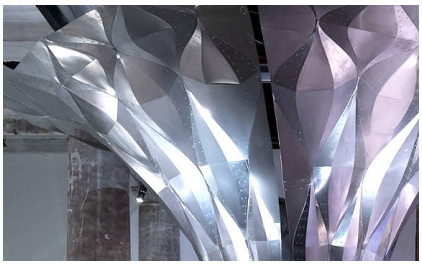


Fig. 2. The "Arum" surface was designed by Zaha Hadid Architects in cooperation with Robofold for the 2012 Venice Biennale. It consists of metal sheets folded along curved creases.

Previous work. The literature on developable surfaces is quite extensive. We already mentioned applications in the introduction above. As to geometric modeling with developables, we start with a discussion of *discrete* approaches to this problem. Mitani and Suzuki [2004] compute triangle meshes with feasible cutting and unfolding for arbitrary shapes. Wang and Tang [2004] deform a triangle mesh in order to make the entire surface developable.

Also buckled surfaces have been modeled by triangle meshes, see [Frey 2004]. Narain et al. [2013] go beyond developability, modeling more realistic material behaviour. Solomon et al. [2012] use a mesh approach to flexibly model the shapes achievable by bending and folding a given planar domain without stretching or tearing. Liu et al. [2006] treat developable surfaces as a limit case of quad meshes. Finally, we mention that Rose et al. [2007] show how to find 'optimal' developables from boundary curves. We would also like to point to the extensive list of references given by Solomon et al. [2012].

Geometric modeling with *continuous* developables is an old topic, but contributions might be summarized by the statement that the nonlinear nature of the developability constraint so far prevented interactive design. The general geometric design problem has not been solved, despite many contributions. Only very special cases have been successfully treated:

Lang and Röschel [1992] give the conditions for developability of rational Bézier (polynomial) surfaces for all degrees. This is a system of cubic equations. Maekawa and Chalfant [1998] perform geometric modeling for spline developables for the special case that boundary curves lie in parallel planes. G. Aumann [1991; 2003] studies the problem of finding a developable Bézier (polynomial) surface through one open boundary curve. Chu and Séquin [2002] discuss developability of spline surfaces and derive explicit forms of the developability constraints for surfaces up to degree 3. Chu and Chen [2004] continue this study and derive the number of degrees of freedom available for modeling. Pottmann et al. [2008] use splines for modeling developable surfaces based on the ideas of [Liu et al. 2006]. They use optimization to achieve approximate developability, using integrals for target functionals.

A different approach to modeling developables is to work with the dual representation. Here a developable is represented as the envelope of its tangent planes and is thus recognized as the projective dual of a space curve. This has been proposed by Boddu-luri and Ravani [1993], has been studied by [Pottmann and Farin 1995; Hoschek and Pottmann 1995; Pottmann and Wallner 1999], and is extensively discussed in [Pottmann and Wallner 2001]. The dual representation, however, is not intuitive and it is difficult to control singularities. Essentially also Peternell [2004] uses the dual representation to solve a fitting problem. Developables have been modeled as graphs of functions: The paper [Chen and Wang 2002] attempts to model developable surfaces via polynomial functions, which restricts its scope to cylindrical surfaces without singular points; moreover, some of the analysis in that paper does not agree with known properties of developables.

Closely related to developability is the topic of *isometric mapping* between planar domains and surfaces, i.e., the question which shapes can be generated by bending a flat domain. We already mentioned Solomon et al. [2012]. Reconstruction of folded objects is the topic of [Kilian et al. 2008]. A method of producing special curved-fold objects by iterated reflection has been presented by Mitani and Igarashi [2011]. There is, of course, an obvious connection to computational origami, which is an active field of research [McArthur and Lang 2012]. We exemplarily point to [Huffman 1976], [Mitani and Suzuki 2004], [Tachi 2010], [Tachi and Miura 2012] and [Wang and Chen 2011], and especially to Demaine et al. [2011b] and Dias et al. [2012] who treat an example also found in this paper. Software capable of producing curved-fold origami in special cases like rotational symmetry is available from [Mitani 2012].

The contributions of this paper are the following:

—we perform interactive modeling of high-quality developable surfaces;
—we use a combined primal-dual spline representation for developables which involves a normal vector field and which allows us to express "developability everywhere" by a finite number of constraints. We thus can solve for developability at interactive speed;
—we interactively handle curved-folding objects and maintain both local and global developability, giving the user a design tool for curved origami;
—we extend our interactive tool to handle design of surfaces which are isometric to a given domain;
—we perform approximation of reference shapes with developable and piecewise-developable surfaces. Success of approximation depends on the above-mentioned normal vector field.

The paper is organized as follows: In § 2 we discuss spline developables as the simplest building blocks of complex developable surfaces. § 3 treats the algorithmic setup for modeling composite surfaces, discusses a guided projection method for constraint solving, and extends the list of constraints we are imposing on splines. In § 4 we deal with interactive modeling and the approximation problem. Computing and updating developments are mesh-based procedures treated in § 5. § 6 summarizes the tools we have provided for modeling curved-creased origami. Finally, § 7 discusses performance and limitations, and concludes the paper.

2. SIMPLE DEVELOPABLES

An important step to achieve the goal of interactive modeling of high-quality developables is a good understanding of *simple* sur-

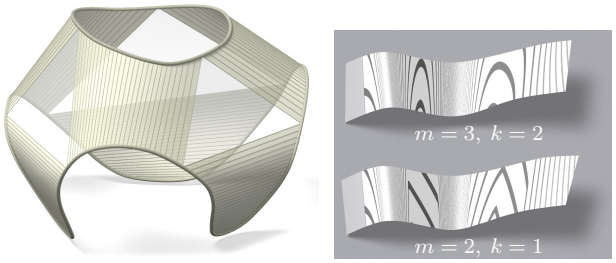


Fig. 3. Developables decompose into planar pieces and ruled patches. Curvature continuity is relevant for surface quality since it becomes visible in reflection patterns. The right hand images show spline developables of varying degree m and smoothness C^k.

faces which serve as building blocks for general developables. Since the latter are composed of ruled surface parts (Fig. 3), and we want to handle only *finitely* many degrees of freedom, the simple parts we employ are ruled spline surfaces which are developable. In the usual spline terminology, those spline surfaces are tensor product surfaces of degree $1 \times m$. This restriction to splines does not (for all practical purposes) diminish the variety of shapes available, in the same sense as using splines does not restrict the available shapes of freeform curves. This vague statement is made a mathematical theorem by proving the approximation power of splines in the curve case [de Boor 1978], and applying that result to the dual representation of developables [Pottmann and Wallner 1999].

Before dealing with the special case of spline developables, we start with some introductory paragraphs on developable ruled surfaces in general. A ruled surface connecting two parametric curves $\mathbf{a}(u)$ and $\mathbf{b}(u)$ has the parametric form

$$\mathbf{s}(u,v) = (1-v)\mathbf{a}(u) + v\mathbf{b}(u). \tag{1}$$

It is developable if and only if, for all u, it has the same tangent plane in the two points $\mathbf{s}(u,0) = \mathbf{a}(u)$ and $\mathbf{s}(u,1) = \mathbf{b}(u)$. There are equivalent ways of expressing this condition. Equation (2) expresses linear dependence of vectors directly with one equation involving first derivatives, while (3) uses three equations involving an auxiliary normal vector field $\mathbf{n}(u)$ to achieve the same purpose:

$$\mathbf{s} \text{ developable} \iff d(u) := \det(\mathbf{a}-\mathbf{b}, \mathbf{a}', \mathbf{b}') = 0 \tag{2}$$

$$\iff \langle \mathbf{n}, \mathbf{b}-\mathbf{a}\rangle = \langle \mathbf{n}, \mathbf{a}'\rangle = \langle \mathbf{n}, \mathbf{b}'\rangle = 0, \tag{3}$$

for all parameter values u. It turns out that it is useful to have two points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$ which span the curve's tangent line in the point $\mathbf{a}(u)$, and similarly for the curve $\mathbf{b}(u)$. A trivial choice would be $\mathbf{a}_0^{(1)} = \mathbf{a}$ and $\mathbf{a}_1^{(1)} = \mathbf{a} + \mathbf{a}'$, a more interesting one can be seen in Figure 4. In any case developability is expressed as

$$\mathbf{s} \text{ developable} \iff \mathbf{a}_0^{(1)}, \mathbf{a}_1^{(1)}, \mathbf{b}_0^{(1)}, \mathbf{b}_1^{(1)} \text{ co-planar} \iff$$

$$\langle \mathbf{n}, \mathbf{b}_1^{(1)} - \mathbf{a}_1^{(1)}\rangle = \langle \mathbf{n}, \mathbf{a}_1^{(1)} - \mathbf{a}_0^{(1)}\rangle = \langle \mathbf{n}, \mathbf{b}_1^{(1)} - \mathbf{b}_0^{(1)}\rangle = 0. \tag{4}$$

These conditions are supposed to hold for all parameter values, but we will see in §2.1 that it is actually sufficient to require them for a finite number of parameter values. Condition (2) is useful for counting degrees of freedom, but we prefer to use the equivalent conditions (3) and (4) in our computations. This is because they lead to quadratic equations expressing developability, while (2) generates cubic equations.

Degenerate cases. We mention a special case: If $\mathbf{b}(u) = \text{const.}$, the surface $\mathbf{s}(u,v)$ is a cone with base curve $\mathbf{a}(u)$ and apex coinciding with $\mathbf{b}_0^{(1)}(u) = \mathbf{b}_1^{(1)}(u) = \mathbf{b}(u) = \text{const}$. The developability condition is automatically satisfied, and a normal vector field exists.

2.1 Spline developables

When dealing with polynomial and piecewise-polynomial curves, it is convenient to represent them in B-spline form. For that purpose we consider the parameter interval $[u_0, u_n] = [0,1]$ which is subdivided into n nonempty sub-intervals $[u_0,u_1] \cup [u_1,u_2] \cup \ldots \cup [u_{n-1},u_n]$. We consider *spline functions* which enjoy C^k smoothness and which are polynomial of degree m in each subinterval ($k < m$). A *spline curve* is a curve whose coordinate functions are spline functions. To describe a *closed* curve, we also require that all derivatives up to order k are equal for $u = u_0$ and $u = u_n$.

The construction of B-spline basis functions $N_1(u), N_2(u), \ldots$ spanning a particular spline space is well known, see [de Boor 1978]. Splines are linear combinations of these basis functions:

$$\mathbf{a}(u) = \sum\nolimits_i \mathbf{a}_i N_i(u) \tag{5}$$

The points $\mathbf{a}_0, \mathbf{a}_1, \ldots$ are called the control points of the curve $\mathbf{a}(u)$. For any u, evaluation of the point $\mathbf{a}(u)$ is performed with the de Boor algorithm (Figure 4) which for $n = 1$ reduces to de Casteljau's algorithm for Bézier curves. It iteratively computes certain affine combinations of points, starting with the control points in the first round of iteration, and ending up with $\mathbf{a}(u)$ in the m-th round. In its next-to-last round, it produces auxiliary "first derivative" points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$, suitable for the developability condition (4), which span the tangent line of the curve in the point $\mathbf{a}(u)$.

Computational setup for spline developables. Our computational setup for dealing with spline developables is the following: We use as variables the B-spline control points for each of the curves $\mathbf{a}$, $\mathbf{b}$. Within each parameter subinterval, developability is characterized by vanishing of the degree $3m-3$ polynomial $d(u)$. It is therefore sufficient to enforce developability for $3m-2$ different values of u. For each of these values of u we use as additional variables a normal vector $\mathbf{n}(u)$ and the points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$, $\mathbf{b}_0^{(1)}(u)$, $\mathbf{b}_1^{(1)}(u)$. As constraints we impose the developability condition (4) as well as the linear defining relations of auxiliary points which originate in de Boor's algorithm.

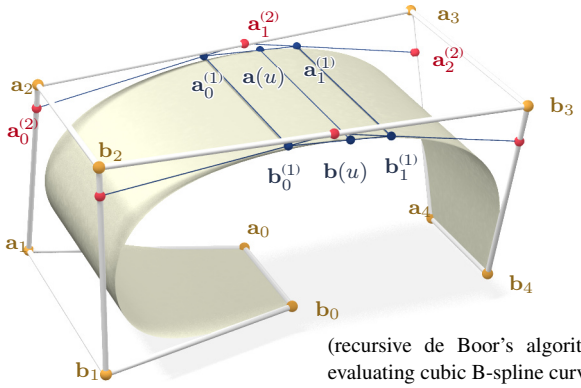


Fig. 4. A spline curve $\mathbf{a}(u)$ is defined by its control points $\mathbf{a}_0, \mathbf{a}_1, \ldots$. It is equipped with auxiliary first derivative points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$ which span the tangent line, and second derivative points $\mathbf{a}_0^{(2)}(u)$, $\mathbf{a}_1^{(2)}(u)$, $\mathbf{a}_2^{(2)}(u)$ which span the osculating plane of the curve. These auxiliary points are computed with de Boor's algorithm. Developability of the ruled surface defined by curves $\mathbf{a}$, $\mathbf{b}$ is equivalent to coplanarity of $\mathbf{a}_0^{(1)}$, $\mathbf{a}_1^{(1)}$, $\mathbf{b}_0^{(1)}$, $\mathbf{b}_1^{(1)}$.

It is actually possible to obtain developability by enforcing it for a smaller number of parameter values. The corresponding counting of degrees of freedom is performed in the appendix. This reduction of the number of constraints is however not so convenient, since the location of the parameter values has to satisfy the Schönberg-Whitney conditions [de Boor 1978]. It is also not necessary.

2.2 Normal splines and dual representation.

A developable defined by linear interpolation of spline curves $\mathbf{a}$, $\mathbf{b}$ has the normal vector field $\mathbf{n}(u) = \mathbf{a}'(u) \times \mathbf{b}'(u)$. It is of polynomial degree $M = 2m - 2$ and smoothness $K = k - 1$. Such vector fields can be added as additional information in form of more control points. Additional constraints prevent the normal vector field from degenerating: We require as a hard constraint that $\mathbf{n}(u)^T\mathbf{n}(u) = 1$, for $u = 0$, and as soft constraint (with a very small weight) that control points are unit vectors. We might even, for additional regularization, use normal vector fields of lower degree and higher smoothness. If such *normal splines* are used, then control points are added as auxiliary variables, and the individual normal vectors which occur in our equations are linked to the normal spline by a linear relation which expresses evaluation at a certain parameter value u.

The normal vector field of a developable is part of its *dual representation*, which means describing the developable by its 1-parameter family of tangent planes (cf. [Bodduluri and Ravani 1993; Pottmann and Wallner 1999]). By using the normal spline, we combine the essential information contained in both the primal and the dual representations. This combination will be important in § 4.2.

2.3 Extension to NURBS

A piecewise-polynomial spline curve in $\mathbb{R}^4$ defines a piecewise-rational spline curve in three-space if it is interpreted as homogeneous coordinates. It is therefore straightforward to extend the spline developables discussed above to the case of NURBS developables. Developability is still characterized by the property that the auxiliary points $\mathbf{a}_0^{(1)}(u)$, $\mathbf{a}_1^{(1)}(u)$, $\mathbf{b}_0^{(1)}(u)$, $\mathbf{b}_1^{(1)}(u)$ lie in a common plane. With $\mathbf{u}(u)$ as the homogeneous coordinate vector of that plane, the developability condition now reads $\langle \mathbf{u}, \mathbf{a}_1^{(1)} \rangle = \langle \mathbf{u}, \mathbf{a}_0^{(1)} \rangle = \langle \mathbf{u}, \mathbf{b}_1^{(1)} \rangle = \langle \mathbf{u}, \mathbf{b}_0^{(1)} \rangle = 0$. The computational setup for NURBS is exactly the same as for polynomial splines, with the exception that we use $\mathbf{u}(u)$ instead of $\mathbf{n}(u)$ and the developability condition is changed. The curve $\mathbf{u}(u)$ is the *complete dual representation* of the developable.

3. COMPUTATIONAL SETUP

This section describes the algorithmic setup we employ for geometric modeling with developables. We use the detailed discussion of simple developables in § 2 to grow a system of variables and equations, representing degrees of freedom and the constraints which are imposed on them. Interactive modeling is possible because we can solve this system quickly.

3.1 Setting up variables and equations

In order to set up geometric modeling with developables, we use a mesh to store the combinatorial information on how its "simple" pieces are connected with each other, see Figure 5. Each edge is labelled either R (*ruling*) or S (*spline*). We allow only 4 types of faces, classified according to boundary labels:

—S R S R: such a face corresponds to a spline developable;
—S R S: the same with a ruling degenerated into a point;

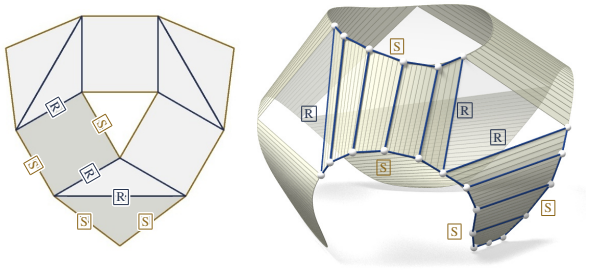


Fig. 5. The combinatorial setup of a composite seamless developable is stored in a mesh, where edges are labelled either R for "ruling" or S for "spline". Faces accordingly correspond to spline developables, cones, or planar parts. The example shown here has no cones. There is smooth transition between surfaces along all nine R-edges, and between boundary curves in all 12 vertices.

—S R R: also a spline developable, but a spline boundary is degenerated to a point, so the surface is a cone;
—R R . . . R: such a face corresponds to a planar n-gon.

It is known that a developable is composed of surfaces of these four types, but there could be an infinite number of them even for finite developables [Pottmann and Wallner 2001]. For geometric modeling we disregard this possibility.

Combinatorial consistency of spline data. In the mesh just introduced, faces correspond to simple surfaces. We therefore assign to each vertex of the mesh a location in space, and to each oriented S-edge $\vec{e}$ a control point sequence (reversing orientation reverses that sequence). For each face f, the boundary cycle $\vec{e}_0\vec{e}_1 \ldots$ shall be numbered such that labels are exactly as in the list above. We assign spline space data to f, namely degree m_f, smoothness k_f, and n_f parameter intervals. Obviously, these data have to be consistent:

• *Edge–face consistency:* Any S-edge contained in a face f must have the number of control points required by the spline space associated with the face. Then the spline space of f defines spline curves $\mathbf{c}_{\vec{e}_0,f}$ and $\mathbf{c}_{-\vec{e}_2,f}$ (if present) which correspond to the pair of opposite edges $\vec{e}_0$, $-\vec{e}_2$. They in turn define the spline surface attached to the face f. If there is only one S-edge, we assign to the face a cone with base curve $\mathbf{c}_{\vec{e}_0,f}$ and its opposite vertex as apex. In order to assign curves also to the edges with reverse orientation, we require $\mathbf{c}_{-\vec{e},f}(u) = \mathbf{c}_{\vec{e},f}(1-u)$.

• *Face–face consistency:* For any S-edge $\vec{e}$ contained in faces f_1, f_2, the curves $\mathbf{c}_{\vec{e},f_1}$ and $\mathbf{c}_{\vec{e},f_2}$ must be the same geometrically, if not parametrically. This is guaranteed by $m_{f_1} = m_{f_2}$, $k_{f_1} = k_{f_2}$, $n_{f_1} = n_{f_2}$, and by the condition that the parameter intervals of f_1, f_2 correspond under a linear parameter transform.

• *Vertex–edge consistency:* the control point sequence of an S-edge $\vec{e}$ must start and end with $\vec{e}$'s initial and final vertex.

The coordinates of control points and vertices are *variables* in our procedures. The above consistency conditions amount to the identification of some of these variables.

Geometric consistency of spline data — constraint equations. Besides the combinatorial consistency discussed above, the variables obey geometric consistency conditions:

—We impose developability on any ruled spline surface associated with a face with two S-edges. We use auxiliary variables and equations as described by § 2.

—No conditions are imposed on cones, i.e., faces with one S-edge. Normal vectors are available as auxiliary variables in the same way as for spline developables.

—A face f without an S-edge is to be a planar n-gon. We introduce its normal vector $\mathbf{n}_f$ as auxiliary variable and impose equations $\mathbf{n}_f^T \mathbf{n}_f = 1$ as well as $\langle \mathbf{n}_f, \mathbf{v}_{i+1} - \mathbf{v}_i \rangle = 0$ for all edges $\mathbf{v}_i \mathbf{v}_{i+1}$ of f.

Smoothness is an important property of a composite surface. It is a design decision to impose smooth transitions between faces, or between curves, see Figure 5. For two surfaces sharing a ruling, the respective unit normal vectors to either side can be expressed in terms of the auxiliary variables in our system, and a smooth transition is easily imposed by requiring equality of these vectors. Spline curves with common endpoints might be required to have the same derivative there. This condition is linear and is added to the system. Without spelling out the details we mention that also the weaker condition that two spline curves have the same tangent is expressible in terms of linear equations, after auxiliary normal vectors have been introduced. Other geometric constraints, like those pertaining to curved folding, or those involving user interaction, are discussed later, in Sections 3.3 and 4.

3.2 Solving constraint equations by guided projection

So far we have assembled a collection of variables which can be seen as a vector $\mathbf{x} \in \mathbb{R}^N$, and equations which can be written in the form $\phi_i(\mathbf{x}) = 0$, where $i = 1, \ldots, M$. The functions ϕ_i we encountered so far are linear or quadratic, and they are all hard constraints, especially those expressing developability and planarity of surfaces. Solving this system of nonlinear equations is nontrivial, since there are redundant equations, and there is a higher-dimensional solution manifold.

Tang et al. [2014] solve such a system of equations by an iterative procedure which starts from an initial configuration $\mathbf{x}_0$ and iteratively computes $\mathbf{x}_1, \mathbf{x}_2, \ldots$ which converge towards a solution. They linearize equations $\phi_i(\mathbf{x}) = 0$, and compute $\mathbf{x}_{k+1}$ as solution of $\phi_i(\mathbf{x}_k) + \langle \nabla \phi_i(\mathbf{x}_k), \mathbf{x} - \mathbf{x}_k \rangle = 0$, $i = 1, \ldots, M$. This system of linearized equations is symbolically written as $H^{(k)}\mathbf{x} = \mathbf{r}^{(k)}$. For the two reasons given it is not solvable (the presence of redundant equations together with numerical inaccuracies inevitably causes any solution of the system to widely diverge from any sensible solution of the original problem). To circumvent this problem, Tang et al. [2014] regularize and compute $\mathbf{x}_{k+1}$ by minimizing

$$\|H^{(k)}\mathbf{x} - \mathbf{r}^{(k)}\|^2 + \varepsilon_1^2 \|K^{(k)}\mathbf{x} - \mathbf{s}^{(k)}\|^2 + \varepsilon_2^2 \|\mathbf{x} - \mathbf{x}_k\|^2, \qquad (6)$$

where $\|K^{(k)}\mathbf{x} - \mathbf{s}^{(k)}\|^2$ is an energy functional and $\varepsilon_1, \varepsilon_2$ are small weights. In this way the system $\{\phi_j(\mathbf{x}) = 0\}_{j=1,\ldots,M}$ is solved by a regularized Newton method, where an energy function guides us towards a point on the solution manifold. As to the weights, ε_1 initially dominates ε_2, but later goes to zero (see §7 for details).

The energy employed in (6) is a sum of different terms: fairness (e.g. squares of second differences) occurs together with other soft targets like proximity to a reference shape. Since splines have a certain kind of fairness already built in, fairness energies are not as important as when modeling with meshes, but still for all control point sequences $\{\mathbf{a}_i\}$ we use the energy $\sum \|\mathbf{a}_{i+1} - 2\mathbf{a}_i + \mathbf{a}_{i-1}\|^2$ with a low weight to avoid degeneracies.

Using the constraint solver for modeling. The general framework described here — variables, equations, and their solution, is the algorithmic core of our applications which include approximation, interpolation, and interactive geometric modeling of developables, in particular curved-folding developables. Various hard and soft targets which correspond to these different applications are discussed in the next sections, and so is initialization of variables.

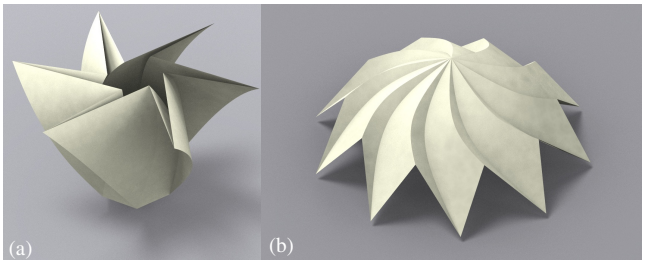


Fig. 6. Shapes foldable from a single sheet of paper consist of ruled developables. Secondly, smooth creases enjoy the "curved fold" property, and thirdly the angle sum around each vertex equals 2π. These three conditions are sufficient for local developability.

3.3 Further geometric properties of developables

Figure 6 illustrates shapes made by folding a single sheet of paper along curved creases. It is well known that this property has implications on the creases' curvatures (curved fold condition, see [Kilian et al. 2008]) and also on the angles between creases in vertices. Conditions on vertices are postponed to § 5; in the following we consider conditions on creases. Assume a crease $\mathbf{c}(u)$ which serves as common boundary of strips Φ^{Left} and Φ^{Right} to either side. We are going to need its geodesic curvatures $\kappa_g^{\text{Left}}(u)$ and $\kappa_g^{\text{Right}}(u)$ w.r.t. Φ^{Left} and Φ^{Right}, respectively. We are interested in cases where $\mathbf{c}$'s osculating plane bisects the tangent planes of Φ^{Left} and Φ^{Right}, which are thought to possess unit normal vectors $\mathbf{n}^{\text{Left}}(u)$, $\mathbf{n}^{\text{Right}}(u)$, respectively.

Curved folds and the geodesic property. There are two ways how the osculating plane mentioned above can bisect the strips to the left and to the right: Either $\mathbf{n}^{\text{Left}} + \mathbf{n}^{\text{Right}}$ or $\mathbf{n}^{\text{Left}} - \mathbf{n}^{\text{Right}}$ is a normal vector of the osculating plane. The definition of geodesic curvature (cf. [do Carmo 1976]) implies that these cases are equivalent to

$$\kappa_g^{\text{Left}}(u) - \kappa_g^{\text{Right}}(u) = 0 \qquad \text{(curved fold condition)}, \qquad (7^*)$$

$$\kappa_g^{\text{Left}}(u) + \kappa_g^{\text{Right}}(u) = 0 \qquad \text{(geodesic condition)}. \qquad (8^*)$$

In the first case, unfolding the two strips yields planar boundaries which fit together, so unfolding can be performed without any cutting along the curve $\mathbf{c}$ (*curved fold condition*, see Figure 6), while the second case yields planar boundaries which are mirror reflections of each other. This property has been called *geodesic* by Pottmann et al. [2008] and is relevant for panelization of architectural freeform skins, see Figure 7.

The algorithm of de Boor for evaluating a B-spline curve $\mathbf{c}$ produces auxiliary "second derivative" points of the curve which we denote by $\mathbf{c}_0^{(2)}(u)$, $\mathbf{c}_1^{(2)}(u)$, $\mathbf{c}_2^{(2)}(u)$ and which span the osculating plane of $\mathbf{c}(u)$, see Figure 4. Using them, we rewrite (7^*) and (8^*) resp. as

$$\langle \mathbf{c}_0^{(2)} - \mathbf{c}_1^{(2)}, \mathbf{n}^{\text{Left}} + \mathbf{n}^{\text{Right}} \rangle = \langle \mathbf{c}_1^{(2)} - \mathbf{c}_2^{(2)}, \mathbf{n}^{\text{Left}} + \mathbf{n}^{\text{Right}} \rangle = 0, \quad (7)$$

$$\langle \mathbf{c}_0^{(2)} - \mathbf{c}_1^{(2)}, \mathbf{n}^{\text{Left}} - \mathbf{n}^{\text{Right}} \rangle = \langle \mathbf{c}_1^{(2)} - \mathbf{c}_2^{(2)}, \mathbf{n}^{\text{Left}} - \mathbf{n}^{\text{Right}} \rangle = 0. \quad (8)$$

To add either property to our computational setup, we require (7) resp. (8) for sufficiently many values of u, each time adding the objects involved as auxiliary variables, and adding as equations the linear defining relations of $\mathbf{c}_i^{(2)}$, an appropriately relabelled Equation (4) which defines the normal vectors $\mathbf{n}^{\text{Left}}$, $\mathbf{n}^{\text{Right}}$, and finally the normalization constraints $\|\mathbf{n}^{\text{Left}}\|^2 = \|\mathbf{n}^{\text{Right}}\|^2 = 1$.

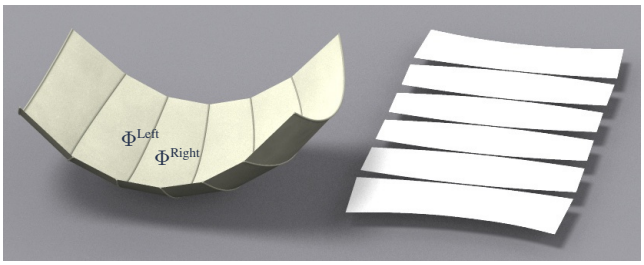


Fig. 7. Here, unfolding neighbouring strips Φ^{Left}, Φ^{Right} maps their common boundary to planar curves which are mirror reflections of each other. This "geodesic" property makes the developments of strips straighter, which is useful for materials like wood [Meredith and Kotronis 2012].

Remark: In § 2.1 we argued that enforcing (2) for $3m-2$ values of u in each parameter sub-interval implies validity of (2) for all u. The corresponding number of evaluations for (7) and (8) would be too high, so these conditions are enforced only numerically.

Straight unfolding property. If the boundary $\mathbf{c}$ of the strip Φ^{Left} has zero geodesic curvature, it unfolds to a straight line. This property can be imposed in a manner analogous to (7) and (8): We require that $\mathbf{c}$'s osculating plane is orthogonal to Φ^{Left}: $\langle \mathbf{c}_0^{(2)} - \mathbf{c}_1^{(2)}, \mathbf{n}^{\text{osc}} \rangle = \langle \mathbf{c}_1^{(2)} - \mathbf{c}_2^{(2)}, \mathbf{n}^{\text{osc}} \rangle = \langle \mathbf{n}^{\text{Left}}, \mathbf{n}^{\text{osc}} \rangle = 0$, where the auxiliary variable $\mathbf{n}^{\text{osc}}$ is a normal vector of the osculating plane.

4. MODELING WITH DEVELOPABLES

In this section we deal with the two main applications of the setup and algorithms described in §2 and §3, namely approximation of reference shapes by developable surfaces, and interactive modeling. We start with interactive modeling, even if approximation is part of it.

4.1 Interactive design of composite surfaces

We have implemented an interactive tool for geometric modeling with composite developables, which includes creating them from scratch. From the strictly algorithmic viewpoint, a user always starts with a composite developable surface governed by a labelled coarse mesh as discussed in § 3.1. This surface can be very simple like a planar or cylindrical piece (see Figure 8), or it can be generated by an approximation procedure as described by § 4.2, or it can be generated by lofting (cf. [Rose et al. 2007]) followed by approximation in order to obtain a spline representation. The capabilities of the modeling tool include:

—subdivision of patches by introducing new creases;
—relabeling edges and changing the role of patches
—dragging vertices to modify the shape of the surface;
—dragging a crease, changing the sizes of adjacent patches;
—selecting a target shape to be approximated;
—special cases of approximation, such as conversion of developables into spline representation, and putting strips onto reference shapes
—imposing properties like 'curved fold' on creases.

We have not implemented a full-fledged modeler, in particular we lack the surface-surface intersection and trimming procedures present in commercial software. We confined ourselves to functionality not otherwise available (see accompanying VIDEO).

Implementation of user interaction. Implementing this tool would be standard except for the nonlinear side-conditions which occur: They are developability of individual surfaces, and may include further constraints like the curved fold property of creases. Therefore each standard procedure (like dragging a vertex) has to be augmented by repeated calls to the *guided projection* algorithm of §3.2. From a high-level viewpoint, every intended modification to the surface is expressed by a possible extension of the collection "$\mathbf{x}$" of variables, and the introduction of constraints $\psi_i(\mathbf{x}) = 0$ and soft targets (energies). We now simply add these to the existing constraints and soft energies, and apply guided projection.

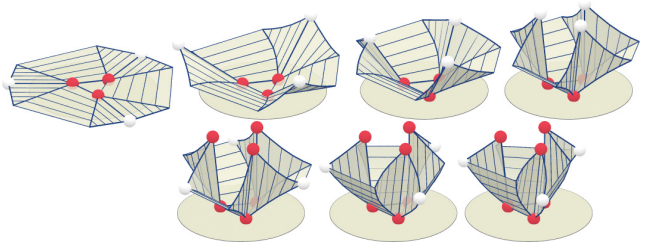


Fig. 8. Interactive modeling under curved-fold side conditions. Starting with a flat composite developable, on top we show dragging selected (white) vertices upwards while other vertices (red) remain fixed. The bottom row illustrates rotating selected vertices around a vertical axis. Control points are connected by blue edges.

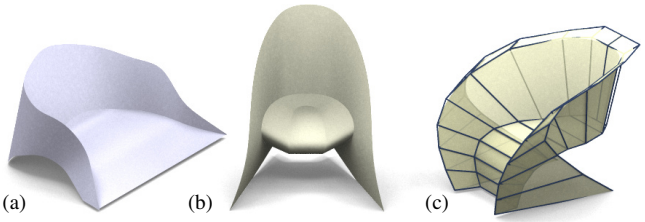


Fig. 9. These surfaces are the product of interactive modeling. They all have the same combinatorics (in the sense of Figure 5). In subfigure (c), control points are connected by blue edges.

Feedback to the user. Assuming the user uses the mouse for modeling, as a rule of thumb we run 1 round of guided projection while the mouse moves in order to give real-time feedback, followed by another 10 iterations after the mouse is released. The software displays the extent to which the desired properties have been achieved. In our experience, 10 iterations were enough. Figure 17 documents the achieved surface quality.

4.2 Approximation

An important approximation task is to represent arbitrary target shapes by composite developables, see Figure 10. Other examples of approximation which occur within interactive modeling are mentioned in §4.1. We solve this task by guiding developables specified by the user towards the target. We do not aim at finding the "best" approximation in any well-defined mathematical sense. Our way of representing piecewise-developables automatically produces watertight composite patches.

Since the target shape may be a noisy point cloud, we measure the approximation error by distances of target points from the moving developable, and not the other way round. Another reason to employ the moving developable's distance field is its smooth nature, and the assistance provided by a smooth normal spline. This method is motivated by Wang et al. [2006] who approximated point clouds by spline curves.

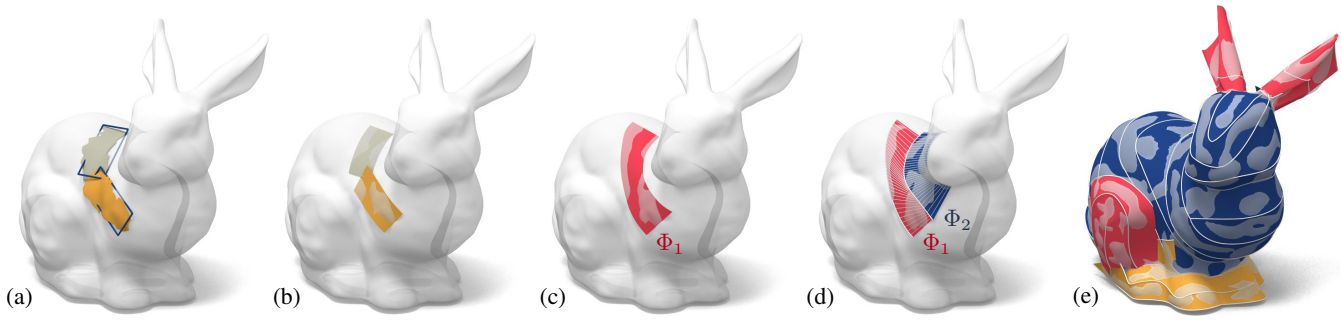


Fig. 10. Covering a complex shape by developable strips. (a) The user marks two areas on the bunny intended to be approximated by cubic developables. PCA is used to fit initial flat strips, which are visualized as rectangles. (b) The result of approximation. (c) The two strips obtained in the previous step are merged by putting together their respective control point sequences (averaging the end control points) and approximating again. (d) Having produced two strips side by side, they are subjected to the approximation procedure, with proximity of boundaries as an additional optimization goal. Such strips serve to initialize a composite surface. (e) The bunny is approximated by 6 composite surfaces shown in different colours. Figure 1 displays the same surface.

Equations expressing approximation. In order to bring the developable close to a reference shape, we represent the latter by a point cloud. A preprocessing step thins out the cloud and computes a "surface normal" $\mathbf{m}_j$ in each remaining point $\mathbf{q}_j$ of the reference shape, by locally fitting a plane to the cloud. Thinning is done similarly to Poisson sampling: When a data point is selected, points within a local ball neighbourhood of fixed radius are used to compute a normal vector and are subsequently discarded. The size of the neighbourhood we employ for local fitting is adapted to the level of detail we want to reproduce by our approximation. The developable $\mathbf{s}(u, v)$ is assumed to be defined by linear interpolation of curves $\mathbf{a}(u)$ and $\mathbf{b}(u)$. For each $\mathbf{q}_j$ we apply closest point projection onto the developable, which results in the point $\mathbf{s}(u_j, v_j)$.

Various aspects of proximity of reference point cloud and developable are expressed in the following equations.

$$\mathbf{s}(u_j, v_j) - \mathbf{q}_j = \mathbf{o}, \tag{9}$$

$$\langle \mathbf{m}_j, \mathbf{a}(u_j) - \mathbf{b}(u_j) \rangle = \langle \mathbf{m}_j, \mathbf{a}'(u_j) \rangle = 0, \tag{10}$$

$$\langle \mathbf{s}(u_j, v_j) - \mathbf{q}_j, \mathbf{n}(u_j) \rangle = 0. \tag{11}$$

Equation (9) is the most direct expression of proximity but is given only a low weight, since its strict enforcement prohibits tangential motion of the developable along the target. Equation (10) is a sanity check, expressing the fact that the developable is tangent to the target. Equation (11) pushes $\mathbf{q}_j$ towards the tangent plane of the developable in $\mathbf{s}(u_j, v_j)$.

Performing approximation is extremely easy, and we do not have to define a new algorithm for it. We only add equations (9)–(11) to our system of constraints, giving a low weight to (9) and (10). The variables in these constraints are the spline coefficients of curves $\mathbf{a}$, $\mathbf{b}$ and of the normal vector field $\mathbf{n}(u)$. The parameter values u_j, v_j are updated after each round of iteration. The points $\mathbf{q}_j$ and normal vectors $\mathbf{m}_j$ are fixed. However we should note that there are no theoretical guarantees of convergence.

Approximation with composite developables. Figure 10 shows a more complex example. If one already guesses at the final strip layout shown by Figure 10e, it is not difficult to perform the iterative procedure described in the figure caption, placing strips on the reference shape, merging and optimizing them again. This paper however does not propose any new method to tackle the difficult task of finding a suitable strip layout. For that we refer to [Mitani and Suzuki 2004] (if a surface is to be approximated) and to [Rose et al. 2007] (if boundary curves are given).

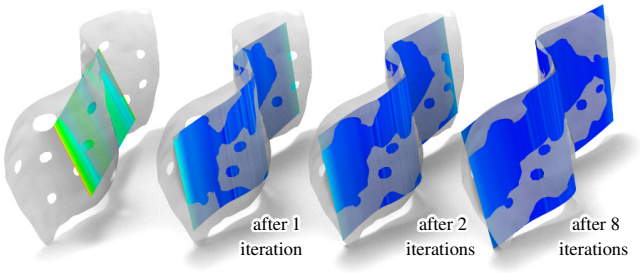


Fig. 11. A simple approximation problem, where a reference shape (point cloud) is approximated by a developable strip. The equations we use to express proximity make the strip grow so as to cover as much of the reference shape as possible. The number of iterations refers to guided projection according to § 3.2.

Figure 10 illustrates the process of creating a composite developable covering a reference shape. Here user-defined regions $\Psi_1, \Psi_2, \ldots$ are approximated by strips $\Phi_1, \Phi_2, \ldots$, respectively. Each Φ_j is defined by boundary spline curves $\mathbf{a}_j, \mathbf{b}_j$. We aim at *merging* these strips so that eventually they form a seamless composite surface: not only must these strips approximate $\Psi_1 \cup \Psi_2 \cup \ldots$, but their boundaries have to approximate each other. These latter proximity constraints are set up in a recursive manner as follows. The variables which represent the boundary spline curves of all strips are already present in the system, and so are the constraints which express approximation of the reference shape. The user now indicates a pairing between one boundary curve, say $\mathbf{a}_i$, with another boundary, say $\mathbf{b}_j$. Sampling the curve $\mathbf{a}_i(u)$ at regularly spaced parameter values creates vertices $\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, \ldots$ which are new variables — each is connected to the remaining variables by the linear evaluation relation (5). We then find the closest point projection of all $\mathbf{v}_{i,k}$'s onto $\mathbf{b}_j$ and compute the tangent line $T_{i,k}$ there. We add the linearized proximity constraints "$\mathbf{v}_{i,k} \in T_{i,k}$" to our system. The user continues to identify pairings, until all variables and constraints have been set up. We then invoke our solver and iterate (recomputing the $T_{i,k}$'s after each iteration).

5. GLOBAL DEVELOPABILITY

Intrinsically flat surfaces and developable surfaces. It is important to appreciate the local nature of the developability conditions men-

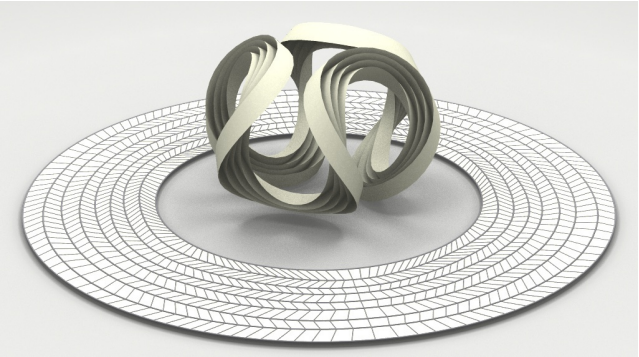


Fig. 12. This curved-crease sculpture is folded from an annulus. It has been interactively created along the lines of Figure 8. The annulus unfolding shown here actually is the mesh $(\bar{V}, \bar{E}, \bar{F})$ used for internal computations. Its edges correspond to creases and rulings.

tioned in previous sections. All points of a smooth ruled strip obeying (2), resp. (4) have a neighbourhood which can be unfolded to the plane in an isometric way, without stretching or tearing. The same is true for points on a smooth crease obeying (7). For a vertex where creases meet, an additional condition is needed to ensure local developability: The sum of angles between successive creases (when circling that vertex) must equal 2π, meaning that the Gauss curvature concentrated in that vertex is zero. A surface locally developable in this way is *intrinsically flat*.

However, global unfolding to the plane might fail for a variety of reasons: overlaps may occur, holes introduce closure conditions in the development, and the surface's topology might prevent developability on principle. E.g. the "hexagonal column" of Figure 14 is unfoldable onto a right circular cylinder, and the flat torus of Figure 15b is not globally unfoldable onto any simple surface. The word "developable" unfortunately is applied to both the local and the global property indiscriminately.

This section completes our algorithmic treatment of developability, by discussing the computation of developments, local developability around vertices, global developability, and isometric mapping.

Conversion of surfaces to meshes. Since spline developables in general do not have spline unfoldings/developments, we must convert data to meshes if developments are going to play a role in our modeling procedures: Each individual spline surface $\mathbf{s}(u, v)$ is converted into a sequence of quads (or possibly triangles, if an edge degenerates) by sampling boundary curves $\mathbf{a}(u)$, $\mathbf{b}(u)$ for predefined values of u. This yields vertices $\mathbf{s}(u_i, 0) = \mathbf{a}(u_i)$ and $\mathbf{s}(u_i, 1) = \mathbf{b}(u_i)$. Together with additional planar parts of our surface we produce a quad-dominant mesh (V, E, F) whose faces automatically are almost planar, as shown by [Liu et al. 2006]. An approximate development in form of an entirely flat quad mesh $(\bar{V}, \bar{E}, \bar{F})$ is easily found, by successively putting together developments of individual faces. If modeling starts with a flat surface, such as in Figures 8, 12 or 14, initializing the development $(\bar{V}, \bar{E}, \bar{F})$ is trivial.

Development management. As the surface and the associated mesh (V, E, F) evolves, the development $(\bar{V}, \bar{E}, \bar{F})$ has to be updated. This could be done by recomputing the development in each step. It turns out, however, that there is a faster alternative, as follows. We add to the guided projection procedure of § 3.2 the vertices $\mathbf{v}_j, \bar{\mathbf{v}}_j$ of $V, \bar{V}$, resp., as variables. We add linear relations

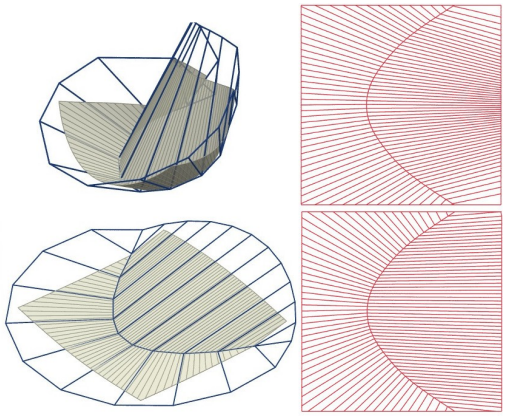


Fig. 13. Folding paper along a curved crease, maintaining isometry to the original rectangular shape, and maintaining the crease. In this case decomposition of the developable into smooth simple ruled pieces is not the most efficient way of computational handling of the problem. Instead we embed this model into a bigger surface which consists of two ruled strips and which develops onto a disk (the control net of the bigger surface is shown).

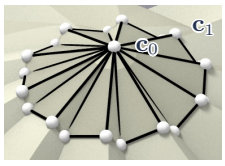
analogous to (5) which state how vertices $\mathbf{v}_j$ are found by evaluating spline curves. Further, for every face $f = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ we add equations expressing isometric development,

$$\|\mathbf{v}_i - \mathbf{v}_j\|^2 = \|\bar{\mathbf{v}}_i - \bar{\mathbf{v}}_j\|^2 \quad (1 \le i < j \le n). \tag{12}$$

Since these distance-based conditions do not prevent unwanted overfolding of the development $(\bar{V}, \bar{E}, \bar{F})$, we punish their occurrence by adding a fairness energy to our guided projection procedure, namely $\sum \|\mathbf{v}_{i+1} - 2\mathbf{v}_i + \mathbf{v}_{i-1}\|^2$ for for every discretized spline curve $\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots$. Thus the development is updated every time guided projection is called. Global developability is guaranteed if the development is maintained during modeling and if that development does not have overlaps.

Only if the surface under consideration is simply connected, it is sufficient to maintain local developability during modeling and to check for overlaps at the end. This is shown by a standard topological argument, see the appendix.

Local developability in vertices. Developability around vertices where creases meet, could be expressed by existence of the mesh-development of a 1-ring neighbourhood of that vertex within (V, E, F). There is however a superior alternative which does not suffer the discretization errors inherent in meshing. The first edge $\mathbf{c}_0\mathbf{c}_1$ of the spline control polygon of a crease curve $\mathbf{c}(u)$ is the initial tangent of that spline. Thus the condition "sum of angles between creases equals 2π" can be expressed as developability of a star of triangles, whose vertices are spline control points, and whose edges include the above-mentioned edges (see inset corresponding to Figure 6b).



Isometric mapping. We also want to be able to model surfaces generated by isometric folding and bending a prescribed domain $\bar{D}$. For that we must keep the boundary of the development $(\bar{V}, \bar{E}, \bar{F})$ confined to the boundary $\partial\bar{D}$ of that domain. Figure 12 is an example of this, and so is Figure 13: Here we maintain the development of a disk-shaped domain with one crease, and impose the condition that the boundary vertices of the development $\bar{V}$ must remain on the

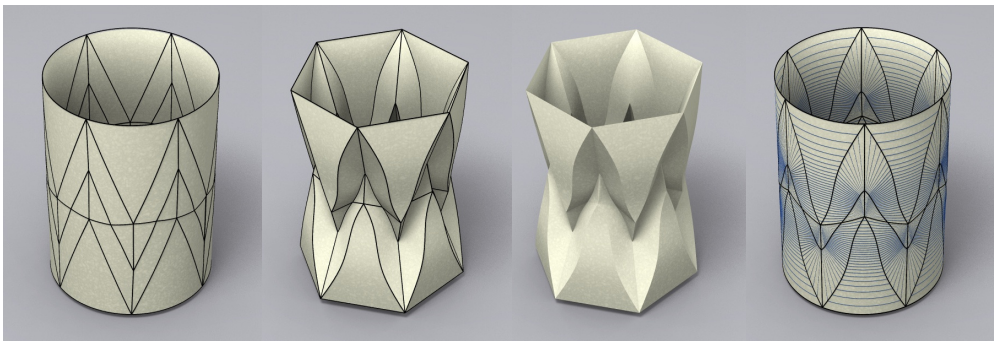


Fig. 14. Interactive modeling of a cylindrical curved-crease sculpture. During modeling we maintain the property of unfoldability onto a cylinder (this is implemented via cutting the surface open and requiring a planar development with periodicity). *From left:* a user's initial sketch of the combinatorics, the final sculpture with and without patch boundaries, and the cylindrical development.

circular boundary. Likewise, the vertices of $\bar{V}$ which correspond to the crease are confined to that crease.

This is implemented as follows: We assume a piecewise-smooth boundary $\partial\bar{D}$. Some boundary vertices $\mathbf{v}_j$ correspond to a corner of the domain. In this case the corresponding vertex $\bar{\mathbf{v}}_j$ is simply kept fixed in that corner, which amounts to two linear equations being added to the guided projection procedure. Other boundary vertices $\mathbf{v}_j$ correspond to smooth parts of $\partial\bar{D}$: for those we compute the footpoint of $\bar{\mathbf{v}}_j$ on $\partial\bar{D}$ and the tangent T_j there. The nonlinear condition "$\bar{\mathbf{v}}_j \in \partial\bar{D}$" is being linearized to "$\bar{\mathbf{v}}_j \in T_j$". This linear equation is added as an additional constraint to the guided projection procedure. T_j is updated after each round of iteration.

6. TOWARDS CURVED ORIGAMI

Curved origami refers to shapes foldable from paper along curved creases, which translates to surfaces which are globally developable, see Figure 6. Sections 2–5 in a cumulative manner collected all the algorithmic ingredients for their interactive modeling. However the individual examples below may still require further extensions and modifications of these algorithms.

Examples of simple topology. In Figure 6 we already encountered shapes which can be folded from a single sheet of paper. Since those surfaces are simply connected, local developability implies global developability (apart from overfoldings).

Example: Folding annuli. Both Figure 1 and Figure 12 show a curved-crease sculpture created from an annulus. Such surfaces are an active topic of study: For shapes obtained by folding along concentric circles, structural analysis has been performed by [Dias et al. 2012]. It has been conjectured that such surfaces do not exist in the strict mathematical sense [Demaine et al. 2011b]. This is confirmed by our own observations during modeling: If the folds are restricted to be concentric circles (by imposing distance constraints on the development's vertices), we do not achieve quite the same quality of developability as when this restriction is relaxed and is only imposed on the inner and outer boundary.

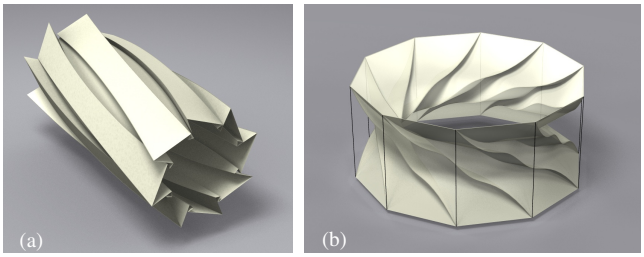


Fig. 15. (a) Creases which enjoy the "geodesic" property are as far removed from the curved-fold property as possible, but can be simulated by three curved folds in close proximity. (b) This surface, whose outer faces are planar and have been removed for better visibility, has the topology of a torus and is intrinsically flat.

Examples of cylinder topology. The curved-crease sculpture presented by Figure 14 is inspired by David Huffman's *hexagonal column* design [Demaine et al. 2011a]. This surface does not enjoy developability onto a plane, but onto a right circular cylinder of radius r. To maintain developability during modeling, we cut that cylinder open and unroll it onto the plane. The coordinates of vertices of the development $(\bar{V}, \bar{E}, \bar{F})$ are considered modulo the vector $(2\pi r, 0)$. Another cylindrical example is shown by Figure 15a.

Example: Flat torus. Figure 15b shows a curved-crease sculpture of torus topology. It has been initialized from a flat torus polyhedron, and has been modeled under constraints (2), (7) which express local developability in faces and creases. As to vertices, the Gauss-Bonnet theorem says that the surface's total curvature is zero. By symmetry, all vertices carry zero Gauss curvature.

7. DISCUSSION

Computational Efficiency. We adopt the method of Tang et al. [2014] to solve the constraint equations related to developability. They argue that their method works best if the constraints are linear or quadratic (which is true in our case), and if the system is sparse. Sparsity holds only to a limited extent, but we are still fast enough for interactive modeling — for details see Figure 17. We verified that using cubic constraints such as (2) causes bad performance: 1 order of magnitude more iterations achieve surface quality which is 1 order of magnitude less than when using quadratic constraints. More details on the comparison between quadratic and cubic constraints are provided by Tang et al. [2014].

Note in particular that using splines allows us to deal with smooth surfaces without the need to go to higher resolutions, which is different from mesh-based methods. An obvious deficiency of the spline representation in presence of non-smooth side conditions is their parametric smoothness. However, some simple tests we made in this regard were reassuring: As illustrated by Figure 16, the effect of enforced non-smoothness is that spline control points coalesce, in agreement with known properties of splines.

An important ingredient in our work is the dual representation, i.e., the normal spline. Modeling developables entirely with the dual representation is not intuitive and makes it difficult to control singularities. In combining it with the primal representation we

Fig	number of ...			weight of constraints related to ...								normal	face flatness	crease flatness	vertex flatness	T_{iter}
No.	ctrl.pts	strips	variables	(4,5)	(7)	(12)	v-dev.	isom.	**	ε_1	ε_2	spline?	$\delta_{\text{mean}}/\delta_{\max}$	$\lvert K\rvert_{\text{mean}}/\lvert K\rvert_{\max}$	$\lvert K\rvert_{\text{mean}}/\lvert K\rvert_{\max}$	[sec]
6a	91	18	12249	1		1	1			$[0.9^i/10]_{i\le 5}$	.01	yes	1.5E-4 / 5.0E-4	2.3E-4 / 7.2E-4	3.6E-5 / 4.7E-5	0.16
6b	101	20	13509	1	1	1	1			$[0.9^i/10]_{i\le 5}$	.01	yes	1.3E-5 / 6.4E-4	8.4E-6 / 2.2E-5	5.6E-4 / 5.6E-4	0.21
8	60	9	6147	1	1	1	1			$[0.9^i/10]_{i\le 5}$	.01	yes	4.7E-5 / 1.6E-4	1.0E-6 / 1.1E-5	2.8E-5 / 4.4E-5	0.06
9b	57	5	5466	1						0	.001	no	2.5E-5 / 1.7E-4	n/a	n/a	0.05
10†	303	94	12777	1					**	0.001	.001	yes	5.6E-5 / 4.0E-3	n/a	n/a	0.5
12	270	48	29574	1	1	1		1		$[0.5^i/10]_{i\le 5}$	.01	yes	1.8E-3 / 5.8E-3	1.2E-3 / 8.2E-3	n/a	0.36
13	36	2	3795	1	1	1		1		$[0.8^i/10]_{i\le 5}$	.01	yes	1.4E-4 / 4.4E-4	4.0E-5 / 6.4E-4	n/a	0.04
15a	320	32	39648	1	1						.01	yes	7.2E-5 / 7.5E-4	1.8E-5 / 7.0E-5	n/a	0.8
15b	140	30	17850	1	1	1	*				.01	yes	1.5E-5 / 2.2E-4	1.8E-6 / 1.3E-5	*	0.23
18‡	24	8	1116	1						0.001	.001	no	2.2E-4 / 7.8E-4	n/a	n/a	0.13

†*blue part* ‡*upper part* ****weights** 1E-3/1E-4/1E-2 *resp. for* (9)–(11) * *in this special case vertex flatness is implied automatically*

Fig. 17. For each example, these statistics show the number of control points, of simple spline developables ("strips"), and of variables which are manipulated by the guided projection procedure. Meshes are rescaled to have diameter 1. We list the weights given to individual constraints: developability of spline strips and of creases, global developability, local vertex-developability, isometric mapping to a prescribed development, and approximation. User-defined constraints like fixed vertices are given the weight 1. The regularizing energy frequently is set to zero (by letting $\varepsilon_1 = 0$) after 5 iterations to save time. We also show the time per iteration, referring to an Intel-core i5 3320m processor. The surface quality achieved after 10 rounds of iteration is illustrated via the auxiliary mesh (V, E, F) of § 5. We measure face flatness by $\delta = (\text{distance of diagonals})/(\text{avg. of 2 short edge lengths})$. Developability around vertices (corresponding to creases or vertices of the surface) is measured by Gauss curvature K (angle sum minus 2π).

experienced great benefits to approximation, and also to modeling complex shapes like Figure 12 (for simpler shapes, apparently the individual normal vectors which occur in our equations can be considered independent). We used C^2 cubic spline surfaces, which in theory would imply C^1 degree 4 normal splines. In practice however we restricted the normal splines to degree 3 and smoothness C^2, using fewer control points and reducing computation times to almost half. This did not impair quality.

Robustness. We made a few tests concerning the robustness of our method, and we did not experience strong sensitivity to the choice of parameters. Changing the importance of the energy functional by changing ε_1 influences the shape of the final result. E.g. in Figure 15b the curved folds become straighter if ε_1 is increased. However changing ε_1 seems to have no influence on the achieved surface quality.

The parameter ε_2 which governs the importance of the second regularizing contribution to (6) apparently can be chosen in the interval $[10^{-6}, 10^{-3}]$ (or $[10^{-6}, 10^{-2}]$, if curved-folding constraints are present) with only small changes in the final surface quality. A smaller value slightly improves the convergence rate, but we used a larger value to favor proximity to the starting configuration. In several cases documened by Figure 17 we set ε_1 to zero after 5 iterations to improve convergence and to save time, since this regularizer seems to have no effect in later stages of iteration.

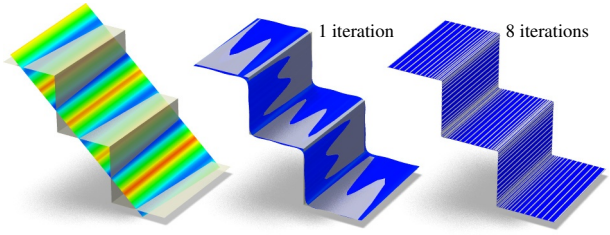


Fig. 16. Spline developables can approximate shapes with edges, if the normal spline is not enforced to have higher order smoothness and if $\mathbf{n}(u_j)$ in (11) is considered fixed in each iteration of constraint solving. This image colour codes the distance of a developable spline strip from the reference shape.

The actual values of $\varepsilon_1, \varepsilon_2$ used for the figures contained in this paper are documented by Figure 17. They are the result of numerical experiments. We observed the following trends: Firstly, a greater number of constraints (in particular curved-folding constraints) requires higher values of ε_1. Secondly some constraints like a prescribed development already have a regularizing effect, so ε_1 can be made smaller in that case.

Limitations. The limitations of our method are mostly the need to decide the "decomposition combinatorics" of a complex surface before modeling. This is felt mostly when exploring shapes with prescribed development, even if sometimes a workaround can be used (see Figure 13). This limitation applies in a similar manner to previous work, e.g. [Solomon et al. 2012]. However, having to choose combinatorics is not a limitation when our aim is to explore the possible shapes of composite developables which have exactly those prescribed combinatorics.

Conclusion. We have presented methods for interactive modeling of piecewise-developable surfaces and especially curved-crease origami. Our approach is based on a spline representation of simple developables, combined with previous work on solving certain nonlinear systems quickly (by "guided projection"). We have shown how to express local and global developability, and how to solve approximation problems with developables, utilizing their smooth and one-dimensional normal vector field. Our method is illustrated by means of different examples, some of which correspond to existing work in real-life origami.

Future work. The topic of piecewise-developables and curved-crease sculptures is not yet exhausted. We have not touched upon computation of an optimal decomposition into simple developables. Neither have we considered the topic of continuous unfolding, and we have no algorithmic treatment of overlaps in the unfolding (which would amount to collision detection in our regularly updated development). Unsolved problems remain, e.g. how to find the closest curved-crease surface which is globally developable and which is closest to a given target shape. This question includes the problem of Origami tessellations with curved folds. Finally, the present method has much potential for extensions. One is enabling real-time modeling with developables: we propose to study a multiresolution approach together with parallelization. Other di-

rections concern more flexibility in modeling by changing the combinatorics, and to include other constraints like statics or shading.

APPENDIX

Counting degrees of freedom of spline developables. The spline functions of C^k smoothness and polynomial degree m in n segments form a linear space, here denoted by $\mathcal{S}^n_{m,k}$. It has dimension $\dim \mathcal{S}^n_{m,k} = n(m+1) - (n - i_{\text{open}})(k+1)$, where $i_{\text{open}} = 1$ for open curves, and $i_{\text{open}} = 0$ for closed curves. This number follows from $m+1$ polynomial coefficients per interval, minus the conditions imposed by equality of derivatives.

Let us now impose the developability condition (2) on a surface defined by two spline curves $\mathbf{a}$, $\mathbf{b}$. The determinant $d(u)$ in (2) enjoys C^{k-1} smoothness and is piecewise-polynomial of degree $\leq 3m-2$. Actually the degree is $\leq 3m-3$, since it is not difficult to see that the coefficient of u^{3m-2} in the expansion of $d(u)$ vanishes. Thus (2) amounts to $\dim \mathcal{S}^n_{3m-3,k-1}$ conditions. Subtracting this from the $6 \dim \mathcal{S}^n_{m,k}$ variables stored in $\mathbf{a}$, $\mathbf{b}$ yields at least

$$n(3m - 5k + 2) + i_{\text{open}}(5k + 6)$$

degrees of freedom for spline developables. In the important case of cubic splines with C^2 continuity ($m = 3$, $k = 2$, see Fig. 4) we get $n + 16i_{\text{open}}$ degrees of freedom. For $n = 1$ (polynomial "Bézier" developables) the formula reduces to $3m + 8$ d.o.f.

Rational surfaces. A similar d.o.f. count yields $4m+9$ degrees of freedom for rational degree m Bézier developables, and $n(4m + 3 - 5k) + i_{\text{open}}(6 + 5k)$ degrees of freedom for a C^k NURBS developable consisting of n segments of degree m.

Remark: The d.o.f. counts for both the Bézier and the cubic spline cases have been derived by [Chu and Séquin 2002] and [Chu and Chen 2004], respectively. As n grows, cubic spline developables appear to have fewer d.o.f. than cubic spline curves themselves, which is at odds with the well-known existence of developable cylinders and cones for every spline curve. This discrepancy is explained by the fact that our d.o.f. count did not take dependencies of equations into account. The solution manifold contains components corresponding to solutions of simple geometry which have a higher dimension than its 'freeform' components.

Simple connectedness. We want to show that local developability of simply connected surfaces implies developability (if one disregards overlaps), i.e., local developments can be pasted together consistently. We must verify that sequentially pasting together the local developments of patches which cover a loop in the surface, will eventually lead back to the local development we started with. Small loops contained in a single developable patch surely are "good loops", meaning that they have this property. Loops do not lose this property upon deformation, since for small enough deformations the same patches can be used. Thus all loops deformable into small loops are good loops. In a simply connected surface, all loops can be deformed into a point, which concludes the argument.

ACKNOWLEDGMENTS

This research was supported by the DFG-Collaborative Research Center, TRR109, *Discretization in Geometry and Dynamics*, through grant I705-N26 of the Austrian Science Fund (FWF), by FWF grant No. P23735, and by the EC under grant 286426 (GEMS). C. Tang and P. Bo were supported by KAUST base funding. P. Bo wants to thank the National Natural Science Foundation of China (grant no. 61202275) and the Open Project Program of the State Key Lab of CAD&CG (grant no. A1410) of Zhejiang University. We are grateful to the anonymous reviewers and colleagues for fruitful discussions, in particular M. Bartoň, E. Demaine, M. Kilian, and E. Vouga. We want to thank Le Shi (KAUST) for the design of Figure 18.

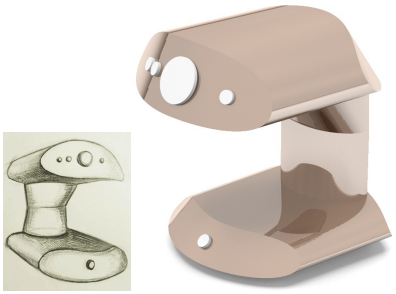


Fig. 18. A piecewise-developable modeled using the methods of this paper, following an artist's sketch (left hand image © Le Shi).

REFERENCES

Günter Aumann. 1991. Interpolation with developable Bézier patches. *Comput. Aided Geom. Des.* 8 (1991), 409–420.

Günter Aumann. 2003. A simple algorithm for designing developable Bézier surfaces. *Comput. Aided Geom. Des.* 20 (2003), 601–619.

Mohan Bodduluri and Bahram Ravani. 1993. Design of developable surfaces using duality between plane and point geometries. *Comput. Aided Des.* 25 (1993), 621–632.

Dongren Chen and Guojin Wang. 2002. Developable Bézier function surface. *Progress in Natural Science* 12, 5 (2002), 383–387.

Ming Chen and Kai Tang. 2010. A fully geometric approach for developable cloth deformation simulation. *Vis. Computer* 26 (2010), 853–863.

Chih-Hsing Chu and Jang-Ting Chen. 2004. Geometric design of uniform developable B-spline surfaces. In *Proc. DETC*. Vol. 1. 431–436. article DETC2004-57257.

Chih-Hsing Chu and Carlo Séquin. 2002. Developable Bézier patches: properties and design. *Comput. Aided Des.* 34 (2002), 511–527.

Carl de Boor. 1978. *A Practical Guide to Splines*. Springer.

Erik Demaine, Martin Demaine, Vi Hart, Gregory Price, and Tomohiro Tachi. 2011b. (Non)existence of pleated folds: how paper folds between creases. *Graphs and Combinatorics* 27 (2011), 377–397.

Erik Demaine, Martin Demaine, and Duks Koschitz. 2011a. Reconstructing David Huffman's Legacy in Curved-Crease Folding. In *Origami*^{5}, Patsy Wang-Iverson et al. (Eds.). A. K. Peters, 39–52.

Marcelo Dias, Levi Dudte, L. Mahadevan, and Christian Santangelo. 2012. Geometric Mechanics of Curved Crease Origami. *Phys. Rev. Lett.* 109, 114301 (2012), 1–13.

Manfredo do Carmo. 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall.

William Frey. 2004. Modeling buckled developable surfaces by triangulation. *Comput. Aided Des.* 36, 4 (2004), 299–313.

Josef Hoschek and Helmut Pottmann. 1995. Interpolation and approximation with developable B-spline surfaces. In *Mathematical methods for curves and surfaces*, M. Dæhlen et al. (Eds.). 255–264.

David A. Huffman. 1976. Curvature and creases: A primer on paper. *IEEE Trans. Computers* 25 (1976), 1010–1019.

Dan Julius, Vladislav Kraevoy, and Alla Sheffer. 2005. D-Charts: Quasi-Developable Mesh Segmentation. *Comput. Graph. Forum* 24, 3 (2005), 581–590. Proc. Eurographics.

Yannick L. Kergosien, Hironobu Gotoda, and Tosiyasu L. Kunii. 1994. Bending and Creasing Virtual Paper. *Comput. Graph. Appl.* 14 (1994), 40–48.

Martin Kilian, Simon Flöry, Zhonggui Chen, Niloy Mitra, Alla Sheffer, and Helmut Pottmann. 2008. Curved Folding. *ACM Trans. Graph.* 27, 3, Article 75 (2008), 9 pages. Proc. SIGGRAPH.

Johann Lang and Otto Röschel. 1992. Developable $(1, n)$-Bézier Surfaces. *Comput. Aided Geom. Des.* 9 (1992), 291–298.

Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.* 25, 3 (2006), 681–689. Proc. SIGGRAPH.

Takashi Maekawa and Julie S. Chalfant. 1998. Design and tessellation of B-spline developable surfaces. *J. Mech. Des.* 120 (1998), 453–461.

Sandy Martedi and Hideo Saito. 2011. Foldable augmented papers with a relaxed constraint. In *Proc. ISAS*. IEEE, 127–131.

Meher McArthur and Robert J. Lang. 2012. *Folding Paper: The infinite possibilities of Origami*. Int. Arts & Artists, Wash. DC.

Neil Meredith and James Kotronis. 2012. Self-detailing and self-documenting systems for wood fabrication: The Burj Khalifa. In *Advances in Architectural Geometry 2012*, Lars Hesselgren et al. (Eds.). Springer, 185–198.

Jun Mitani. 2012. Origami Applications. *http://mitani.cs.tsukuba.ac.jp/origami_application*. (2012).

Jun Mitani and Takeo Igarashi. 2011. Interactive Design of Planar Curved Folding by Reflection. In *Pacific Graphics, Short Papers*. 77–81.

Jun Mitani and Hiromasa Suzuki. 2004. Making papercraft toys from meshes using strip approximate unfolding. *ACM Trans. Graph.* 23, 3 (2004), 259–263. Proc. SIGGRAPH.

Rahul Narain, Tobias Pfaff, and James F. O'Brien. 2013. Folding and Crumpling Adaptive Sheets. *ACM Trans. Graph.* 32, 4, Article 51 (2013), 8 pages. Proc. SIGGRAPH.

Francisco Pérez and José Antonio Suárez. 2007. Quasi-developable B-spline surfaces in ship hull design. *Comp. Aided Geom. Design* 39 (2007), 853–862.

Mathieu Perriollat and Adrien Bartoli. 2012. A computational model of bounded developable surfaces with application to image-based three-dimensional reconstruction. *Comput. Anim. Virt. Worlds* 24, 5 (2012), 459–476.

Martin Peternell. 2004. Developable Surface Fitting to Point Clouds. *Comput. Aided Geom. Des.* 21 (2004), 785–803.

Helmut Pottmann and Gerald Farin. 1995. Developable rational Bézier and B-spline surfaces. *Comput. Aided Geom. Des.* 12 (1995), 513–531.

Helmut Pottmann, Alexander Schiftner, Pengbo Bo, Heinz Schmiedhofer, Wenping Wang, Niccolo Baldassini, and Johannes Wallner. 2008. Freeform surfaces from single curved panels. *ACM Trans. Graph.* 27, 3, Article 76 (2008), 10 pages. Proc. SIGGRAPH.

Helmut Pottmann and Johannes Wallner. 1999. Approximation algorithms for developable surfaces. *Comput. Aided Geom. Des.* 16 (1999), 539–556.

Helmut Pottmann and Johannes Wallner. 2001. *Computational Line Geometry*. Springer.

Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paul Cani, and Boris Thibert. 2007. Developable surfaces from arbitrary sketched boundaries. In *Symp. Geom. Processing*, Alexander Belyaev and Michael Garland (Eds.). 163–172.

Justin Solomon, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. 2012. Flexible Developable Surfaces. *Comp. Graph. Forum* 31, 5 (2012), 1567–1576. Proc. Symposium Geometry Processing.

Tomohiro Tachi. 2010. Origamizing polyhedral surfaces. *IEEE Trans. Vis. Comp. Graphics* 16, 2 (2010), 298–311.

Tomohiro Tachi and Koryo Miura. 2012. Rigid-Foldable Cylinders and Cells. *J. Int. Assoc. Shell & Spatial Structures* 53, 4 (2012), 217–226.

Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. 2014. Form-finding with Polyhedral Meshes Made Simple. *ACM Trans. Graph.* 33, 4, Article 70 (2014), 9 pages. Proc. SIGGRAPH.

Charlie Wang and Kai Tang. 2004. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *Vis. Computer* 20 (2004), 521–539.

K Wang and Y Chen. 2011. Folding a Patterned Cylinder by Rigid Origami. In *Origami 5*, Patsy Wang-Iverson et al. (Eds.). A.K. Peters, 265–276.

Wenping Wang, Helmut Pottmann, and Yang Liu. 2006. Fitting B-spline Curves to Point Clouds by Curvature-based Squared Distance Minimization. *ACM Trans. Graph.* 25, 2 (2006), 214–238.

Hitoshi Yamauchi, Stefan Gumhold, Rhaleb Zayer, and Hans-Peter Seidel. 2005. Mesh segmentation driven by Gaussian curvature. *Vis. Computer* 21 (2005), 659–668.

Received April 2015; accepted September 2015