Developable Quad Meshes

VICTOR CEBALLOS INZA, King Abdullah University of Science and Technology, Saudia Arabia
FLORIAN RIST, King Abdullah University of Science and Technology, Saudia Arabia
JOHANNES WALLNER, Technische Universität Graz, Austria
HELMUT POTTMANN, King Abdullah University of Science and Technology, Saudia Arabia

Fig. 1. We make surfaces developable by imposing a novel local condition on the edges and faces of a quad mesh. That mesh does not have to be aligned with any special curves on the developable surface. For these two examples we take a surface represented by a quad mesh, perform a partition into strips along mesh polylines, and make individual strips developable while boundaries remain fixed. We also show the Gauss image, whose curve components correspond to the developable parts of the surface.

There are different ways to capture the property of a surface being developable, i.e., it can be mapped to a planar domain without stretching or tearing. Contributions range from special parametrizations to discrete-isometric mappings. So far, a local criterion expressing developability of general quad meshes has been lacking. In this paper we propose a new and efficient discrete developability criterion which is based on a property well known from differential geometry, namely a rank deficient second fundamental form. This criterion is expressed in terms of the canonical checkerboard patterns inscribed in a quad mesh which already was successful in describing discrete-isometric mappings. In combination with standard global optimization procedures we are able to perform developable lofting, approximation and design. The meshes we employ are combinatorially regular quad meshes with isolated singularities but are otherwise not required to follow any special curves. They are thus easily embedded into a design workflow involving standard operations like remeshing, trimming and merging operations.

1 INTRODUCTION

The numerical and geometric modeling of developable surfaces has attracted attention for many years, starting with the first mesh representations proposed by R. Sauer, cf. the textbook [Sauer 1970]. One reason for that is the great practical importance of developables, which represent shapes made by bending flat pieces of inextensible sheet material. New algorithms continue to emerge, as do new applications — we only point to the construction of metamaterials that can be speedily laser-cut and fill volumes in the manner of ruffles by Signer et al. [2021], and the use of developables in interactive physical book simulation by Wolf et al. [2021].

A developable surface is unanimously defined by the existence of local isometric mappings to planar domains. As it turns out, the physical reality of bending thin sheets is modelled by surfaces exhibiting piecewise $C^2$ smoothness, which
means surfaces exhibiting curvature continuity except for creases along curves. Geometric modeling of developables has been confined to this case. The frequency of new proposals for the computational treatment of developables is a sign that the problem has not yet been satisfactorily solved. Another reason probably is the mathematical complexity of the subject itself, as well as the fact that $C^2$ developables enjoy many different but equivalent characterizations. These include vanishing Gauss curvature or other equivalent infinitesimal properties; line contact with tangent planes; or existence of a local planar development. Each of these properties has been the basis of a computational approach, and each serves as motivation for defining a certain kind of discrete developable surface. This is also true for the present paper: We use the fact that developability is characterized by a rank deficient second fundamental form, and we employ the checkerboard patterns proposed by Peng et al. [2019] to express this in a discrete way.

1.1 Overview and Contributions

- We present a new quad-mesh based discrete model of developable surfaces which does not require the use of a development. Neither do edges have to be aligned with special curves on the surface under consideration. It is based on a certain way of expressing local rulings in terms of a checkerboard pattern (§ 2.2).
- We achieve developability by global optimization, essentially performing a projection onto the constraint manifold. That projection is guided by soft constraints like fairness and proximity to a reference surface. When needed, we can combine our method with the isometric mappings proposed by Jiang et al. [2020]. This is easy to do, since this method, too, is based on checkerboard patterns (§ 2.3).
- Interactive design of developables can mean the isometric deformation of a given flat piece as well as generally modifying a design surface such that developability is maintained as a constraint. We are able to do both (§ 3).
- We can approximate surfaces with piecewise developables. The viewpoint assumed by this paper is that the decomposition into developables needs to be influenced by a designer (see Fig. 1). This is made possible by our developability condition being almost invariant w.r.t. remeshing and similar operations, so we are handling a single consistent quad mesh during the entire workflow (§ 4.1).
- Modeling tools treated in this paper include developable lofting, which is an old problem not easily accessible with previous methods. A user can interactively modify developables by pulling on handles and letting developables glide through guiding curves; attaching a developable patch to a surface; and positioning singular curves.
- While the main distinction of our method compared to previous work is its modeling capability, a direct comparison shows that also approximation performs better than with recent automatic methods.

1.2 Previous Work

There is a large body of literature about geometric modelling of and with developable surfaces. Our brief overview is subdivided according to the way developables are represented, either as spline surfaces or as discrete surfaces.

1.2.1 Previous work based on splines. It is known that developables consist of ruled surfaces enjoying torsality, i.e., the tangent plane along a ruling is constant. A ruled surface can be modelled as a degree $(1, n)$ B-spline surface — one family of parameter lines then will be the surface’s rulings. Torsality is a nonlinear constraint [Lang and Röschel 1992] that can be achieved by optimization, see e.g. [Jiang et al. 2019; Tang et al. 2016]. One limitation of such a method is the necessity to decompose developables into ruled pieces. The method thus may not be suitable for modelling deformations of developables where that decomposition may change.
A torsal ruled surface is the envelope of its tangent planes, and thus essentially is a curve in the dual space of planes. This fact has been exploited by [Bodduluri and Ravani 1993] and follow-up publications. It reduces the design of ruled developables to the design of curves. The drawback of this method, in addition to the one mentioned in the previous paragraph, is that working in plane space is not intuitive and does not naturally avoid singularities.

Finally we point to Jiang et al. [2020] who impose developability on spline surfaces in an approximate way via conversion to a quad mesh. Here rulings do not have to coincide with parameter lines (see §1.2.2 below).

1.2.2 Previous work based on quad meshes. The textbook [Sauer 1970] proposes discrete developables based on the fact that developables are the envelopes of their tangent planes: A discrete ruled developable is simply a sequence of flat quads, and the edges between them play the role of rulings. This property lies on the basis of quad-meshing of developables, see recent work by Verhoeven et al. [2022]. Such ruling-based developables are the basis of work by Liu et al. [2006] and Solomon et al. [2012]. Their disadvantages are the same as for other ruling-based methods: it is difficult to model situations where the ruling pattern of a developable changes.

A different characterization of developables, the existence of a network of orthogonal geodesics, is the basis of work by Rabinovich et al. [2018a; 2018b; 2019] and Ion et al. [2020]. Here developables are represented by quad meshes whose edges are not necessarily aligned with the rulings, but nevertheless are in special position — they discretize a network of orthogonally intersecting geodesics.

Jiang et al. [2020] use discrete-isometric mappings to handle developables: A surface is represented by a quad mesh whose edges do not have any special relation to the surface. Developability is imposed by maintaining a second mesh in \( \mathbb{R}^2 \) isometric to the first mesh. Similarly, the work by Chern et al. [2018] on isometric mappings is capable of handling developables.

1.2.3 Previous work based on other discretizations. A triangle mesh is intrinsically flat except at the vertices, and it is in fact locally flat if and only if the angle sum in its vertices equals \( 2\pi \). This natural discretization of the concept of developable surface has not been in much use in geometric modeling. An exception is provided by meshes consisting of equilateral triangles as used by [Jiang et al. 2015]. Better suited for geometric design of developables is the ‘local hinge’ condition imposed by Stein et al. [2018]. It ensures existence of rulings and also implies that the position of edges is not arbitrary; every face has an edge which represents the direction of a ruling.

Binninger et al. [2021] approximate surfaces with piecewise-developable ones by thinning the Gauss image; their method operates on a triangle mesh.

Sellán et al. [2020] use an entirely different way of imposing developability. A height field \( z = f(x, y) \) represents a developable surface if and only if the Hessian of \( f \) has vanishing determinant. A certain convex optimization converts a given height field to one where the Hessian is nonsingular only along 1D curves. This leads to a piecewise-developable surface.

2 DISCRETE DEVELOPABLES

2.1 Differential Geometry of Smooth Developables

We consider surfaces that are piecewise curvature continuous and which enjoy the property of being locally isometric to a planar domain. It is well known (see e.g. [Guggenheimer 1963]) that this intrinsic flatness is characterized by the vanishing of Gauss curvature, \( K = 0 \). It is also well known that one family of principal curvature lines of such surfaces are straight, and that the tangent plane along these straight lines (the rulings) is constant. The rulings extend all the way
Developable Quad Meshes

to the boundary of the surface. A developable thus decomposes into ruled surface pieces and planar parts. In geometric modeling, the number of pieces is assumed to be finite. We also consider surfaces consisting of individual developable pieces.

**Gauss Image and 2nd Fundamental Form.** The Gauss image of a developable is the set of its unit normal vectors. It is contained in the unit sphere $S^2$, and it decomposes into curves, one for each ruled piece.

In each point $p$ of the surface, the second fundamental form $\text{II}_p(v, w)$ governs curvatures. It takes as arguments vectors $v, w$ tangent to the surface. If $x(u, v)$ is a parametrization, and $n(u, v)$ is the corresponding unit normal vector field, then the second fundamental form relates their derivatives via

$$\text{II}_p(x_u, x_u) = \langle x_u, -n_u \rangle,$$
$$\text{II}_p(x_u, x_v) = \langle x_v, -n_u \rangle,$$
$$\text{II}_p(x_v, x_v) = \langle x_v, -n_v \rangle.$$

By linearity, $\text{II}_p$ is now defined for all tangent vectors.

**The Conjugacy Relation and Developability.** Tangent vectors $v, w$ attached to the same point $p$ are called conjugate, if

$$\text{II}_p(v, w) = 0.$$

Gauss curvature vanishes in $p$ if and only if $\text{II}_p$ has rank less than 2 and exhibits a kernel (the ruling) which is conjugate to all tangent vectors. This can equivalently be expressed by existence of a tangent vector which obeys

$$\text{II}(x_u, r) = \text{II}(x_v, r) = 0 \quad (r \neq 0).$$

(1)

There is another property relating conjugacy of tangent vectors with developable surfaces: Suppose we have a curve $c(t)$ contained in a surface and a vector field $r(t)$ which is conjugate to the derivative $\varepsilon(t)$. Then the ruled surface $y(t, s) = c(t) + sr(t)$ is tangent to the original surface along the curve $c$, and is itself developable, see e.g. [Liu et al. 2006]. It is the envelope of the surface’s tangent planes along the curve $c$, and it is a geometric fact that the rulings of this envelope are conjugate to the tangents of the curve — see Figure 2.

![Fig. 2. A developable touching another surface along a curve $c(t)$. Its rulings $r(t)$ are conjugate to the derivative vectors $\varepsilon(t)$.](image)

### 2.2 Developable Quad Meshes

It is our aim to express developability as a local property of a quad mesh whose edges and faces are allowed to be arbitrary. In contrast to previous work [Liu et al. 2006; Sauer 1970] we do not require the faces to be planar, nor do we require the edges to be aligned with special curves on the surface, as is done by the previous references and by [Rabinovich et al. 2018b; Stein et al. 2018]. We also do not need an isometric mapping to a planar domain.
2.2.1 Conjugacy and Ruling Line Fields. We propose a developability condition which uses the checkerboard pattern associated with the quad mesh under consideration, see [Jiang et al. 2020]. For each edge we consider the midpoint, and for each face \( f \) the inscribed parallelogram formed by those edge midpoints, see Figure 3.

![Fig. 3. The checkerboard pattern associated with a quad mesh contains, for each face \( f \), the parallelogram formed by edge midpoints of \( f \). It features a center \( b_f \) and a unit normal vector found by normalizing the cross product \((v_2 - v_0) \times (v_3 - v_1)\). Its plane \( \tau_f \) serves as a tangent plane.](image)

We think of the given quad mesh as approximating a smooth surface. For each face \( f \), the plane containing the inscribed parallelogram represents a tangent plane \( \tau_f \). Every face is thus naturally equipped with a unit normal vector \( n_f \) which we would like to attach to the midpoint \( b_f \) of the inscribed parallelogram. We assume that all these unit normal vectors are consistently oriented and point to one side of the mesh.

![Fig. 4. The discrete version of Fig. 2. The strip of faces \( f_0, f_1, \ldots \) between two mesh polylines represents a developable tangentially circumscribed to a surface along a curve. Inscribed parallelograms span tangent planes \( \tau_{f_i} \), which in turn envelop a developable with rulings \( r_{e_i} = \tau_{f_i} \cap \tau_{f_{i+1}} \).](image)

Consider a sequence of faces \( f_0, f_1, \ldots \) which have common edges \( f_i \cap f_{i+1} \), see Figure 4. Successive inscribed parallelograms meet at the midpoint of the edge \( f_i \cap f_{i+1} \). We now perform a construction which is a discrete analogue of the envelope of tangent planes shown by Figure 2. Successive discrete tangent planes \( \tau_{f_i}, \tau_{f_{i+1}}, \ldots \) associated to points \( b_{f_0}, b_{f_1}, \ldots \) intersect in lines

\[
 r_{e_i} = \tau_{f_i} \cap \tau_{f_{i+1}}, \quad \text{where} \ e_i = f_i \cap f_{i+1}.
\]

The lines \( r_{e_i} \) are the discrete rulings of the discrete envelope of tangent planes along the discrete curve \( b_{f_0}, b_{f_1}, \ldots \) [Sauer 1970]. The direction of the ruling \( r_{e_i} \) is indicated by a ruling vector associated with an oriented edge (half edge)
Developable Quad Meshes

\( \tilde{e}_i \). If \( \tilde{e}_i \) and \(-\tilde{e}_i\) are the two half-edges corresponding to the edge \( e_i = f_i \cap f_{i+1} \), we let

\[
r_{\tilde{e}_i} = n_{f_i} \times n_{f_{i+1}}.
\]

(2)

Here we assume that \( f_i \) is to the left and \( f_{i+1} \) to the right, and we also have \( r_{-\tilde{e}_i} = -r_{\tilde{e}_i} \). In analogy to the smooth case, we postulate:

\[
e_i = f_i \cap f_{i+1} \implies r_{\tilde{e}_i}, b_{f_i} - b_{f_{i+1}} \text{ are conjugate}.
\]

The cross product in (2) is zero if neighbouring tangent planes \( \tau_{f_i}, \tau_{f_{i+1}} \) coincide. This happens e.g. if the mesh is flat, or the strip under consideration is flat.

Fig. 5 shows what happens when we assign a ruling \( r_e \) to all edges of a quad mesh.

![Fig. 5. Left: On a general surface, the rulings associated with edges correspond to two distinct line fields (yellow and red). Right: This mesh represents a developable surface, since these line fields coincide.](image)

This way of computing rulings makes sense because we are only interested in fair meshes which discretize smooth surfaces. We consider only such meshes where the edge polylines themselves are fair, so that they locally could be interpreted as a discrete version of the parameter lines of a \( u,v \) parametrization (which is no restriction). Only in that case (2) computes a ruling \( r_e \) conjugate to the edge \( e \).

2.2.2 Definition of Developable Quad Meshes. We express developability via the following property: A surface is developable if and only if its 2nd fundamental forms do not have full rank. For a discrete version of this statement it is convenient to associate part of a 2nd fundamental form with each face of the mesh.

Consider a face \( f = v_0v_1v_2v_3 \) with its boundary cycle \( \tilde{e}_0, \ldots, \tilde{e}_3 \), assuming \( e_i = f \cap f_i \). We define edge vectors \( e_i = v_{i+1} - v_i \) (indices modulo 4). We use (2) to compute vectors \( r_{\tilde{e}_0}, \ldots, r_{\tilde{e}_3} \) indicating rulings:

\[
r_{\tilde{e}_i} = n_f \times n_{f_i}.
\]

Figure 6 shows this configuration. Note that the ruling vectors associated with opposite edges point in the opposite direction, like the edge vectors themselves.

We now partly define the matrix \( II_f \) of a 2nd fundamental form attached to the barycenter \( b_f \) of the face \( f \). We already know vectors \( r_{e_i} \) conjugate to edge vectors \( e_i \). We now combine information attached to opposite edges by
averaging: Average edge vectors \( \frac{1}{2}(\mathbf{e}_1 - \mathbf{e}_3) \) and \( \frac{1}{2}(\mathbf{e}_0 - \mathbf{e}_2) \) are conjugate to average ruling vectors \( \frac{1}{2}(\mathbf{r}_1 - \mathbf{r}_3) \) and \( \frac{1}{2}(\mathbf{r}_0 - \mathbf{r}_2) \), respectively. We thus write

\[
(\mathbf{e}_1 - \mathbf{e}_3)^T \mathbb{II}_f (\mathbf{r}_1 - \mathbf{r}_3) = (\mathbf{e}_0 - \mathbf{e}_2)^T \mathbb{II}_f (\mathbf{r}_0 - \mathbf{r}_2) = 0.
\]

These two linear conditions imposed on the 3 entries of \( \mathbb{II}_f \) determine \( \mathbb{II}_f \) uniquely up to a factor. We do not need to perform this computation here, which would require to express all involved vectors in some appropriate basis. We are now able to define:

**Definition.** A quad mesh is developable if for all faces \( f \) the second fundamental form \( \mathbb{II}_f \) does not have full rank:

\[
\det(\mathbb{II}_f) = 0.
\]
It follows directly from (3) that $\det II_f$ is rank-deficient (i.e., $\det II_f = 0$) if and only if

$$ (r_{z_1} - r_{z_2}) \times (r_{z_3} - r_{z_4}) = 0. \tag{4} $$

A further equivalent condition is

$$ \det (r_{z_1} - r_{z_2}, r_{z_3} - r_{z_4}, n_f) = 0. $$

**Remark 1.** As clearly indicated by Fig. 6, the ruling vectors $r_{z_i}$ associated with the oriented edges of a mesh are not samples of a continuous vector field. However in the case of a developable mesh they indicate the kernel of the 2nd fundamental form, so they correspond to a single continuous line field.

**Remark 2.** In planar parts of a developable, all rulings vectors $r_{z_i}$ vanish and (4) is fulfilled automatically. The same is true along an inflection ruling of a developable: Fig. 7 shows that already close to the inflection ruling, vectors $r_{z_i}$ approach zero. An individual ruling vector $r_{z_i}$ vanishes if and only if the normal vectors $n_f, n_{f_i}$ are parallel. This situation happens if faces $f, f_i$ are arranged along a ruling contained in the developable mesh; similar to the previous special cases it is consistent with Equation (4).

**Remark 3.** Equation (4) has interesting consequences for the Gauss image. We illustrate this in the generic case (away from an inflection ruling), where a developable locally is convex. Equation (4) is expressed by the condition that

$$ n_f \times (n_{f_1} - n_{f_2}), \quad n_f \times (n_{f_3} - n_{f_4}) $$

are parallel.

All vectors involved here approximately lie in a plane orthogonal to $n_f$ (namely the unit sphere’s tangent plane in $n_f$). If the quad $n_{f_1}, \ldots, n_{f_4}$ were exactly planar, then Equation (4) would express the fact that diagonals are parallel (see inset figure; diagonals are yellow). This characterizes quads of zero oriented area, and is nicely consistent with the fact that the Gauss image of a developable is curve-like and has zero area.

### 2.2.3 Singularities of Developable Quad Meshes.

Developables can exhibit geometric singularities like a cone’s vertex or the line of regression in a torsal ruled surface. The criterion (4) does not prevent singularities from emerging, and in fact extremely singular meshes formed by the tangents of a curve may well enjoy discrete developability. We prevent such singularities by fairness imposed on the normal vector field $n_f$. Combinatorial singularities do not pose a problem (see Fig. 8), since also in this case the geometric interpretation of (4) remains valid.

**Fig. 8.** *Singularities. Left:* We show a discrete-developable mesh and the good behaviour of ruling vectors $r_{z_i}$ in the vicinity of combinatorial singularities (red). *Right:* This mesh exhibits a geometric singularity typical for developables, namely a sharp curve of regression. The singularity is not detectable via checking fairness of mesh polylines and is made visible by clipping by a plane. The mesh also fulfills the developability constraint (4).
All our computations are based on an optimization procedure which achieves constraints and yields low values of fairness functionals. We operate with a quad mesh \((V, E, F)\) which is thought to follow the parameter lines of a smooth surface. We have regular grid combinatorics except for isolated singularities, and we assume that we have a consistent orientation of faces. Our variables are the coordinates of each vertex \(v \in V\); a unit normal vector \(n_f\) of each face \(f \in F\), and a ruling vector \(r_e\) for each oriented edge \(e \in E\).

The normal vector \(n_f\) is initialized according to Fig. 3 (taking the orientation into account). During optimization, we use the constraints
\[
\begin{align*}
\c_{\text{norm},1}(f) &:= \langle n_f, v_2 - v_0 \rangle = 0, \\
\c_{\text{norm},2}(f) &:= \langle n_f, v_3 - v_1 \rangle = 0, \quad \text{where} \ f = v_0v_1v_2v_3, \\
\c_{\text{norm},3}(f) &:= \|n_f\|^2 - 1 = 0.
\end{align*}
\]
These normal vectors are recomputed after each round of optimization, since we cannot be sure that the implicit conditions above are sufficient to keep a consistent orientation. For every oriented edge (half edge) \(e\), we have a ruling vector \(r_e\) defined by the constraint
\[
\c_{\text{rul}}(e) := r_e - n_f \times n_{f'} = 0,
\]
where \(f\) to the left and \(f'\) to the right of the half-edge \(e\). Developability is expressed by Equ. (4):
\[
c_{\text{dev}}(f) = (r_{e_1} - r_{e_3}) \times (r_{e_0} - r_{e_2}) = 0,
\]
expresses the boundary cycle of \(f\). The constraints above are combined into energy functionals,
\[
\begin{align*}
E_{\text{norm}} &= \sum_{f \in F} \sum_{k=1,2,3} \c_{\text{norm},k}(f)^2, \\
E_{\text{rul}} &= \sum_{e} \c_{\text{rul}}(e)^2, \quad E_{\text{dev}} = \sum_{f \in F} \c_{\text{dev}}(f)^2.
\end{align*}
\]
To ensure that the mesh polylines approximate smooth parameter lines of a surface, we employ fairness functionals: We define
\[
E_{V, \text{fair}} = \sum_{\text{triples } v_i, v_j, v_k} \|v_i - 2v_j + v_k\|^2,
\]
where the sum is over all triples \(v_i, v_j, v_k\) of successive vertices on a discrete parameter polyline. Likewise we measure fairness of the normal field by the energy
\[
E_{n, \text{fair}} = \sum_{\text{triples } f_i, f_j, f_k} \|n_{f_i} - 2n_{f_j} + n_{f_k}\|^2.
\]
Here the sum is over all triples of successive faces arranged in a strip like shown in Fig. 4. We found that imposing fairness on the normal vector field prevents singularities like the one in Fig. 8, right.

Even if our method is not based on isometries, it is nevertheless convenient to have constraints at our disposal which express isometry between faces \(f, f'\). They can be used to make the surface of interest isometric to a given surface. With a low weight, such terms have a regularizing effect. Following [Jiang et al. 2020], we express isometry between
Developable Quad Meshes

Fig. 9. A simple developable computed with our method. Top row: An optimized quad mesh and its Gauss image consisting of face normal vectors $\mathbf{n}_f$. Bottom row: A smooth developable contains rulings. We verify this property by showing straight lines that fit the optimized mesh. They follow the discrete ruling vector field $\mathbf{r}_g$ of which a detail is shown at right.

faces $f = v_0v_1v_2v_3$ and $f' = v'_0v'_1v'_2v'_3$ by

$$c_{iso,1}(f,f') := \|v_2 - v_0\|^2 - \|v'_2 - v'_0\|^2 = 0,$$

$$c_{iso,2}(f,f') := \|v_3 - v_1\|^2 - \|v'_3 - v'_1\|^2 = 0,$$

$$c_{iso,3}(f,f') := (v_2 - v_0, v_3 - v_1) - (v'_2 - v'_0, v'_3 - v'_1) = 0.$$

We form the energy

$$E_{iso} = \sum_{(f,f')} \sum_{k=1,2,3} c_{iso,k}(f,f')^2$$

Summing up, in our optimization we minimize the functional

$$E = E_{\text{norm}} + w_{rul} E_{rul} + w_{\text{dev}} E_{\text{dev}} + w_{\text{fair}}^V E_{\text{fair}}^V + w_{\text{fair}}^R E_{\text{fair}}^R + w_{iso} E_{iso}.$$

The weights have to be chosen according to the particular application – see Table 1. In the last steps of the iteration, weights of terms used for regularization are set to 0. This enables $E_{\text{dev}}$ to approach zero itself. Table 1 refers to the status immediately before that.
Table 1. Optimization Details. For selected examples, we give the number of faces (the number of variables is roughly 18 times this), weights used in optimization, the number of individual surfaces this example consists of, the number of iterations (resp. an average number of iterations, if marked by "∼"), the average time in seconds needed for a single iteration and a single surface, and the total time used for optimization. The face pairs \((f, f')\) occurring in \(E_{\text{iso}}\) are a face \(f\) of the current mesh and its corresponding face \(f'\) in the previous iteration.

3 DESIGN TOOLS FOR DEVELOPABLE SURFACES

We here discuss several tools for modeling developables. Generally, design constraints are expressed by energies that can be added to the total energy.

3.1 Interactive Editing

A basic way of design is the interactive manipulating of a surface by pulling on handles, and by requiring that certain vertices stay close to prescribed positions. This is easily handled by adding an energy of the form \(E_{\text{pos}} = \sum \|v_i - v_i^0\|^2\) to the total energy defined by (5). Here the sum is over all vertices \(v_i\) for which target positions \(v_i^0\) are available. Fig. 10 shows an example.

Gliding Constraint. Another basic design requirement would be that our surface \(M\) is to glide through a reference shape represented by a point cloud \(\Phi\) (e.g. a curve). For all \(p \in \Phi\) we compute the closest point projection \(p^* \in M\) which is contained in some face \(f(p)\). We now require that \(p\) does not deviate from the tangent plane associated with this face, which passes through the face midpoint \(b_{f(p)}\) and has normal vector \(n_{f(p)}\). This is expressed by a low value of the energy

\[
E_{\text{pos}} = \sum_{p \text{ active}} (p - b_{f(p)}, n_{f(p)})^2
\]
The sum is over all “active” points \( p \) in the cloud \( \Phi \) (which are not variables), where “active” means they are not too far from the variable mesh \( M \). The faces \( f(p) \) are recomputed in each round of the optimization. The approximation of the distance field of \( M \) by distances to tangent planes is done on basis of [Pottmann et al. 2006]. It is known to be accurate of 2nd order in the case of zero distance, and it prevents unwanted effects if the pool of “active” points in \( \Phi \) is not entirely correct. Figure 11 shows an example where a developable is gliding along a curve.

**Soft Isometry Constraints as Regularizers.** In interactive modelling applications, we offer to the designer different kinds of material behaviour as illustrated by Fig. 10, if desired.

- If the design surface \( M \) is to behave like an inextensible material, geometric design can be done e.g. with the method of [Jiang et al. 2020]. The new approach of the present paper is not needed.

- The design surface may behave in an elastic way. Then it is modelled as a quad mesh which enjoys discrete developability as defined above. In our optimization we include the property of being isometric to a reference mesh, but with a small weight. We do not claim to accurately model any exact material behaviour here, we simply want to provide this feature to the designer and be able to profit from its regularizing effect.

- The design surface experiences “plastic” deformation. This is similar to the “elastic” case. However in each round of optimization we include isometry to the previous state as a soft constraint. This has a regularizing effect without pulling the surface back to its initial position.

**Fig. 11. Gliding Constraint.** 3 positions of a developable gliding along a curve \( \Phi \)

**Fig. 12. Developable Lofting.** The surface with cylinder topology on the left is optimized for developability so that the two boundary curves remain unchanged.
3.2 Developable Lofting

A basic way how a designer may specify a developable is to prescribe two boundary curves – see Fig. 12 for an example. This procedure is referred to as lofting. Developable lofting is a problem with a long history, and early solutions for special cases. Within the framework of our optimization, we set up lofting as follows: We connect the two given curves by an arbitrary quad mesh (e.g. by a ruled surface) which we subsequently optimize. Actually, lofting is the basis of the more complicated approximation tasks discussed in the next section.

Previous approaches to discrete developables cannot take this road easily:

• The semidiscrete representation of piecewise-developables by Pottmann et al. [2008] uses quad meshes with planar faces. Edges correspond either to boundary curves or rulings. Apart from the difficulty of having planar faces, such a mesh also cannot describe strips that are cut off in an arbitrary way, not exactly along a ruling. Since in our approach rulings are not aligned with edges, such problems do not occur.

• The ruling-based method of Tang et al. [2016] suffers the same deficiencies.

• The methods based on orthogonal geodesic nets like [Rabinovich et al. 2018a] and follow-up work cannot describe a collection of developable strips without trimming, since the boundary curves in general are not geodesics.

• The method of Jiang et al. [2020], which is based on isometries, cannot easily solve examples like that of Fig. 12. This is because the development (which does not even exist globally) has to be initialized and then optimized simultaneously with the surface.

These items might look like trivial or engineering problems but they nevertheless make a big difference.

Fig. 13. Developable lofting of skew straight lines. Here we try to achieve developability despite well known obstacles to the existence of developable lofting (if a developable surface contains straight lines in addition to its rulings, it necessarily is planar). Our algorithm computes a solution with boundary singularities. We also show a detail of the ruling line field $r_e$.

Solvability of the Lofting Problem. Lofting is actually a difficult problem, which does not need to have a smooth solution, but has many continuous solutions (e.g. a union of cones whose vertices lie on the given boundary curves in an alternating way). Our algorithm also in such cases will try to find a solution which enjoys discrete developability and such that $E^V_{fair}$ and $E^n_{fair}$ are small. In our experience there will be convergence towards a partial solution where
developability in high curvature areas (singularities) is not achieved, see Fig. 13. Even if a smooth solution exists, developable lofting is known to be challenging. An example demonstrating the capabilities of our method is shown by Fig. 14.

![Image](image1.png) ![Image](image2.png)

Fig. 14. An architectural design based on challenging developable lofting tasks. CAD software has been used to connect given boundaries by a NURBS loft, followed by quad remeshing. Subsequently our optimization towards developability has been applied. Observe that patches have several boundaries, and meshes need to have combinatorial singularities.

4 GUIDED SURFACE APPROXIMATION

Our aim is to approximate a general surface by a collection of developable strips. It is important that this decomposition is not performed automatically, but patches are defined by the user. In a second step the individual patches are made developable. § 4.1 describes the procedure in general and identifies cases where we can expect it to work.

4.1 User-Defined Patch Decomposition

We assume that we are given a surface and prescribed boundaries of strips. In a first step we apply any of the well known methods for quad remeshing, e.g. [Ebke et al. 2016]. For explanation purposes we assume we are given a quad mesh and a decomposition of this mesh into strips, where the strip boundaries (“strip curves”) follow the edges of the mesh. We imagine that the width of each individual strip is at least three faces. Fig 1 shows such a mesh and two different decompositions into strips. In a second step, we apply optimization to make the individual strips developable.

Obstacles to developability. Previous methods cannot easily use this method. The reasons are the same as were already listed in § 3.2. It is however important to know that there are some fundamental, geometric, obstacles to developability which hinder the success of optimization.

In the case of lofting two curves, these are sometimes not easy to see. However, in case of lofting curves that sample a surface (see Fig. 15) we can draw on geometric information contained in that surface to infer solvability of the lofting problem: In a developable surface, rulings are conjugate to all other tangent vectors, as exemplified by Figure 2. Since we wish to make strips developable while keeping the strip curves fixed, we can predict future rulings by computing directions that are conjugate to edges. This can be done simply by intersecting discrete tangent planes $\tau_f$ according to Fig. 4. We can therefore immediately visually identify strips where optimization towards developability is likely to fail. This happens in cases where the prospective rulings are nearly tangent to the strip boundary, see Fig. 15.
Fig. 15. *Lofting in Surfaces*. In all three subfigures we select two curve segments $c_1, c_2$ in the same given reference surface (left) and optimize the surface strip defined by them towards developability (right). Gauss images of the strips involved show to which extent we succeed. Success is directly correlated to the line field of prospective ruling directions computed according to Fig. 4. In case (c) this line field runs across the strip, so we can expect a smooth solution. This is indeed confirmed by optimization (right hand image). In cases (a) and (b) the line field is not transverse to the strip, which means we cannot expect that lofting leads to a smooth developable.
4.2 Developable Patches Adapting to a Reference Surface

Figure 16 shows different ways developable patches can approximate, or adapt to, a given reference surface $S$, including approximation in the vicinity of a guiding curve, growing patches from a planar seed, and letting a simple developable like a plane wrap itself around $S$. These tasks can be formulated as

$$ E(x) \rightarrow \min, \quad \text{subject to} \quad C(x) = 0, $$

where $E \rightarrow \min$ expresses proximity to $S$, and the energy $C$ assumes its minimum $C = 0$ on the constraint manifold $M$. $M$ consists precisely of those $x$ which obey constraints.
We recompute \( v^* \) and \( n_{v^*} \) before every round of optimization. Like for Equ. (5), we replace the distance field of \( S \) by distances to tangent planes (regularized by distances to \( v^* \), with a small factor \( \lambda \)).

The energy \( C(x) \) is the sum of squares of real-valued functions \( c_j \), so the constraint manifold \( M \) is equivalently defined by equations \( c_1 = c_2 = \ldots = c_N = 0 \). Linearization of those equations in a point \( x \in M \) yields the tangent space of \( M \):

\[
v \in T_x M \iff \langle \nabla c_1(x), v \rangle = \ldots = \langle \nabla c_N(x), v \rangle = 0.
\]

After these preparations we use the following simple procedure for minimizing \( E \) under the constraint \( C(x) = 0 \):

1. Starting with the initial value \( x \in M \), compute a minimizer \( x + h \) of \( C + \mu E \), which lies outside the constraint manifold.
2. Correct \( h \) by projecting it into the tangent space \( T_x M \). Since \( T_x M^\perp \) is spanned by gradients \( \nabla c_j(x) \), we let \( h' = h + \sum \lambda_j \nabla c_j(x) \) and solve \( \langle h + \sum \lambda_j \nabla c_j(x), \nabla c_i(x) \rangle = 0, i = 1, \ldots, N \). This yields an improved increment \( h' \).
3. Compute a minimizer \( x^* \) of \( C \), starting with initial value \( x + h' \).
4. Let \( x = x^* \) and repeat from (1). We terminate the loop when there is no more improvement.

This procedure was sufficient for our needs and constitutes only a minor modification of our other optimization procedures. We do not claim that it would work in more general circumstances.

We applied the constrained optimization procedure described here to compute the developable patches \( B, C, D \) shown by Fig. 16, with patch \( C \) being initialized according to Fig. 2.

5 DISCUSSION

5.1 Implementation

All interactive design tools described in § 3, as well as the various methods of § 4 were implemented as part of Rhinoceros3D. Our plugin is a C++ dynamic-link Windows® library that can directly interact with the Rhino application. The benefit of our implementation choice is two-fold: first, it seamlessly combines with all the existing geometry.
Developable Quad Meshes

processing tools already in Rhino. This results in a powerful design environment. Secondly, Rhino is widely used within
the architectural community, and thus provides a natural, broad user base for the new algorithms. Our plugin will be
made available as open source software.

The optimization of §2.3 and §4.2 was solved using a Levenberg-Marquardt method according to [Madsen et al. 2004,
§3.2], with a damping parameter of $10^{-6}$. In the interactive application, the optimization is restarted on every user
input. We terminate the optimization loop when the energy value falls below a certain threshold, or when there is no
more improvement.

Our implementation uses Rhino’s openNURBS toolkit for elementary geometric manipulations, and the Intel®
oneAPI Math Kernel library for efficient sparse matrix manipulations, and for solving large sparse symmetric linear
systems.

The behaviour of energies during the course of optimization is exemplarily shown by Figure 17. Table 1 provides
statistics on the size of optimization problems, the choice of weights, and the time needed. The latter are measured on an
Intel® Xeon® CPU E5-2695 v3 @ 2.30GHz x64-based processor, running 64-bit Windows® 10. Our plugin is configured
to use 64 parallel openMP threads. This was chosen to achieve interactivity for the models shown throughout the
figures, but could be tuned for larger designs.

5.2 Validation and Comparison

Our developability condition (4) imposed on meshes is comparable to the requirement that the Gauss curvature of a
smooth developable vanishes. The latter has a list of local and global implications, including a curve-like Gauss image
and existence of rulings. These properties could be verified up to tolerances.

Visualization of Gauss curvature. The property of the Gauss image being curve-like is a very sensitive indicator of
developability. In contrast to this, visualizing the numerical values of Gauss curvature is not suitable to properly identify
developable surfaces, because the numerical errors inherent in the approximate computation of Gauss curvature is
bigger than the value of Gauss curvature itself. This was confirmed in experiments on fair quad meshes sampled from
mathematically correct developables, using the jet fit method of [Cazals and Pouget 2003] as provided by libigl [Jacobson
et al. 2018]. For this reason we validate developablity via the Gauss image throughout this paper.

Fig. 18. The reference surface (Fig. 1, left) is approximated by a piecewise-developable surface using the method of [Sellán et al. 2020].
The parameter of the method has been set to (a) $\lambda = 10^4$; (b) $10^5$; (c) $10^6$. One can see that this automatic method tends to flat pieces
and individual small single-curved pieces that become smaller as the resolution increases. For this reason the result on the right
appears as if it was double-curved.
Comparisons. The focus of our paper is on design rather than on automatic approximation, which is mainly due to the fact that we found automatic methods not sufficient for the applications we have in mind. Our aim is to give the designer sufficient control over the way a given double-curved surface is represented by developable pieces. We were not able to recreate decompositions as shown by Fig. 1 with other methods. Figure 18 shows selected results of experiments with the method by Sellan et al. [2020] which applies to heightfields. Figure 19 shows a few of our many experiments with the method of Binninger et al. [2021]. Both methods have no provision for a decomposition into strips; a decomposition into the same number of patches as we did in Fig. 1 obviously does not exhibit good approximation quality. There is another feature distinguishing these methods from ours, namely the sharpness of creases between developable pieces. The fact that they are not sharp is clearly visible in the Gauss images of Figures 18 and 19; a sharp crease would have to be achieved in a post-processing step.

For comparisons with [Ion et al. 2020] and [Stein et al. 2018] we refer to [Sellán et al. 2020] and [Binninger et al. 2021].

5.3 Conclusion
The developability criterion for quad meshes presented in this paper has successfully been used to solve design problems with developable surfaces, which is a well known and difficult topic with a long list of individual contributions. The fact that the edges of our developable quad meshes do not have to be aligned with special curves, represents a great practical advantage. Another advantage is the fact that we do not have to design the actual development at the same time as the developable surface.
When approximating a given surface by a piecewise-developable one we emphasized the viewpoint that the decomposition into individual developable pieces would be performed by an algorithm but should be guided by the designer (Fig. 1, Fig. 20).

Fig. 21. Creases as a design element of piecewise-developable surfaces.

Of course, user-defined patches are not necessarily capable of being optimized to become developable, at least not without violating tolerances. Our way out of this dilemma is to inform the user of this fact and introduce creases as a design element (Fig. 21). So are singularities, which we achieve with a trick, using lofting (Fig. 22).

Fig. 22. Lofting Singularities. By choosing boundaries and optimizing the surfaces between them we achieve piecewise developability. The singularity at the top is actually a strip boundary that doubles back onto itself.

Interactive Modeling. Our method is interactive in two ways. Firstly it can be used to interactively model developables (see Fig. 10 and the accompanying video). Secondly, we provide immediate feedback to the user: The line field $r_e$ of prospective rulings informs on the quality of a developable (see Fig. 15, right hand column), in part even before optimization is performed (see Fig. 15, left hand column). The Gauss image directly shows if developability has been achieved. The user can react and change constraints, or the weights given to constraints.

The developability condition proposed in this paper is almost-invariant under remeshing, see Fig. 23. This means representing a given developable mesh by another quad mesh with fair mesh poly-lines yields a mesh that almost fulfills our criterion (4) again (this is because the underlying geometric property has not been changed). This property has
been used to create the examples of Figures 20c and 24, and is extremely useful e.g. for trimming and for joining patches. We emphasize that we can work with most methods for user-guided quad remeshing, e.g. [Ebke et al. 2016].

We emphasize that we can work with most methods for user-guided quad remeshing, e.g. [Ebke et al. 2016].

For the same reason, (4) is almost-invariant under projective transformations, see Fig. 23. Affine transformations were used in the modelling process that led to Figure 24.

Limitations. One major limitation of our methods is geometric in nature. While an experienced user can generate developables easily, this is more difficult for a user without prior knowledge of the quirks of developable surfaces, even if we support the user’s design choices e.g. by showing prospective ruling vector fields. Our academic implementation currently requires the user to choose weights meaningfully.

Future Research. Our aim is to publish a plugin for the software Rhino, which would benefit from a conversion to NURBS format and which requires a higher level of system intelligence than is present at the moment. For actual applications, material properties and tolerances need to be considered (which works in our favour).

Fig. 23. Stability of the developability measure $E_{\text{dev}}$. Remeshing mesh (a) yields meshes (b), (c), and a projective transformation yields (d). $E_{\text{dev}}$ remains small if these transformations are applied.

For the same reason, (4) is almost-invariant under projective transformations, see Fig. 23. Affine transformations were used in the modelling process that led to Figure 24.

Limitations. One major limitation of our methods is geometric in nature. While an experienced user can generate developables easily, this is more difficult for a user without prior knowledge of the quirks of developable surfaces, even if we support the user’s design choices e.g. by showing prospective ruling vector fields. Our academic implementation currently requires the user to choose weights meaningfully.

Future Research. Our aim is to publish a plugin for the software Rhino, which would benefit from a conversion to NURBS format and which requires a higher level of system intelligence than is present at the moment. For actual applications, material properties and tolerances need to be considered (which works in our favour).

For the same reason, (4) is almost-invariant under projective transformations, see Fig. 23. Affine transformations were used in the modelling process that led to Figure 24.

Limitations. One major limitation of our methods is geometric in nature. While an experienced user can generate developables easily, this is more difficult for a user without prior knowledge of the quirks of developable surfaces, even if we support the user’s design choices e.g. by showing prospective ruling vector fields. Our academic implementation currently requires the user to choose weights meaningfully.

Future Research. Our aim is to publish a plugin for the software Rhino, which would benefit from a conversion to NURBS format and which requires a higher level of system intelligence than is present at the moment. For actual applications, material properties and tolerances need to be considered (which works in our favour).
REFERENCES

Alexandre Binninger, Floor Verhoeven, Philipp Herholz, and Olga Sorkine-Hornung. 2021. Developable Approximation via Gauss Image Thinning. *Comput. Graph. Forum* 40, 5 (2021), 289–300. Proc. SGP.

R.M.C. Bodduluri and Bahram Ravani. 1993. Design of developable surfaces using duality between plane and point geometries. *Computer-Aided Design* 25 (1993), 621–632.

Frédéric Caazals and Marc Pouget. 2003. Estimating differential quantities using polynomial fitting of osculating jets. In *Proc. Symp. Geometry Processing*. 177–178.

Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2018. Shape from Metric. *ACM Trans. Graph.* 37, 4 (2018), 63:1–17.

Hans-Christian Ebke, Patrick Schwädt, Marcel Campen, and Leif Kobbelt. 2016. Interactively Controlled Quad Remeshing of High Resolution 3D Models. *ACM Trans. Graph.* 35, 6 (2016), 218:1–13.

Heinrich Guggenheimer. 1963. *Differential Geometry*. McGraw-Hill.

Alexandra Ion, Michael Rabinovich, Philipp Herholz, and Olga Sorkine-Hornung. 2020. Shape Approximation by Developable Wrapping. *ACM Trans. Graph.* 39, 6 (2020), 200:1–12.

Alec Jacobson, Daniele Panozzo, et al. 2018. *libigl*: A simple C++ geometry processing library. https://libigl.github.io

Caigui Jiang, Klara Mundilova, Florian Rist, Johannes Wallner, and Helmut Pottmann. 2019. Curve-pleated structures. *ACM Trans. Graph.* 38, 6 (2019), 169:1–13.

Caigui Jiang, Chengcheng Tang, Marko Tomičić, Johannes Wallner, and Helmut Pottmann. 2015. Interactive Modeling of Architectural Freeform Structures – Combining Geometry with Fabrication and Statics. In *Advances in Architectural Geometry 2014*, P. Block et al. (Eds.). Springer, 95–108.

Caigui Jiang, Cheng Wang, Florian Rist, Johannes Wallner, and Helmut Pottmann. 2020. Quad-Mesh Based Isometric Mappings and Developable Surfaces. *ACM Trans. Graph.* 39, 4 (2020), 128:1–13.

Johann Lang and Otto Röschel. 1992. Developable $(1, n)$-Bézier Surfaces. *Comput. Aided Geom. Design* 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.

Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. 2004. *Methods for non-linear least squares problems* (2nd ed.). Technical Univ. Denmark.

Chi-Han Peng, Caigui Jiang, Peter Wonka, and Helmut Pottmann. 2019. Checkerboard Patterns with Black Rectangles. *ACM Trans. Graph.* 38, 6 (2019), 228:1–17.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018a. Discrete Geodesic Nets for Modeling Developable Surfaces. *ACM TOG* 37, 2 (2018), 16:1–17.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2018b. The Shape Space of Discrete Orthogonal Geodesic Nets. *ACM TOG* 37, 6 (2018), 228:1–17.

Michael Rabinovich, Tim Hoffmann, and Olga Sorkine-Hornung. 2019. Modeling Curved Folding with Freeform Deformations. *ACM TOG* 38, 6 (2019), 170:1–12.

Robert Sauer. 1970. *Differenzgeometrie*. Springer.

Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2020. Developability of Heightfields via Rank Minimization. *ACM Trans. Graph.* 39, 4 (2020), 109:1–15.

Madalina Signer, Alexandra Ion, and Olga Sorkine-Hornung. 2021. Developable Metamaterials: Mass-Fabricable Metamaterials by Laser-Cutting Elastic Structures. In *Proc. CHI Conf. Human Factors in Computing Systems*. ACM, 674:1–13.

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

Oded Stein, Eitan Grinspun, and Keenan Crane. 2018. Developability of Triangle Meshes. *ACM Trans. Graph.* 37, 4 (2018), 77:1–14.

Chengcheng Tang, Pengbo Bo, Johannes Wallner, and Helmut Pottmann. 2016. Interactive design of developable surfaces. *ACM Trans. Graph.* 35, 2 (2016), 12:1–12.

Floor Verhoeven, Amir Vaxman, Tim Hoffmann, and Olga Sorkine-Hornung. 2022. Dev2PQ: Planar Quadrilateral Strip Remeshing of Developable Surfaces. *ACM Trans. Graphics* 41, 3 (2022), 29:1–18.

Thomas Wolf, Victor Cornillère, and Olga Sorkine-Hornung. 2021. Physically-based Book Simulation with Freeform Developable Surfaces. *Comput. Graph. Forum* 40, 2 (2021), 449–460.