Quad-mesh based isometric mappings and developable surfaces

CAIGUI JIANG, KAUST
CHENG WANG, KAUST
FLORIAN RIST, KAUST / TU Wien
JOHANNES WALLNER, TU Graz
HELMUT POTTMANN, KAUST

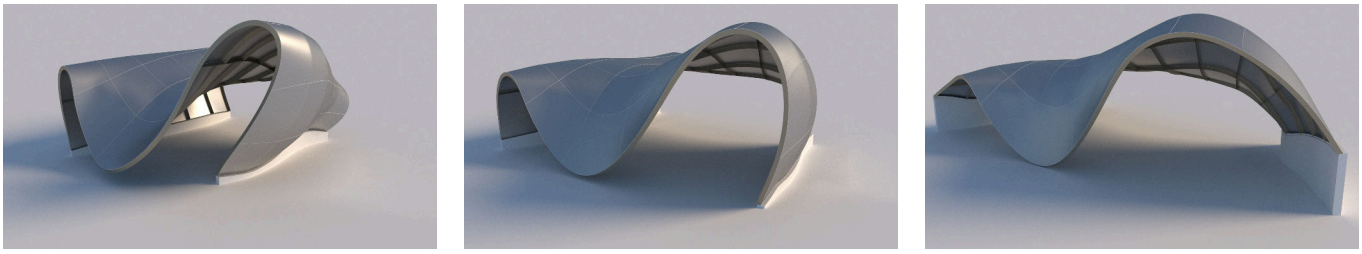

Fig. 1. *Design with developables in freeform architecture.* We propose a simple approach to isometric mappings between quad meshes, which immediately leads to a way to model developable surfaces. For this example we started from a segmentation into panels of part of a freeform concrete shell by Zaha Hadid Architects, built in 2014 in Baku. We approximated it by a quad mesh which is piecewise discrete-developable in the sense of our framework. Afterwards this mesh undergoes an isometric deformation which respects the panelization. Our approach to discrete-developable surfaces is very flexible in the sense that the edges of meshes do not have to be aligned with rulings or otherwise special curves on developables, and can be aligned with boundaries and features instead.

We discretize isometric mappings between surfaces as correspondences between checkerboard patterns derived from quad meshes. This method captures the degrees of freedom inherent in smooth isometries and enables a natural definition of discrete developable surfaces. This definition, which is remarkably simple, leads to a class of discrete developables which is much more flexible in applications than previous concepts of discrete developables. In this paper, we employ optimization to efficiently compute isometric mappings, conformal mappings and isometric bending of surfaces. We perform geometric modeling of developables, including cutting, gluing and folding. The discrete mappings presented here have applications in both theory and practice: We propose a theory of curvatures derived from a discrete Gauss map as well as a construction of watertight CAD models consisting of developable spline surfaces.

CCS Concepts: • **Computing methodologies** → **Shape modeling**; *Optimization algorithms.*

Additional Key Words and Phrases: discrete differential geometry, computer-aided design, computational fabrication, shape optimization, discrete isometry, developable surface, developable spline surface

Authors' addresses: Caigui Jiang, KAUST; Cheng Wang, KAUST; Florian Rist, KAUST / TU Wien; Johannes Wallner, TU Graz; Helmut Pottmann, KAUST.

0730-0301/2020/7-ART128 $15.00
https://doi.org/10.1145/3386569.3392430

ACM Reference Format:
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, Article 128 (July 2020), 13 pages. https://doi.org/10.1145/3386569.3392430

1 INTRODUCTION

The geometric modeling of developable surfaces is a topic attracting attention since many years. One reason for that is the great practical importance of developables, which represent shapes made by bending flat pieces of inextensible sheet material into space. Materials which fall into this category include paper and sufficiently thin plates [Audoly and Pomeau 2009]. New algorithms and computational representations of developables continue to emerge. This steady progress is a sign that the problem of modeling developables still has no complete and satisfactory solution. The mathematical theory of developables is far from simple, which probably accounts for a part of the computational difficulties which occur. The approach to developables presented in this paper is via a systematic theory of isometric mappings. It is based on correspondences between quad meshes, with no specific restrictions on the meshes themselves.

1.1 Overview and Contributions

We propose to manage discrete surfaces and their mappings not directly, via properties of the vertices, edges and faces, but via properties of a checkerboard pattern inscribed in the original mesh. That pattern is created by inserting midpoints of edges – the edge midpoints belonging to a face will always form a parallelogram, regardless of the shape of the original face.

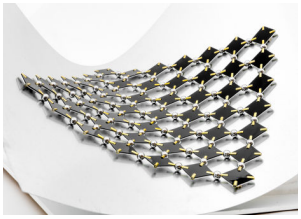

Fig. 2. A checkerboard pattern. The black parallelograms are inscribed in the original faces. Physical models such as this can be moved around on developable reference surfaces, which is evidence for our claim that checkerboard patterns are a good discretization.

Two combinatorially equivalent quad meshes, and their derived checkerboard patterns are now used to discretize the concept of mapping between surfaces (Sec. 2). First-order properties of mappings are defined in terms of the checkerboard pattern, e.g. for an isometric mapping we require that corresponding parallelograms are congruent.

The setup we use for mappings between surfaces can also be employed to define a discrete version of the Gauss mapping, from which curvatures are derived (Sec. 2.2).

The constraints we use to express the isometric property leave just the right number of degrees of freedom that we expect from the smooth case. This is why we are able to use our discrete-isometric mappings directly for geometric modeling. In particular we are able to define discrete developables as surfaces isometric to a quadrangulation of a planar domain (Sec. 3).

We demonstrate the capabilities of our discrete-developable surfaces by means of geometric modeling tasks, the most important one being watertight developable CAD models. We approach this hitherto unsolved problem by combining our discrete-isometric mappings with subdivision (Sec. 4) .

Section 5 contains more results, including approximating a given surface with a developable one, and how to increase the developability of a given surface. We also show how to compute developables defined by gluing boundaries, and by folding along curves.

1.2 Previous Work

There is a large amount of literature on developable surfaces, and even more about the topic of mappings between surfaces, which includes mesh parametrization. Related work in discrete and numerical differential geometry is discussed later, in §2.2.2.

Previous work on developable surfaces. Computational treatments of developables can be divided into two groups. One kind of methods is based on the fact that developables are, essentially, one-dimensional objects. Each developable consists of pieces of ruled surfaces with enjoy the property that the tangent plane along a ruling is constant. In geometric modeling, a big advantage can be gained by reducing the dimensionality of the problem from two to one. Rose et al. [2007] use this principle in solving the problem of finding all developables defined by a given boundary. The dual approach by [Bodduluri and Ravani 1993] and follow-up publications like [Pottmann and Wallner 1999] treats developables as curves in dual space. Ruled surfaces as degree $(1, n)$ B-spline surfaces, with developability imposed as an extra nonlinear side condition have been proposed early, and form the basis of the work by [Jiang et al. 2019; Tang et al. 2016] (for a history of this approach we refer to the extensive bibliographies contained in these papers). In the discrete realm, a ruled surface can be represented in a sequence of quadrilaterals, with developability expressed by planarity of quads. This principle proposed by [Sauer 1970] is the basis of [Liu et al. 2006] and [Solomon et al. 2012].

A disadvantage of these methods is that the decomposition of a developable into its ruled pieces can be arbitrarily complicated combinatorially [do Carmo 1976, p. 195]. The rulings are no stable function of the location on a developable surface: This dependence may be non-smooth [Hartman and Nirenberg 1959, p. 916f], and the ruling pattern can change in unpredictable ways during deformation – e.g. when passing through a flat state, the ruling patterns before and after the flat position are unrelated. With ruling based-methods it is more difficult to model situations where the ruling pattern changes much, and such methods are naturally biased towards solutions where big changes do not occur.

Alternative computational models are discrete two-dimensional surfaces characterizing developability in a local manner. This can be done e.g. by requiring a triangle mesh to be intrinsically flat, having angle sum 2π in each vertex. It is however not straightforward to model, in addition to developability, also the bending behaviour of developables, since the developability constraints can easily be stronger than we expect from the smooth situation. Recent examples of discrete developables are the orthogonal-geodesic nets proposed by Rabinovich et al. [2018a; 2018b; 2019], and the developable triangle meshes of Stein et al. [2018] where a local 'hinge' condition ensures existence of rulings.

Approximation with developables has been done on various levels, starting with [Mitani and Suzuki 2004]. Stein et al. [2018] use their definition of discrete developability to modify surfaces such that the Gauss curvature becomes concentrated on curves, achieving a piecewise-developable approximation. A different approach exploits the essentially one-dimensional nature of developables. While the Gauss image (normal vector image) of an arbitrary surface covers an area of the unit sphere, the Gauss image of a developable or piecewise-developable surface has zero area [do Carmo 1976, p. 167] and thus, assuming smoothness, is a curve on the unit sphere. Approximating an arbitrary surface by a few developable patches is therefore at least as difficult as approximating a 2D domain by a few 1-dimensional arcs. Recently, Gavriil et al. [2018] employed this viewpoint to realize architectural freeform shapes by individual developable panels.

Previous work on checkerboard patterns. This paper has been motivated by recent work on discrete surfaces exhibiting two classes of 'white' and 'black' faces arranged in a checkerboard pattern. Peng et al. [2019] discuss patterns with 'black rectangles', their degrees of freedom and their computation. Our paper operates with checkerboard patterns with 'black parallelograms' generated by subdivision from an arbitrary quad mesh. A special case, namely patterns with 'black squares' turns out to be related to [Rabinovich et al. 2018a].

Previous work on mappings between surfaces. Finding correspondences and mappings between geometric shapes is one of the most significant and diverse problems in geometry processing, cf. e.g. the recent contributions [Chern et al. 2018; Ezuz et al. 2019]. In our own work we focus on *isometric* mappings. They have been studied in detail, often from the viewpoint that true isometries cannot

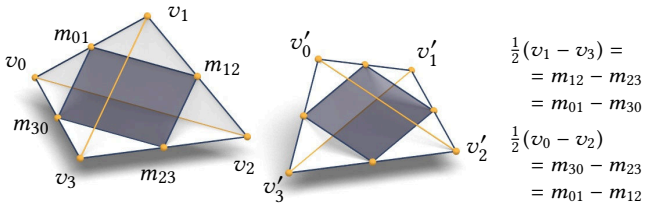

Fig. 3. For any face with vertices v_j, the midpoints $m_{j,j+1}$ of edges obey the equations above, implying that those midpoints form a parallelogram. A correspondence between faces $v_0v_1v_2v_3$ and $v'_0v'_1v'_2v'_3$ induces an affine mapping between inscribed midpoint parallelograms. The correspondence is conformal resp. isometric, if that affine map is a similarity resp. a congruence.

be achieved. As a consequence, as-rigid-as-possible mappings resp. near-isometric mappings have been investigated, see e.g. [Claici et al. 2017; Liu et al. 2009; Pietroni et al. 2010; Sorkine and Alexa 2007]. Our idea to model isometry via keeping the shape of small elements is not entirely new, see e.g. constrained shape exploration as set up by Bouaziz et al. [2012]. Peng et al. [2019] already propose to define isometric and conformal mappings by local isometries and local similarities of checkerboard patterns, but considered restricted shapes of patterns only.

Relevant work on isometric mappings has been done by Chern et al. [2018]. Based on a discrete theory of spin structures, they treat isometric mappings, particularly immersions. Recently, Sassen et al. [2020] investigated the flexions of triangle meshes with rigid faces via the manifold of realizations of fixed edge lengths but variable dihedral angles. They formulate the corresponding integrability conditions and solve both exact and approximate reconstruction of meshes, which includes computing isometric mappings. The main points in which our work differs from these two recent contributions is the different focus (ours is geometric modeling, cutting and gluing, applications in paneling) and the simplicity of our approach.

Conformal mappings are not a focus of the present paper, and we only refer to the bibliography in [Peng et al. 2019].

2 MAPS BETWEEN SURFACES AS CORRESPONDENCES BETWEEN CHECKERBOARD PATTERNS

In the same way a mesh M is a discrete surface, a pair M, M' of combinatorially equivalent meshes can be seen as a discrete version of a *mapping* between surfaces. This viewpoint has been highly successful e.g. in the study of discrete conformal mappings as correspondences between special meshes (namely, circle patterns, see [Stephenson 2005]). In the same tradition, Peng et al. [2019] suggest the following setup. A control mesh $M = (V, E, F)$ is associated with a *checkerboard pattern* (V_P, E_P, F_P) derived from M.

- Every edge $vw \in E$ defines a vertex $m_{vw} = \frac{1}{2}(v + w)$, which is the midpoint of that edge, see Fig. 3.
- Every vertex $v \in V$ defines a new 'white' face with vertices m_{vw} where vw is an edge incident with v.
- Every face $f \in F$ defines a new 'black' face with vertices m_{vw}, where vw is an edge contained in f.
- Vertex–face inclusion translates to adjacency of white and black faces in the checkerboard pattern.

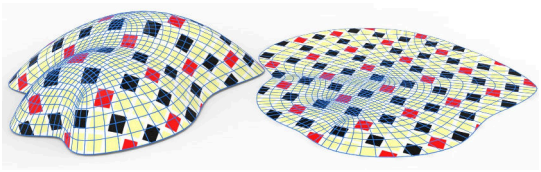

Fig. 4. A conformal mapping between quad meshes M and M'. Here M' is flat, which makes the mapping from M' to M a conformal parametrization. In fact this parametrization is as isometric as possible.

Varignon's theorem. For any quadrilateral face of M, the edges of the inscribed 'black' face obey the equations shown by Fig. 3. They say that these edges are parallel to the diagonals of the original face, implying the inscribed face is a parallelogram. This fact, holds even if the original face is not planar, and is sometimes called *Varignon's theorem.*

2.1 Conformal maps and isometric maps

Peng et al. [2019] propose to discretize a mapping between surfaces as the correspondence between combinatorially equivalent meshes M and M'. They consider only such meshes where induced checkerboard patterns exhibit black rectangles, and they propose that the mapping is isometric (resp., conformal), if corresponding black faces are related by a congruence (resp. similarity) transformation. We are going to demonstrate that the very same definition is sensible also in the general case, without any restriction to the shape of faces.

Consider corresponding faces $f = v_0v_1v_2v_3$ and $f' = v'_0v'_1v'_2v'_3$ of the given meshes M and M', see Figure 3. The inscribed parallelograms are congruent, if and only if their edge lengths and angles coincide, i.e., if and only if

$$\begin{aligned} c_{iso,0}(f) &= \|v_0 - v_2\|^2 - \|v'_0 - v'_2\|^2 = 0, \\ c_{iso,1}(f) &= \|v_1 - v_3\|^2 - \|v'_1 - v'_3\|^2 = 0, \\ c_{iso,2}(f) &= \langle v_0 - v_2, v_1 - v_3\rangle - \langle v'_0 - v'_2, v'_1 - v'_3\rangle = 0. \end{aligned} \quad (1)$$

In an analogous way we state conditions expressing a similarity transformation: The equations

$$\begin{aligned} c_{conf,0}(f) &= \lambda_f\|v_0 - v_2\|^2 - \|v'_0 - v'_2\|^2 = 0, \\ c_{conf,1}(f) &= \lambda_f\|v_1 - v_3\|^2 - \|v'_1 - v'_3\|^2 = 0, \\ c_{conf,2}(f) &= \lambda_f\langle v_0 - v_2, v_1 - v_3\rangle - \langle v'_0 - v'_2, v'_1 - v'_3\rangle = 0 \end{aligned} \quad (2)$$

say that edge lengths of the inscribed parallelogram are multiplied with the factor $(\lambda_f)^{1/2}$, but angles stay the same. Both (1) and (2) directly refer to the control mesh M.

2.2 Curvatures for quad meshes

Checkerboard patterns as introduced above are well suited to introduce curvatures, because the 'black' faces are planar see (Fig. 3) and have well-defined normal vectors. The classical definition of curvatures via the Gauss map is nicely compatible with our definition of discrete mappings between discrete surfaces. We here describe our construction and its relation to other definitions of curvatures.

2.2.1 Definition of curvatures per face. The main idea is to define a normal vector n_i for each vertex v_i of the control mesh M, and consider offset meshes M^δ with vertices $v_i + \delta n_i$. We require that

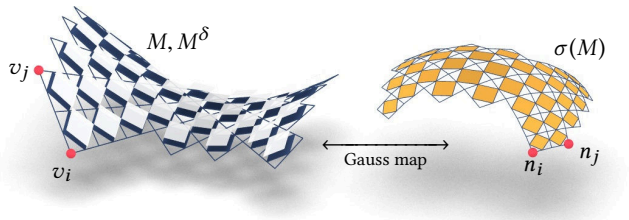

Fig. 5. A mesh M (left) with vertices v_i, normal vectors n_i (Gauss image $\sigma(M)$, at right) and an offset M^δ with vertices $v_i + \delta n_i$. The distance of corresponding 'black' parallelograms inscribed to M, M^δ is δ.

in the checkerboard patterns derived from meshes M, M^δ, corresponding 'black' parallelograms lie in parallel planes at distance δ from each other. Figure 5 illustrates this situation. By linearity, it is sufficient to require this property only for $\delta = 1$.

For the face $f = v_0v_1v_2v_3$ with normal vectors $n_0, \dots, n_3$, our distance requirement translates to the four conditions

$$c_{norm,j}(f) = \langle n_j + n_{j+1}, n_f \rangle - 2 = 0, \quad (j = 0, \dots, 3) \tag{3}$$

where n_f is a unit normal vector of the inscribed parallelogram, consistently pointing to one side of the mesh. Indices are taken modulo 4. Actually $c_{norm,3}(f)$ is redundant (because if 3 vertices of a parallelogram lie in a certain plane, also the 4th vertex does).

Note that the normal vectors n_i are *not* unit vectors, and they are also not uniquely defined by (3) alone. In practice, they are computed via optimization, see Sec. 2.3.1. They constitute the vertices of the Gauss image $\sigma(M)$. By construction, the checkerboard pattern derived from $\sigma(M)$ has 'black' parallelograms tangentially circumscribed to the unit sphere. The Gauss map within each parallelogram defines a shape operator, mapping edges of a parallelogram inscribed to M to the corresponding edges of a parallelogram inscribed in $\sigma(M)$. The usual sign convention (the shape operator is the negative derivative of the Gauss map) implies that the discrete shape operator in the face f reads

$$s_f : \begin{cases} v_2 - v_0 \longmapsto -(n_2 - n_0), \\ v_3 - v_1 \longmapsto -(n_3 - n_1). \end{cases}$$

Both parallelograms inscribed in $f = v_0v_1v_2v_3$ and in $\sigma(f) = n_0n_1n_2n_3$ have n_f as a normal vector. It therefore makes sense to view s_f as a mapping of the two-dimensional subspace ${n_f}^\perp$ into itself. Then both Gauss curvature K and mean curvature H (in the pointwise sense) are naturally defined via determinant and trace of s_f, just like in the smooth theory:

$$K(f) = \det(s_f) = \frac{[n_3 - n_1, n_2 - n_0]}{[v_3 - v_1, v_2 - v_0]}, \tag{4}$$
$$H(f) = \frac{tr(s_f)}{2} = -\frac{[n_3 - n_1, v_2 - v_0] + [v_3 - v_1, n_2 - n_0]}{2[v_3 - v_1, v_2 - v_0]}.$$

Here we used the notation $[a, b] = \det(a, b, n_f)$ for the determinant of vectors in the subspace ${n_f}^\perp$. Further, it is natural to consider the shape operator's eigenvalues $\kappa_1(f), \kappa_2(f)$ as the principal curvatures, with eigenvectors indicating principal directions. Eigenvalues are computed e.g. as the roots of the equation $x^2 - 2Hx + K = 0$.

The principal directions must be orthogonal to each other, which happens if and only if the shape operator is symmetric, i.e,

$$c_{sym}(f) = \langle n_3 - n_1, v_2 - v_0 \rangle - \langle v_3 - v_1, n_2 - n_0 \rangle = 0. \tag{5}$$

We add this condition as a constraint in our computations. It acts as a regularizer and contributes to eliminating the ambiguity which is still left after conditions (3) are imposed. The results are numerically convincing, see Fig. 6.

2.2.2 Comparison with other quad mesh-based curvature theories. Curvatures of polyhedral surfaces were introduced by [Bobenko et al. 2010; Pottmann et al. 2007]. They consider a polyhedral mesh M endowed with polyhedral offsets M^δ at distance δ such that corresponding faces f, f^δ of M, M^δ lie in parallel planes. Mean curvature $H(f)$ and Gauss curvature $K(f)$ of the face f are derived from Steiner's formula,

$$\text{area}(f^\delta) = \text{area}(f)(1 - 2\delta H(f) + \delta^2 K(f)). \tag{6}$$

The very same relation applies to our curvatures and the areas of 'black' parallelograms. This can be easily seen from (4), because the area of the parallelogram inscribed in the face $v_0v_1v_2v_3$ is exactly $\frac{1}{4}[v_2 - v_0, v_3 - v_1]$. The major difference between our setup and that of Bobenko et al. [2010] is that in our case, the relation (6) applies to only one half of the faces. If it applies to all faces, it represents a rather strong condition on the meshes involved — for quad meshes this essentially requires that edges follow principal curvature lines.

A further development is the theory of curvatures for *edge-constraint nets* by Hoffmann et al. [2017]. They operate with unit normal vectors of vertices. From the computational viewpoint, their constraints on normal vectors are more involved than ours. While the work of Hoffmann et al. [2017] is capable of unifying several previous constructions, a thorough theoretical investigation of our definition of curvatures is a topic of future research.

2.3 Computation of mappings via optimization

In geometric modeling with mappings, a basic task is to make them isometric or conformal, changing them as little as possible (this includes the task of making surfaces developable, changing them as little as possible in the process). Obviously it is important that this can be done quickly. Since the conditions we impose on mappings are not sufficient to guarantee smoothness, we perform all computational tasks via optimization and employ suitable regularizers.

2.3.1 Setup of variables and mapping-related constraints. The computations in this paper involve two quad meshes $M = (V, E, F)$ and

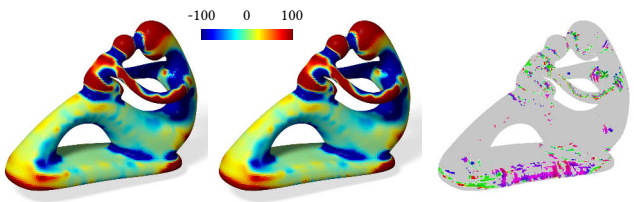

Fig. 6. Comparing the value of Gauss curvature computed with our method (left) with the *jet fit* method of [Cazals and Pouget 2003] (center). The differences, visualized via color difference (right) are barely visible.

$M' = (V', E', F')$ which are combinatorially equivalent. The natural correspondence between vertices encodes a mapping. The variables of our computation, including the coordinates of vertices, are stored in a vector $X \in \mathbb{R}^N$. Constraints originally formulated as equations are expressed via minimization of energies. The isometry constraint of Equ. (1) is expressed by $w_{iso}E_{iso}(X) \to \min$, where

$$E_{iso} = \sum\nolimits_{f\in F} \sum\nolimits_{j=1}^{3} c_{iso,j}(f)^2,$$

and w_{iso} is a positive weight. Similarly, to compute a conformal mapping between meshes M, M' we convert Equ. (2) into the optimization problem $w_{conf}E_{conf}(X) + w_\lambda E_\lambda(X) \to \min$, where

$$E_{conf} = \sum\nolimits_{f\in F} \sum\nolimits_{j=1}^{3} c_{conf,j}(f)^2, \quad E_\lambda = \sum\nolimits_{f\in F} (\lambda_f - 1)^2.$$

The energy E_λ penalizes deviation of conformal factors from 1. It is used for regularization and (if w_λ is big enough) to make a conformal map as isometric as possible.

In handle-based editing there are vertices v_i which have to be close to fixed positions a_i and pairs of vertices v_i and v_j which are forced to lie on top of each other (this is necessary for gluing). Similar conditions can be imposed on the mesh M'. These requirements are handled by energies of the form

$$E_{pos} = \sum\nolimits_{i\in I} \|v_i - a_i\|^2 + \sum\nolimits_{(i,j)\in J} \|v_i - v_j\|^2 + (\ldots),$$

where $(\ldots)$ stands for analogous contributions from the mesh M'. A suitable multiple $w_{pos}E_{pos}$ is to be combined with the other energies, as required by the task at hand.

A special case of mappings discussed here is where M' is the Gauss image of M. We express conditions (3) and (5) in the form $w_{norm}E_{norm} + w_{sym}E_{sym} \to \min$, with

$$E_{norm} = \sum\nolimits_{f\in F} \sum\nolimits_{j=0}^{3} c_{norm,j}(f)^2, \; E_{sym} = \sum\nolimits_{f\in F} c_{sym}(f)^2.$$

2.3.2 Soft constraints and approximation. The magnitude of weights employed for the different energies can be used to inform the optimization algorithm of the importance of individual constraints. We speak of soft constraints if weights are small. Typical examples are constraints where we already know they cannot be fulfilled, like the requirement $\lambda_f = 1$ in conformal mappings which was mentioned above. Another instance of soft constraints are the ones used to express proximity of a mesh M to a reference shape Φ.

Such a condition is implemented by computing the closest point projection $v_i^* \in \Phi$ of a representative sample $\{v_i\}_{i\in I}$ of vertices. We subsequently represent the tangent plane of Φ in that point by a linear equation $\tau_i(x) = 0$. Then proximity is expressed by $w_{prox,1}E_{prox,1} + w_{prox,2}E_{prox,2} \to \min$, with

$$E_{prox,1} = \sum\nolimits_{i\in I} \|v_i - v_i^*\|^2, \quad E_{prox,2} = \sum\nolimits_{i\in I} \tau_i(v_i)^2.$$

The footpoints v_i^* and tangent planes are recomputed in each round of our iterative optimization procedure. Using only $E_{prox,1}$ amounts to a version of the well known ICP algorithm and is suitable only if the distance from Φ is still big. Close to Φ, the shape of Φ is much better represented by its tangent plane than by a single point, and $w_{prox,1} \to 0$ greatly speeds up convergence. This matter is discussed in detail by [Wang et al. 2006].

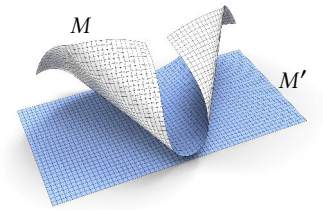

Fig. 7. An isometric mapping between meshes M and M'. Since M' is contained in a plane, M is a discrete developable.

2.3.3 Regularizing meshes and mappings. The discrete-isometric relation between meshes does not prevent zigzagging of mesh polylines, and regularization is necessary to achieve a fair solution. A natural fairness functional is found by penalizing irregular spacing of vertices on mesh polylines. In our optimization, we add the term $w_{fair,M}E_{fair,M}$ to the other energies already in use, with

$$E_{fair,M} = \sum\nolimits_{\substack{v_iv_jv_k \text{ successive} \\ \text{vertices in } M}} \|v_i - 2v_j + v_k\|^2.$$

Similarly we consider $w_{fair,M'}E_{fair,M'}$. These regularizers can be employed whenever the mesh they refer to is not fixed anyway during optimization. A prototypical algorithm where hard and soft constraints are combined is shown below. The weight of the regularizing fairness term is small to begin with, but we are making provision to lower it further in case the fairness term is too dominant and prevents achieving isometry. For different examples in this paper the energy used for optimization is different, and it might not be M' that is the variable but M or even both M and M'. In those cases Algorithm 1 has to undergo obvious modifications.

Algorithm 1: Compute a mesh M' isometric to M

```
Data: Mesh M, initial value for M', side conditions encoded in E_pos
Fix energy thresholds E_min, E_iso,min;
Initialize weights by letting w_iso = 1, w_pos = 1, w_fair,M' = 0.1;
repeat
    repeat
        Subject M' to optimization by minimizing
        E := w_iso E_iso + w_fair,M' E_fair,M' + w_pos E_pos;
    until E ≤ E_min or number of iterations exceeds maximum;
    w_fair,M' ← w_fair,M' / 10;
until E_iso ≤ E_iso,min;
return M'
```

Regularizing mappings between meshes. We want to be able to deal with meshes M, M' that are not particularly fair in terms of these energies, even if the natural correspondence between M and M' approximates a smooth mapping between surfaces. If, say, M is fixed during optimization, then regularizing M' will have a detrimental effect on the quality of the mapping. In that case it is better to regularize the *mapping* instead of the mesh.

In order to achieve this, we recall the following properties of a smooth mapping ψ between surfaces Φ, Φ'. For corresponding curves $c(t)$ and $c'(t) = \psi(c(t))$ corresponding tangent vectors dc/dt, dc'/dt at a certain time instant are related by the linearization (the differential) $d\psi$, which is a linear mapping between corresponding tangent planes of Φ, Φ'. For *isometric* mappings, even more is true: $d\psi$ is an isometric linear mapping which maps also the tangential

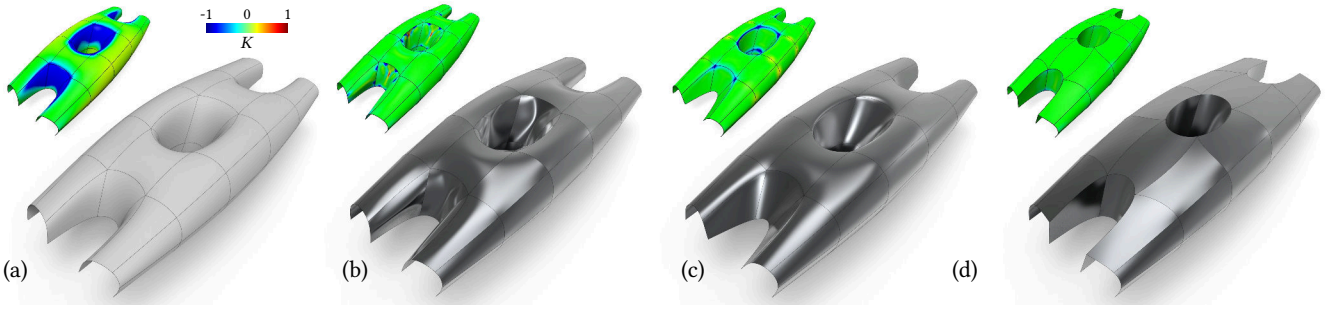

Fig. 8. *Degrees of freedom in computing developables.* (a) An input surface is subdivided into 46 individual patches and yields a wireframe future computations are based on. (b) Each 4-sided boundary is filled by a developable. One can see that in the negatively curved regions, this is not possible with a regular surface. (c) The same, but with boundaries movable and smoothness across boundaries as a soft constraint. The resulting surface is not entirely piecewise developable, since smoothness interferes with the developability constraint. (d) The same, with no smoothness across boundaries. Here the individual patches are developable. The inset images show the Gaussian curvature (green is zero; bounding box size equals 11).

components of d^2c/dt^2, d^2c'/dt^2 onto each other [do Carmo 1976, p. 239]. This property is equivalent to requiring

$$\langle \vec{v}, \tfrac{d^2}{dt^2} c \rangle = \langle d\psi(\vec{v}), \tfrac{d^2}{dt^2} c' \rangle \tag{7}$$

for all tangent vectors $\vec{v}$. For a discrete analogue of this property, surfaces Φ, Φ' are represented by meshes M, M'. Vectors $\vec{v}$, $d\psi(\vec{v})$ are replaced by corresponding edge vectors $v_l - v_j$ and $v'_l - v'_j$. Second derivatives of curves are replaced by 2nd differences of mesh polylines. This yields the energy

$$E_{map} = \sum_{\substack{v_i v_j v_k \text{ successive} \\ \text{vertices, } v_j v_l \in E}} (\langle v_i - 2v_j + v_k, v_l - v_j \rangle - \langle v'_i - 2v'_j + v'_k, v'_l - v'_j \rangle)^2.$$

$E_{map} \to \min$ expresses the discrete version of (7). If, say, M is fixed and M' is variable, a contribution $w_{map}E_{map}$ to the total energy will regularize the mapping and in turn, regularize the shape of M' (without regularizing the polylines contained in M').

Example. Figure 4 illustrates a conformal mapping of a given mesh to a planar mesh M'. Here variables are the x, y coordinates of vertices of M' as well as conformal factors λ_f for each face of the mesh, using the terminology of Equ. (2). The z coordinates of vertices of M' are set to zero. The computation of M' is performed by minimizing $w_{conf}E_{conf} + w_\lambda E_\lambda + w_{fair,M'}E_{fair,M'}$. Here the energy E_λ makes this conformal mapping as isometric as possible (in the ℓ^2 sense).

3 DISCRETE DEVELOPABLE SURFACES

We define a discrete developable surface as a quad mesh M which is isometric, in the sense of Equ. (1), to the quadrangulation of a planar domain, M'. We do not impose further constraints on either M or M'.

This definition is illustrated by Figure 7. It leads to meshes which exhibit the known characteristics of smooth developables. In this section we discuss the computation of discrete developables, the extraction of basic geometric properties such as rulings, and how our definition relates to other approaches to discrete developables.

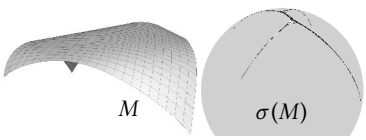

Fig. 9. The Gauss image $\sigma(M)$ of a mesh M is curve-like if and only if M itself is piecewise developable.

3.1 Computing developables

By definition, a mesh M is developable, if there is an isometric mesh M' which is contained in a plane. In different computational tasks either M or M' may be fixed, or both may be variable. Each task is solved by minimizing a suitable linear combination of the energies defined in Sec. 2.3. The weights of individual contributions to the total energy are summarized in the table in Fig. 22.

In Fig. 7, the unfolding M' is given, and M is found by a handle-based isometric bending. The shape of M is the result of minimizing an energy of the form $w_{iso}E_{iso} + w_{pos}E_{pos} + w_{fair,M}E_{fair,M}$, using a suitably modified Algorithm 1.

An analogous setup is used to compute the isometric deformation M' of a partly developable surface M in Fig. 10. Here M first undergoes a handle-based deformation, which yields a reference shape Φ. We now find a discrete surface M' isometric to M which approximates Φ, by minimizing an energy of the form $w_{iso}E_{iso} + w_{prox,1}E_{prox,1} + w_{prox,2}E_{prox,2} + w_{fair,M'}E_{fair,M'} \to \min$.

An unfolding M' of a developable surface M (see Fig. 12) is computed in an analogous way, using the energy $w_{iso}\ E_{iso} + w_{fair,M'}\ E_{fair,M'}$. Examples where both meshes M, M' undergo simultaneous optimization are shown throughout the paper, e.g. in Fig. 8.

Degrees of freedom and solvability of the optimization problem. Figure 8 illustrates the fact that imposing developability significantly reduces the number of degrees of freedom:

— In Fig. 8b, curvilinear quads are to be filled by developables. This is possible e.g. if such a quad lies on the boundary of its convex hull, but not always. Singularities develop, and our approach to discrete isometries is no longer guaranteed to accurately model isometries of smooth surfaces. Similarly we must expect the computation of Gauss curvature to break down in singularities. In any case such unsolvable cases are identified by pockets of nonzero Gauss curvature.

— In Fig. 8c, a wireframe, which is itself variable, is filled by developable patches, with smoothness across boundaries enforced.

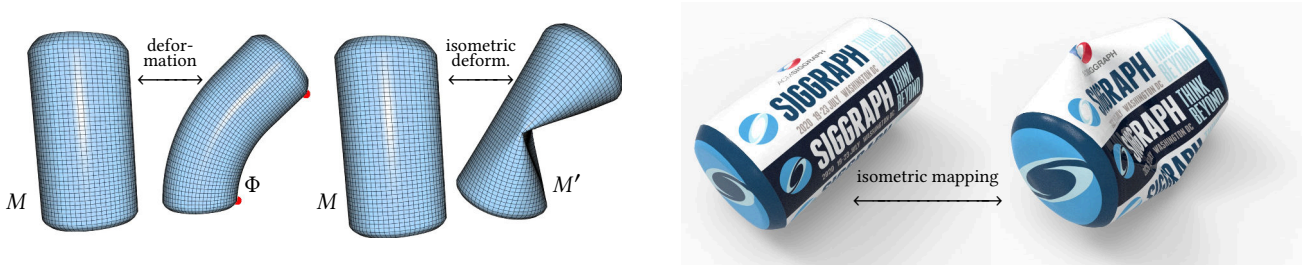

Fig. 10. Handle-based isometric editing. We show a deformed state Φ of a tin-can shaped surface M (the deformation model is that of *Rhinoceros*® and is not relevant for our result). We use the isometric constraint to simulate a surface M' close to Φ and isometric to M. Since M is piecewise developable, so is M'.

This requirement is not compatible with developability, which can be clearly seen by means of nonzero values of Gaussian curvature.

— Figure 8d illustrates a solvable problem: a wireframe, itself subject to optimization, is to be filled by a piecewise-developable surface.

3.2 Verifying developability

In order to verify developability of the surfaces we have computed, we firstly observe if they are isometric to the planar domains used in their computation. In addition we evaluate the Gauss curvature (which should be zero) and check if the surface contains rulings.

Estimating normals and curvatures. The Gauss image of a discrete surface M consists of its normal vectors, and can be computed by several methods. We could e.g. simply take the normal vectors n_f of 'black' parallelograms. An alternative is to compute vertex normal vectors n_i according to Sec. 2.2, by minimizing the energy $w_{norm}E_{norm} + w_{sym}E_{sym} + w_{fair}E_{fair} \to \min$. In any case, developability of M is characterized by the fact that the Gauss image is one-dimensional, see Fig. 9.

For computing curvatures, many more or less equivalent methods are available, for which we refer back to §2.2.1 and Fig. 6. We visualize Gauss curvature information in Figures 8 and 13.

Visualizing Rulings. In each edge midpoint, where two 'black' parallelograms meet, we can compute a ruling by intersecting the planes of the two parallelograms. This procedure discretizes the computation of rulings of smooth developables via intersection of two infinitesimally close tangent planes.

Another approach to rulings is based on an alternative criterion for developability. Gauss curvature, by its very definition as determinant of the shape operator s, vanishes if and only if s has a zero eigenvalue. We consider the conjugacy relation $\langle \vec{v}, s(\vec{r})\rangle = \langle s(\vec{v}), \vec{r}\rangle = 0$ between tangent vectors $\vec{v}, \vec{r}$ and compute a vector $\vec{r}$ conjugate to a given vector $\vec{v}$ via $\vec{r} = s(\vec{v})^{\perp}$. Then we have the following equivalence: The surface is developable in the point under consideration $\iff$ s is rank-deficient $\iff$ $\vec{r} = s(\vec{v})^{\perp}$ always lies in the zero eigenspace of s, regardless of $\vec{v}$.

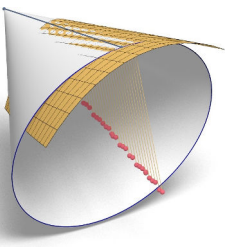

Fig. 11. Developables possess right circular *osculating cones* which in the generic case tangentially pass from one side of the developable to the other (analogous to curves' osculating circles). Existence follows from the fact that along a ruling, the principal radius of curvature is a linear function. We verify this by experiment.

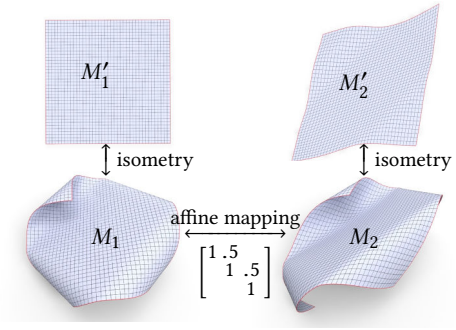

Fig. 12. A surface M_1 retains developability when mapped to a surface M_2 via an affine transformation. Here we illustrate the fact that no such simple correspondence exists between the respective unfoldings M_1', M_2'.

That zero eigenspace of the shape operator, indicating the principal direction corresponding to zero curvature, is the ruling. We use this fact to visualize rulings in Figure 13, where in the bottom row surfaces are near-developable. Vectors $\vec{r}$ nicely align along rulings.

Visualizing Osculating Cones. Smooth developables posses osculating cones of revolution, which are in second order contact with the surface along an entire ruling. Their existence corresponds to the fact that along a ruling, the principal radius of curvature (the inverse of the nonzero principal curvature) is a linear function of arc length. We were able to experimentally verify this, see Fig. 11.

3.3 Relation to other definitions of discrete developables

The relation of our definition of developables to the *orthogonal geodesic nets* proposed by Rabinovich et al. [2018a] is as follows. Our definition considers developables as a discrete version of a parametric surface which happens to be developable, but without any restriction on the nature of the parametrization. A discrete orthogonal-geodesic net, on the other hand, discretizes a special parametrization – both parameter lines are geodesics and orthogonal to each other. Such a parametrization does not have to be of constant speed (i.e., the faces do not need to approximate squares) but we can achieve this property by re-parametrizing each parameter separately.

If within our framework, we obtain a mesh whose derived checkerboard pattern has 'black' faces which are squares of the same size, then all face diagonals in the mesh are orthogonal and have the same length. A quad mesh consisting of diagonals of the original mesh

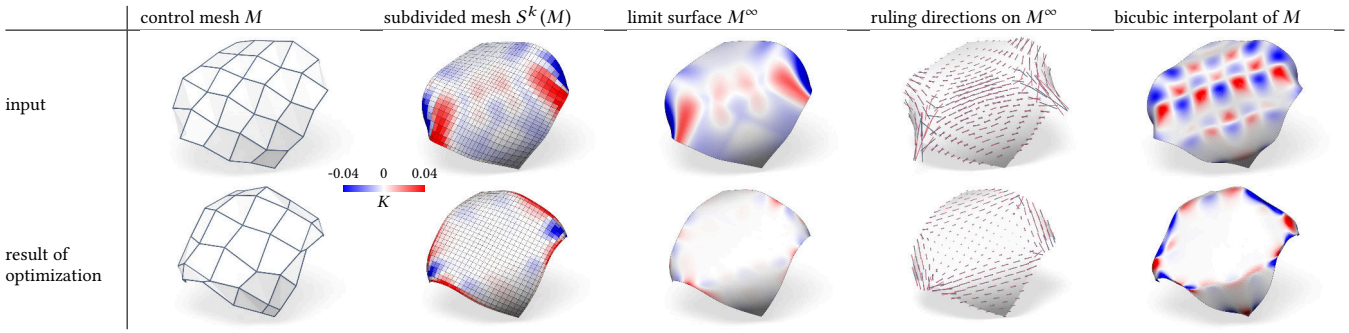

Fig. 13. *Developable spline surfaces.* A mesh M after k rounds of subdivision becomes a finer mesh $S^k(M)$. Here $k = 3$, and $S^k(M)$ is already a good approximation of the limit surface M^∞. By using an appropriate subdivision scheme we achieve that M^∞ is a bi-cubic B-spline surface. By optimizing M such that $S^k(M)$ is discrete-developable we make M^∞ a near-developable spline surface. We here show the situation before and after optimization. The color coding illustrates Gauss curvature (white is zero; bounding box size equals 10). We also show a bicubic *interpolant* defined by M, which exhibits particularly bad curvature behaviour, despite M already being discrete-developable. The column labelled "ruling directions" shows tangents conjugate to parameters lines which, in case of developability, arrange themselves along rulings.

therefore discretizes an orthogonal Chebyshev net, which is known to be geodesic and to occur precisely in developable surfaces. In this way a special checkerboard pattern discretizes the same smooth object as a discrete orthogonal-geodesic net.

However from the perspective of applications presented in this paper, it does not make sense to restrict ourselves to this special case, since the freedom of choosing the parametrization is a great advantage in practical modeling tasks. For example, we can align boundaries and features with edges (or with diagonals). Another instance where the greater freedom of our definition becomes evident is when we transform a developable by an affine transformation (see Fig. 12) or by offsetting. Such a transformation would not be possible with orthogonal-geodesic nets without a global remeshing. Also the watertight CAD models presented the next section illustrate the flexibility of our definition.

4 DEVELOPABLE SPLINE SURFACES AND WATERTIGHT CAD MODELS

It is not straightforward to express developable surfaces in the framework of B-splines implemented in most CAD systems. It can be done by segmenting developables into planar and ruled patches. However this decomposition might be much more complex than the first visual appearance of the developable suggests. From the viewpoint of modeling, it would be highly preferable not having to worry about it, and being able to use spline surfaces whose control points are not aligned with rulings. This problem has been described as unsolved recently by [Rabinovich et al. 2018a]. Our solution, which represents a passage from discrete differential geometry to computer-aided geometric design, is a main contribution of the present paper. We show how to perform modeling with watertight spline surfaces which are developable to an extent sufficient for applications.

The idea is is illustrated by Fig. 13 and Alg. 2. We take a control mesh M and derive a B-spline surface M^∞ from it. An alternative way to produce this spline surface is to construct it as limit surface of a certain stationary subdivision rule S. The idea is now to use a small number k of rounds of subdivision to create a fine mesh $S^k(M)$ which approximates the final B-spline surface M^∞ with sufficient numerical accuracy, but is still small enough to be subjected to optimization, in order to achieve developability.

Algorithm 2: Modeling with a developable spline surface M^∞

Data: Initial control mesh M, subdivision level k (e.g. $k = 2$)
Macro $S^k(mesh\ M)$
  k rounds of subdivision applied to each rectangular patch of M;
Initialize 2D mesh M' combinatorially equivalent to $S^k(M)$;
Set up energy E_{iso} expressing isometry of $S^k(M)$ and M';
Set up energies $E_{fair,S^k(M)}$ and $E_{fair,M'}$;
while *user is imposing constraints on design surface* **do**
  Express user's constraints as energy $E_{pos}(S^k(M))$;
  Use a version of Alg. 1 on variables M, M' to minimize
  $w_{iso}E_{iso} + w_{pos}E_{pos} + w_{fair,S^k(M)}E_{fair,S^k(M)} + w_{fair,M'}E_{fair,M'}$;
end
return *the bicubic spline surface* M^∞ *whose control mesh is* M;

The subdivision scheme we employ is an extension of the well known Catmull-Clark scheme which in its combinatorially regular case converges to bicubic spline surfaces [Peters and Reif 2008]. We decompose the given mesh into quadrilateral patches bounded by mesh polylines (the example of Fig. 13 has only one such patch). For each patch we have 4 boundary polylines. The vertices $\{p_j\}_{j=1,\dots,M}$ of such a boundary polyline are the control points of a cubic B-spline curve (with uniform interior knots) interpolating the endpoints p_1 and p_M. This curve is alternatively produced as the limit curve of a stationary subdivision rule whose stencil is derived from Boehm's knot insertion formula for B-splines [Prautzsch et al. 2002]: For $i \in \{1,\dots,2M-3\}$, we let

$$p_i^{\text{new}} = \sum_{j=1}^{M} \alpha_{ij} p_j, \text{ where } \alpha_{ij} = \frac{1}{16}\begin{bmatrix} 16 & & & & & & \\ 8 & 8 & & & & & \\ & 12 & 4 & & & & \\ & 3 & 11 & 2 & & & \\ & & 8 & 8 & & & \\ & & 2 & 12 & 2 & & \\ & & & & \ddots & & \end{bmatrix}, \tag{8}$$

cf. [Shen et al. 2014, Fig. 6]. In the interior of the polyline, this is the usual cubic Lane-Riesenfeld subdivision rule. By applying it first to

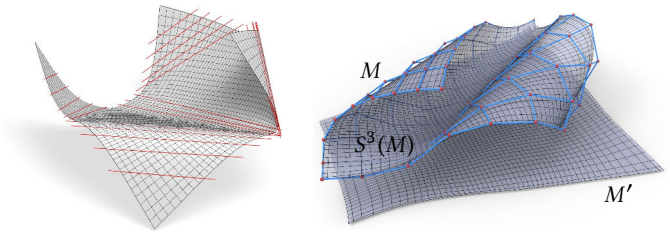

Fig. 14. Left: A near-developable spline surface, its parameter lines and rulings. Right: Coarse control mesh M (only edges shown), its subdivision $S^3(M)$, and the unfolding M' of $S^3(M)$.

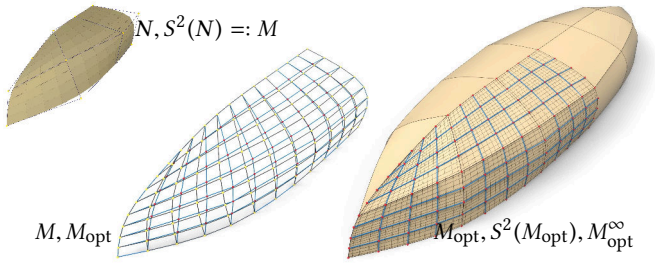

Fig. 15. *A watertight CAD model.* A mesh M is optimized such that a subdivided mesh $S^2(M)$ becomes a discrete developable. We here show the control mesh M and its optimized version M_{opt} as well as the near-developable B-spline surface M^∞_{opt}. Actually, M has been created by subdivision in the first place from a very coarse mesh N (top left) which however does not have sufficiently many degrees of freedom to be used directly.

the rows and subsequently to the columns of an $M \times N$ rectangular control point arrangement, we create $(2M-3) \times (2N-3)$ control points in the next level of recursion. Apart from the 4 boundary rows, this amounts to Catmull-Clark subdivision.

The collection of bicubic surface patches produced in this way is watertight, because neighbouring patches share a boundary which is interpolated on both sides. The resulting composite surface however is smooth across patch boundaries only if this property is enforced by optimization.

The geometric modeling procedure is summarized by Algorithm 2. The user modifies the control mesh M until satisfied with the B-spline surface M^∞ derived from it, while in the background a refined mesh $S^k(M)$, which closely approximates M^∞, is optimized to be discrete-developable. For that purpose an auxiliary flat mesh M' which is isometric to $S^k(M)$, has to be computed, cf. Fig. 14.

Figures 13, 15 show examples. Figure 13 in particular illustrates the fact that the control mesh M does not have to be fair for $S^k(M)$ to be a fair discrete-developable surface.

Remark. The surfaces produced by this method are developable only up to a certain amount of numerical inaccuracy. If exact developability is required, the surface computed in this way could be decomposed into its flat parts and ruled parts, and subsequently be approximated by exact developables using e.g. the methods of [Tang et al. 2016].

Remark: Such a subdivision method would not work for the discrete-orthogonal nets of [Rabinovich et al. 2018a], since there are no spline surfaces with orthogonal-geodesic parameter lines except for cylinders (see the appendix).

5 RESULTS

Cutting and gluing. When computing a developable M from its unfolding M', we might force pairs of vertices to lie on top of each other. Adding such constraints to our optimization procedure simulates gluing, if the selected vertex pairs define an arclength-preserving correspondence between boundaries in M'. Since our method does not require that edges follow special curves on developables, we can align edges with boundaries and features. It might even be desirable to represent features not by the edges of a mesh, but by *diagonals* of faces, since the isometry condition (1) operates on diagonals rather than on edges. Examples are shown by Figures 18 and 16.

Cone points. By appropriately gluing planar domains one achieves surfaces which are developable except in individual singularities, where the intrinsic metric behaves like that of a non-flat cone. Fig. 17 exhibits a cone point with angle sum less than 360 degrees, and another cone point with negative Gaussian curvature concentrated in that point, i.e., with an angle sum greater than 360 degrees. We include this example to show the flexibility of our approach.

D-forms and non-convex generalizations. The shapes obtained by gluing together two planar domains with the same perimeter have been of interest since the *sphericon* was proposed [Phillips 1999] and the name *D-form* for such shapes was coined by [Wills 2006]. The domains in question can be n-gons or can be smooth; classically convexity is assumed. It is well known that a unique convex surface isometric to the glued domains exists, if that union is intrinsically convex. This means that in corresponding points, the sum of curvatures is nonnegative (if we use the convention that curvatures of convex curves are always nonnegative). See [Bobenko and Izmestiev 2008] for an algorithmic solution in the discrete case. Here we are interested in the non-convex case and the shape of "D-forms" obtained by gluing domains which violate the curvature condition. Figure 19 gives some examples.

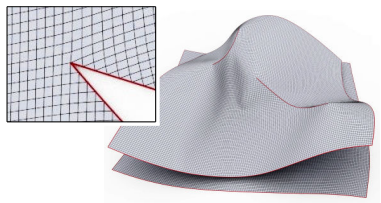

Fig. 16. The features of this cut and glue example are not aligned with edges of the mesh, but with diagonals. In this way lengths are preserved more accurately.

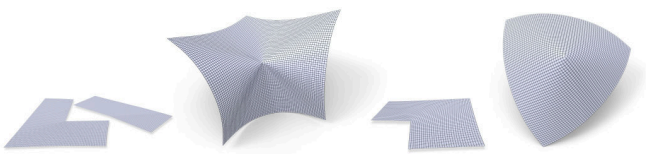

Fig. 17. Geometric singularities (cone points) can be achieved by appropriate gluing of planar domains.

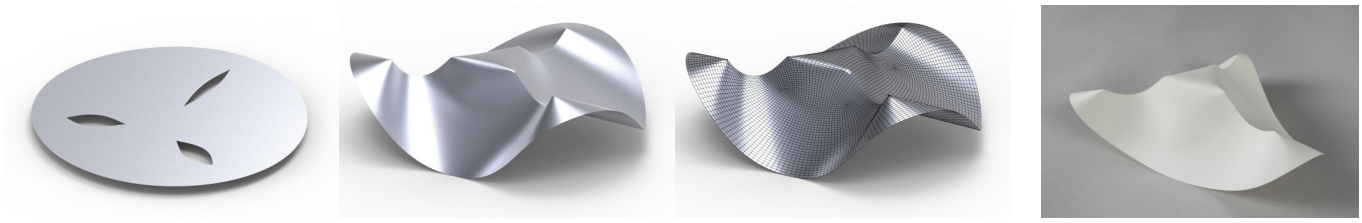

Fig. 18. *Cutting and Gluing.* This example of computing a developable from its unfolding involves the gluing shut of holes. From left, we show the unfolding, the developable, its mesh representation, and a photo of a paper model. Here all boundaries, including holes, are aligned with edges of the mesh. The combinatorial singularities in the meshes which inevitably occur are not noticeable in the final rendering.

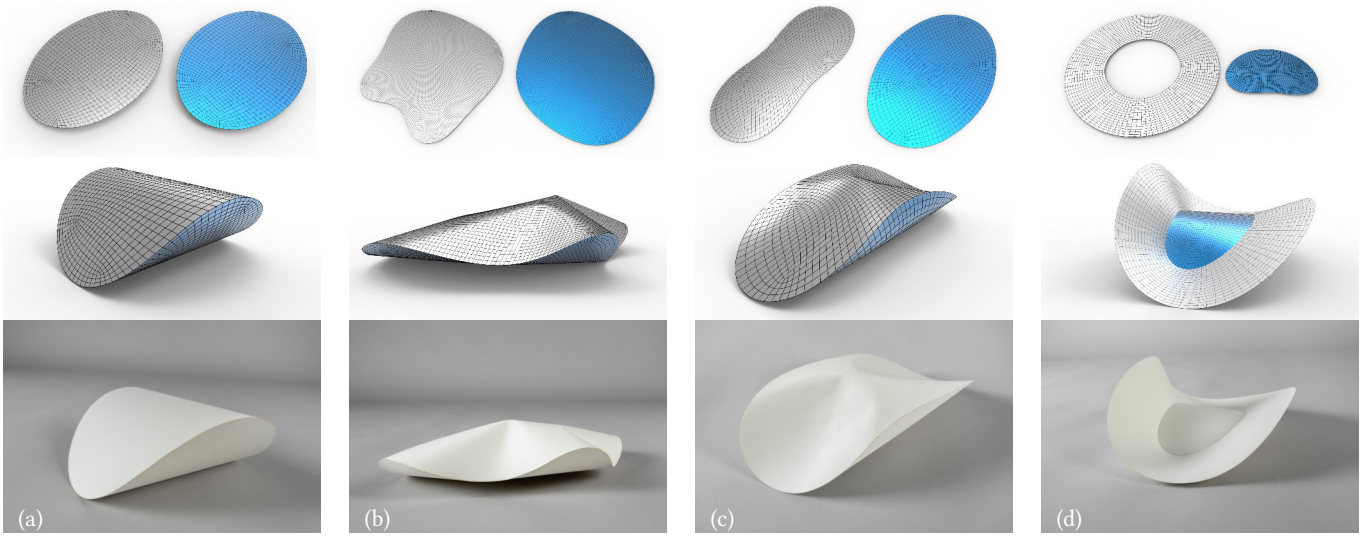

Fig. 19. *D-Forms and their generalizations.* The two convex planar domains in (a) with the same perimeter can be glued together along their respective boundaries to form a unique convex surface. If the domains are not convex, a unique convex surface only exists if in corresponding boundary points, curvatures κ, κ' obey the condition $\kappa + \kappa' \geq 0$. This is not the case in (b) and (c), where one can observe the emerging singularities on non-convex piecewise-developable surfaces. Subfigure (d) shows a further kind of gluing domains along boundary components. In all cases we show, from top to bottom, the unfolding, the (generalized) D-form, and a photo of a paper model.

Approximation with developables. For any given non-developable surface, we may ask for a developable or piecewise-developable surface approximating it. This is a difficult problem and one can approach it from various angles, see e.g. [Mitani and Suzuki 2004]. Here we consider only a sub-problem: We assume that a reference shape Φ has been segmented into patches, and we wish to approximate the individual patches by developables. The simplest case is approximating one reference shape by a single developable. From there, it is only a small step to simultaneously approximate a segmented reference shape by a union of patches glued together along their boundaries. Creating the segmentation is beyond the scope of this paper. To approximate a reference shape Φ, we do the following:

- We represent Φ by a mesh M. M is subdivided into patches M_j which correspond to the patches on Φ.
- We conformally and near-isometrically map each patch M_j to a planar mesh M'_j, storing the patch connectivity in gluing data for the unfoldings M'_j.
- We simultaneously optimize all M_j and M'_j such that the correspondence between M_j and M'_j becomes isometric and gluing is respected. M_j must be in proximity with Φ.

Examples are shown by Fig. 21. The same procedure can be applied to increase the developability of a surface, if we set aside the question of segmentation. We should mention that the problem of making a surface developable has also been studied by [Stein et al. 2018], where a mesh is driven towards piecewise-developability by clustering Gauss curvature along feature lines. This method is more rigid than our approach, since edges are aligned with rulings and feature lines.

Deformations. Figure 26 shows various stages in a deformation sequence, computed analogous to Fig 10. It is important to appreciate the fact that developables occur naturally as the shapes of thin sheets of inextensible material. For this reason, developables and their deformations can be used to compute believable geometric shapes without any simulation of the actual physics involved, see e.g. Fig. 21.

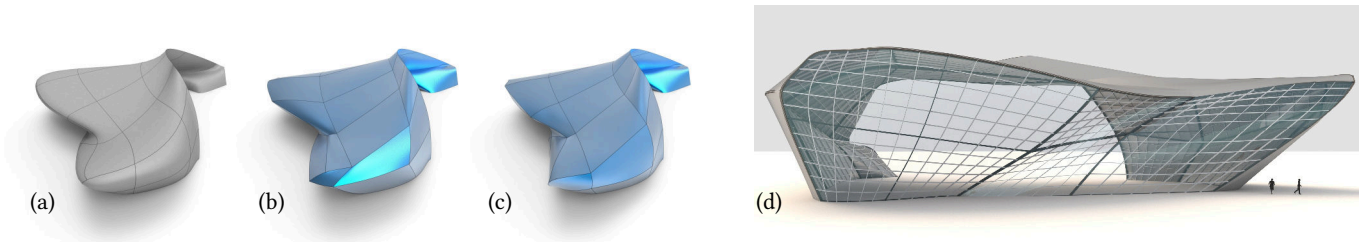

Fig. 20. *Paneling freeform designs in architecture.* Here a detail of the NHHQ skyscraper project by Zaha Hadid Architects (a) is being approximated by a piecewise-developable (b). For the surface in subfigure (c), more smoothness across patch boundaries has been required. That piecewise-developable surface is realized as a piecewise single-curved architectural freeform skin in (d).

This observation extends to materials which do stretch, like metals, but which nevertheless assume developable shapes when they revert to a state of minimal tension. Figure 10 shows a deformed can. It apparently does not resume its original cylindrical shape because in several places the limits of elasticity have been exceeded and deformation remains permanent. No such property was present in the computation.

Paneling architectural freeform shapes. A specific and important special case of the approximation problem is the paneling of architectural freeform shapes. Eigensatz et al. [2010] considered surface segmentation into different kinds of panels, such that by using simple panels, and panels manufacturable from the same mold, the overall cost is reduced. Since developable panels are much less costly than double-curved ones, a combination of their segmentation with our approximation technique deserves investigation. A similar approach is pursued by Gavriil et al. [2018] who perform segmentation by an analysis of the Gauss image. Once segmentation is performed – either automatically or by the user – we can use our methods to approximate the given design shape by a piecewise-developable surface. Figure 20 shows a result.

6 DISCUSSION

Verification of results. We regard reconstructing the well known geometric properties of developables as the main tools in verifying the validity of our results. These include the visualization of rulings (see Figures 13, 14, left) and even osculating cones (see Fig. 11). We also check the values of Gauss curvature (Figures 8 and 13) and we give statistics of length distortion in the table of Fig. 22. The

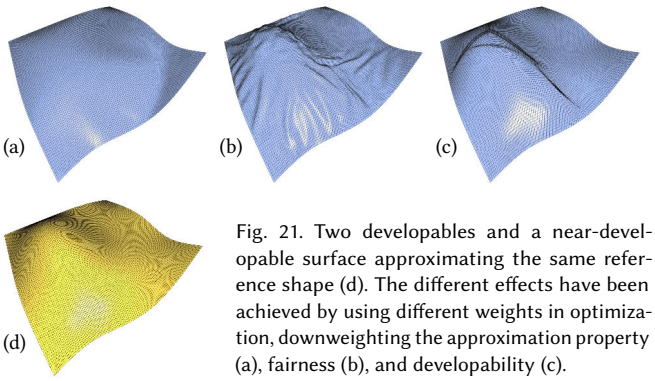

Fig. 21. Two developables and a near-developable surface approximating the same reference shape (d). The different effects have been achieved by using different weights in optimization, downweighting the approximation property (a), fairness (b), and developability (c).

Fig.	$\lvert V\rvert^*$	$\lvert F\rvert^*$	#var	w_{iso}	$w_{prox,1}$		w_{fair}	L-Err.	A-Err.	#it	T
				w_{pos}	$w_{prox,2}$			[%]	[%]		[s]
1$^{\times}$	15k	15k	81k	1	1	.001	.001 .1	.29	.038	20	53.2
4**	1.1k	1.1k	3k				.01	4.2	7.8	20	0.4
6**	29k	29k	87k				.01	n/a	n/a	5	10.5
7	1.9k	1.8k	6k	1	1		.1	.093	.18	10	1.2
8b	50k	47k	250k	1	1		.1	.38	.54	20	124.2
8c	48k	47k	250k	1			.1	1.8	1.1	10	412.3
8d	50k	47k	250k	1	1		.1	.035	.26	10	65.8
10	4.1k	4.1k	12k	1	1	.001	.001 .1	.38	.48	10	7.2
12	1.7k	1.6k	5k	1			.1	.11	.27	10	1.0
13	36	25	4k	1			.01	.12	.21	10	0.9
15	150	64	11k	1	1		.01	.026	.038	10	2.9
16	14k	14k	43k	1	1		.1	.065	.087	10	13.4
18	8.4k	8.2k	25k	1	1		.1	.081	.17	10	7.7
19a	2.5k	2.4k	7k	1	1		.1	.0042	.0057	10	2.5
19b	11k	11k	33k	1	1		.1	.15	.23	10	9.0
19c	2.5k	2.4k	7k	1	1		.1	.054	.031	20	5.2
19d	2.9k	2.8k	9k	1	1		.1	.023	.0072	20	7.4
20b	28k	26k	160k	1	1	.01	.01 .1	.041	.088	10	42.9
20cd	28k	26k	160k	1	1	.01	.01 .1	.058	.13	10	43.2
25	1.5k	1.4k	4k	1	1		.1	.012	.021	20	4.2
26$^{\times}$	20k	18k	98k	1	1	.001	.001 .1	.21	.055	20	67.8

* For Figs. 13, 15, both $\lvert V\rvert$ and $\lvert F\rvert$ refer to the control mesh M.
$^{\times}$ Data for one subfigure. All subfigures have similar data
** In Fig. 4, $w_{conf} = 1$, $w_{\lambda} = 0.1$. In Fig. 6, $w_{norm} = w_{sym} = 1$.

Fig. 22. Overview of the size of optimization problems solved for the examples in the paper. We also give the weights of energies used for optimization and the computation time in seconds. The isometric property of mappings is verified by the relative L^2 error of edgelengths, defined as $\|L' - L\|/\|L\|$, where L, L' refer to the vector of edgelengths of meshes M, M', and $\|\cdot\|$ is the Euclidean norm. We also show the relative error in the area of faces, which for nonplanar quads is computed via a subdivision into triangles.

obtained results are useful for practical applications, e.g. paneling freeform architectural designs (Fig. 20) and engineering applications (Fig. 15). For several developables in this paper, especially cut and glue examples and D-forms, we verified the obtained shapes experimentally by building paper models, see Figures 18, 19, 25.

Limitations. The rigid nature of developables is still noticeable in all applications, which is a limitation not of the method, but of the subject matter. We found that our method can produce developable surfaces and also (if they exist) isometric mappings to a satisfactory extent. We also compute *conformal* and as-rigid-as-possible

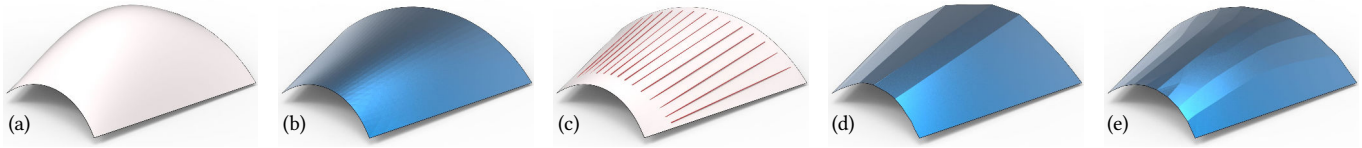

Fig. 23. *Properties of ruling-based methods for computing developables.* Here we highlight how a ruling-based method for computing developables behaves when forced to perform a task it was not designed to do, namely filling the boundary of the quadrilateral surface patch (a) with a developable surface. The solution (b), computed with our method, has rulings intersecting the boundary at nonzero angles (c). Stein et al. [2018] model discrete developables whose rulings are aligned with the edges of the underlying mesh. Their method cannot recreate the solution. It instead produces surfaces with creases whose location is mesh-dependent (d,e).

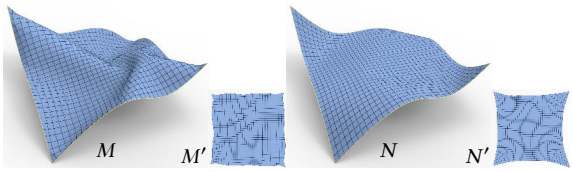

Fig. 24. *The behaviour of orthogonal-geodesic nets.* Here developables M, N (with unfoldings M' resp. N') interpolate the same boundary curve c. M is an orthogonal-geodesic net according to [Rabinovich et al. 2018a], simulating a rectangular piece of paper pressed onto c. N is computed with our method with N' still variable, simulating paper of any shape interpolating c. The relative length errors (cf. Fig. 22) are .015 and .0003 for M resp. N, consistent with the fact that orthogonal-geodesic nets have fewer degrees of freedom than our developable surfaces.

mappings, e.g. to initialize an isometric mapping. We found that only a moderate effort was required to reproduce the kind of results achieved by [Liu et al. 2009] in such cases where near-isometric mappings exist. While our method can treat conformal mappings of more challenging shapes like the well known Max Planck's head mesh, imposing the condition of near-isometry will cause overfoldings, if we do not add additional energies to prevent them. Since we do not consider conformal mapping a contribution of this paper, we did not pursue this subject further.

Implementation Details. The target functionals according to §2.3 are optimized by a Levenberg-Marquardt method according to [Madsen et al. 2004, §3.2]. The damping parameter was set to 10^{-6}. As a stopping criterion we used a small value of the energy. The initial values for optimization are often obvious like in deformation tasks and in approximation problems. As a general rule an initial solution can be a surface which fits the problem without the side-condition of being isometric to the reference mesh. Our implementation in C++ uses the data structures of *OpenMesh* [Botsch et al. 2002] and the TAUCS library for sparse linear solvers [Toledo 2003].

Detailed statistics are provided by the table in Fig. 22. These computation times refer to an Intel Xeon E5-2687W 3.0GHz processor without parallel processing or other acceleration techniques. Fig. 22 also shows the weights of the individual energies which make up the target functional. We conducted numerical experiments to check the sensitivity of our method w.r.t. the choice of weights. We generally observed robustness w.r.t. the choice of w_{iso}, w_{pos}. E.g. in Fig. 7 no perceptible change in the results is observed if those weights range in the interval $[0.1, 10]$. Our method is more sensitive w.r.t. the choice of w_{fair}, which is a fact accounted for by Algorithm 1.

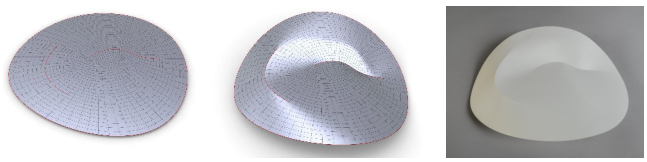

Fig. 25. *Curved Folds.* By Cutting holes of zero width and gluing we create developables with curved folds. The folded shape is enforced by using the energy E_{pos} to move points to prescribed positions.

Comparison with Previous Work. Figure 24 illustrates how a discrete orthogonal-geodesic net M according to [Rabinovich et al. 2018a] interpolates a given boundary, and how such an interpolation problem is solved by a quad mesh N which is developable in our sense. The edges of the latter do not have to follow a network of geodesics, but for the former, the rectangular combinatorics of M already fixes its unfolding M' to be rectangular. As expected, N enjoys better developability, when measured via the relative length error. In fact, the mesh M does not look like paper but like some fabric which allows a small amount of stretching. We can recreate such a result with our method by constraining the development N' to be rectangular – in that case N would look just like M, with the same length error. Developables according to [Stein et al. 2018] enjoy even fewer degrees of freedom, as detailed in Fig. 23.

Future Research. There are some obvious directions of future research. One is the segmentation problem when approximating arbitrary surfaces with piecewise-developable surface. The first step would be to investigate how previous work on segmentation can be combined with our approximation procedures. Other directions of future research include incorporating more properties of materials, thereby extending the class of mappings under consideration. Further, it would be interesting to develop a theory of curvatures based on the Gauss map introduced in this paper, in particular because there is already a relation to [Bobenko et al. 2010].

ACKNOWLEDGMENTS

This work was supported by the SFB-Transregio programme *Discretization in geometry and dynamics*, through grant I2978 of the Austrian Science Fund. Caigui Jiang, Florian Rist, and Cheng Wang were supported by KAUST baseline funding. We wish to thank Jonathan Schrodt for his contribution in the project's initial phase.

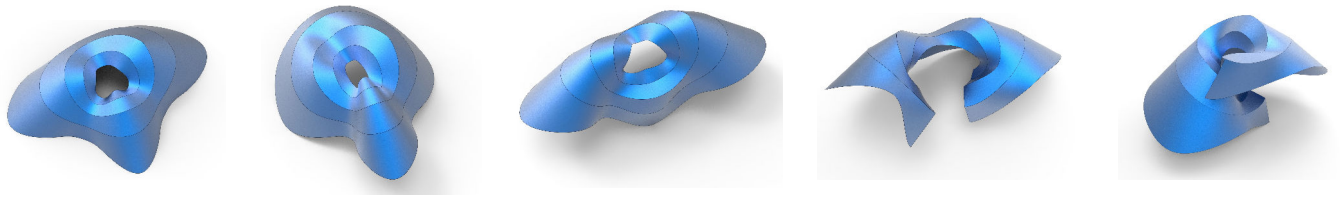

Fig. 26. *Isometric deformation.* Once a piecewise-developable surface is found, handle-based editing can be used to simulate isometric bending.

REFERENCES

Basile Audoly and Yves Pomeau. 2009. *Elasticity and Geometry: From hair curls to the nonlinear response of shells.* Oxford University Press.

Alexander Bobenko, Helmut Pottmann, and Johannes Wallner. 2010. A curvature theory for discrete surfaces based on mesh parallelity. *Math. Annalen* 348 (2010), 1–24.

Alexander I. Bobenko and Ivan Izmestiev. 2008. Alexandrov's theorem, weighted Delaunay triangulations, and mixed volumes. *Ann. Inst. Fourier* 58 (2008), 447–505.

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.

Mario Botsch, Stephan Steinberg, Stephan Bischoff, and Leif Kobbelt. 2002. OpenMesh: A Generic and Efficient Polygon Mesh Data Structure. Proc. OpenSG Symposium. *https://graphics.uni-bielefeld.de/publications/openmesh.pdf.*

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. Shape-Up: Shaping Discrete Geometry with Projections. *Computer Graphics Forum* 31, 5 (2012), 1657–1667. Proc. Symposium Geometry Processing.

Frédéric Cazals 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.

Sebastian Claici, Mikhail Bessmeltsev, Scott Schaefer, and Justin Solomon. 2017. Isometry-Aware Preconditioning for Mesh Parameterization. *Computer Graphics Forum* 36, 5 (2017), 37–47. Proc. Symposium Geometry Processing.

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

Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy Mitra, Helmut Pottmann, and Mark Pauly. 2010. Paneling Architectural Freeform Surfaces. *ACM Trans. Graph.* 29, 4 (2010), 45:1–10.

Danielle Ezuz, Justin Solomon, and Mirela Ben-Chen. 2019. Reversible Harmonic Maps between Discrete Surfaces. *ACM Trans. Graph.* 38, 2 (2019), 15:1–12.

Konstantinos Gavriil, Alexander Schiftner, and Helmut Pottmann. 2018. Optimizing B-spline surfaces for developability and paneling architectural freeform surfaces. arXiv 1808.07560.

Philip Hartman and Louis Nirenberg. 1959. On spherical image maps whose Jacobians do not change signs. *Amer. J. Math* 81 (1959), 901–920.

Tim Hoffmann, Andrew Sageman-Furnas, and Max Wardetzky. 2017. A discrete parametrized surface theory in $\mathbb{R}^3$. *Int. Math. Research Notices* (2017), 4217–4258.

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

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2009. A local/global approach to mesh parametrization. *Computer Graphics Forum* 27, 5 (2009), 1495–1504. Proc. Symposium Geometry Processing.

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.

Jun Mitani and Hiromasa Suzuki. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. In *ACM SIGGRAPH 2004.* 259–263.

Chi-Han Peng, Caigui Jiang, Peter Wonka, and Helmut Pottmann. 2019. Checkerboard Patterns with Black Rectangles. *ACM Trans. Graph.* 38, 6 (2019), 171:1–13.

Jörg Peters and Ulrich Reif. 2008. *Subdivision Surfaces.* Springer.

Tony Phillips. 1999. The differential geometry of the sphericon. Feature Column 10/99, American Math. Soc. *www.ams.org/samplings/feature-column/fcarc-sphericon1.*

Nico Pietroni, Marco Tarini, and Paolo Cignoni. 2010. Almost Isometric Mesh Parameterization through Abstract Domains. *IEEE TVCG* 16, 4 (2010), 621–635.

Helmut Pottmann, Yang Liu, Johannes Wallner, Alexander Bobenko, and Wenping Wang. 2007. Geometry of Multi-layer Freeform Structures for Architecture. *ACM Trans. Graph.* 26, 3 (2007), 65:1–11.

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

Hartmut Prautzsch, Wolfgang Boehm, and Marco Paluszny. 2002. *Bézier and B-Spline Techniques.* Springer.

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

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

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

Kenneth Rose, Alla Sheffer, Jamie Wither, Marie-Paule Cani, and Boris Thibert. 2007. Developable Surfaces from Arbitrary Sketched Boundaries. In *Proc. Symposium Geometry Processing.* 163–172.

Josua Sassen, Behrend Heeren, Klaus Hildebrandt, and Martin Rumpf. 2020. Geometric optimization using nonlinear rotation-invariant coordinates. *Comp. Aided Geom. Des.* 77, Article 101829 (2020).

Robert Sauer. 1970. *Differenzengeometrie.* Springer.

Jingjing Shen, Jiří Kosinka, Malcolm A. Sabin, and Neil A. Dodgson. 2014. Conversion of Trimmed NURBS Surfaces to Catmull-Clark Subdivision Surfaces. *Comput. Aided Geom. Des.* 31 (2014), 486–498.

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

Olga Sorkine and Mark Alexa. 2007. As-rigid-as-possible surface modeling. In *Proc. Symposium Geometry Processing.* Eurographics, 109–116.

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

Kenneth Stephenson. 2005. *Introduction to Circle Packing.* Cambridge Univ. Press.

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

Sivan Toledo. 2003. TAUCS, A Library of Sparse Linear Solvers. *www.tau.ac.il/~stoledo/taucs.*

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.

Tony Wills. 2006. D-Forms: 3D Forms from Two 2D Sheets. In *Proc. Bridges.* 503–510.

APPENDIX

LEMMA. *A spline surface $x(u,v)$ whose parameter lines are geodesic and orthogonal to each other is cylindrical of the form $x(u,v) = a(u) + b(v)$ where $a(u)$ is a planar curve and $b(v)$ parametrizes a straight line orthogonal to it (or possibly vice versa).*

PROOF. Using subscripts for partial derivatives, the orthogonality mentioned here is expressed as $\langle x_u, x_v\rangle = 0$. Parameter lines are geodesic $\iff$ planes $[x_u, x_{uu}]$ and also $[x_v, x_{vv}]$ are orthogonal to the surface $\iff$ conditions $\langle x_{uu}, x_v\rangle = \langle x_{vv}, x_u\rangle = 0$ hold.

By differentiating the orthogonality condition (and using the geodesic property) we get $\langle x_{uv}, x_v\rangle = \langle x_{uv}, x_u\rangle = 0$, which implies $\partial_v \|x_u\|^2 = 2\langle x_{uv}, x_u\rangle = 0$. I.e., the length of the vectors x_u does not depend on v. For fixed u, x_u moves in a sphere. B-spline surfaces are piecewise polynomial, and so is x_u. The only polynomial curves contained in spheres are constant, so x_u does not depend on v. Then $x_{uv} = 0$ implies that the surface has the form $x(u,v) = a(u) + b(v)$, with curves a, b.

By orthogonality, for all u, v we have $\langle x_u, x_v\rangle = \langle \dot{a}(u), \dot{b}(v)\rangle = 0$. Unless $a(u)$ is a straight line, $\dot{a}(u)$ assumes at least two linearly independent values. If $\dot{b}(v)$ is to be orthogonal to both of them, $\dot{b}$ cannot change direction and $b(v)$ is a straight line. In turn, all $\dot{a}(u)$'s are orthogonal to this line, and the curve $a(u)$ lies in a plane. □