Shape Approximation by Developable Wrapping

ALEXANDRA ION, MICHAEL RABINOVICH, PHILIPP HERHOLZ, and OLGA SORKINE-HORNUNG,
ETH Zurich, Switzerland

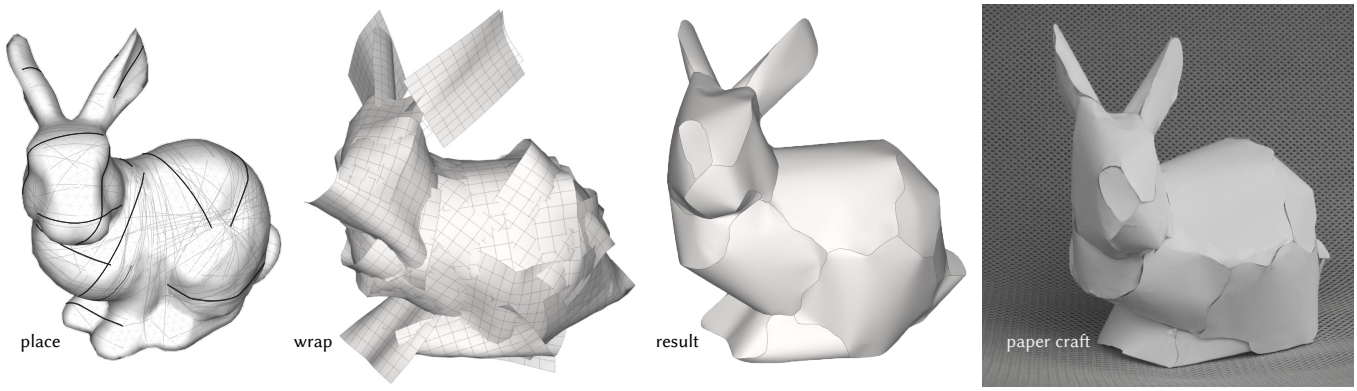

Fig. 1. Our algorithm converts a given 3D shape into a piecewise developable surface that approximates it. We wrap the model in developable patches and nonlinearly project the original mesh onto the developables. This makes our algorithm mesh-independent and allows users to choose the approximation quality. Each patch in the end result can be fabricated using sheet material, e.g., paper.

We present an automatic tool to approximate curved geometries with piecewise developable surfaces. At the center of our work is an algorithm that wraps a given 3D input surface with multiple developable patches, each modeled as a discrete orthogonal geodesic net. Our algorithm features a global optimization routine for effectively finding the placement of the developable patches. After wrapping the mesh, we use these patches and a non-linear projection step to generate a surface that approximates the original input, but is also amendable to simple and efficient fabrication techniques thanks to being piecewise developable. Our algorithm allows users to steer the trade-off between approximation power and the number of developable patches used. We demonstrate the effectiveness of our approach on a range of 3D shapes. Compared to previous approaches, our results exhibit a smaller or comparable error with fewer patches to fabricate.

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Mesh geometry models**;

Additional Key Words and Phrases: developable surfaces, discrete differential geometry, geodesic nets, shape modeling

ACM Reference Format:
Alexandra Ion, Michael Rabinovich, Philipp Herholz, and Olga Sorkine-Hornung. 2020. Shape Approximation by Developable Wrapping. *ACM Trans. Graph.* 39, 6, Article 200 (December 2020), 12 pages. https://doi.org/10.1145/3414685.3417835

Authors' address: Alexandra Ion; Michael Rabinovich; Philipp Herholz; Olga Sorkine-Hornung , ETH Zurich, Department of Computer Science, Universitätstrasse 6, Zurich, 8092, Switzerland.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2020 Copyright held by the owner/author(s).
0730-0301/2020/12-ART200
https://doi.org/10.1145/3414685.3417835

1 INTRODUCTION

Shapes that can be fabricated by bending or stamping sheets of material (e.g., metal) are highly relevant to the manufacturing industry, since these fabrication processes are more energy efficient than stretching materials. Such shapes are known as developable surfaces, best illustrated by a sheet of paper—at any point it can only be bent in one direction and cannot stretch or compress. Unfortunately, developable surfaces are difficult to design due to their globally constrained geometry, a fact that is likely evident to anyone trying to wrap a noncuboid-shaped gift.

In an attempt to relieve users from this difficult design problem, various research works investigate algorithms to automatically convert 3D shapes into piecewise developable representations. One approach is to deform a given shape until it becomes developable, i.e., until its Gaussian curvature vanishes [Wang and Tang 2004] and creases emerge [Stein et al. 2018]. Other approaches automatically produce individual developable strips [Massarwi et al. 2007; Mitani and Suzuki 2004] or patches [Shatz et al. 2006] that approximate the input shape and can be cut and assembled. Such piecewise approximation problems with individual patches are twofold: they involve (1) *locally* fitting a good developable representation and (2) *globally* finding a good placement for the individual developables. The aforementioned methods cover the shape locally with simple triangle strips (i.e., torsal developable surfaces [Pottmann and Wallner 2001]) in a greedy manner: the placement of their developables relies on prior mesh segmentation (e.g., part-based segmentation) and subsequent decomposition into torsal patches.

In this paper, we propose a novel, fully automatic approximation algorithm that improves on both the local and the global aspects of the problem. Locally, we use *general* developable surfaces instead of solely torsal patches, by employing discrete orthogonal geodesic nets (DOGs) [Rabinovich et al. 2018a,b] in our algorithm. The DOG

model is known to capture both extrinsic and intrinsic deformations of developable surfaces without suffering from deformation locking [Chapelle and Bathe 1998; Alessio 2012; Tang et al. 2016]. This model eliminates the need for combinatorial partition of the developable surface into planar and torsal patches and results in a better coverage, since it allows to optimize over the entire developable shape space. Globally, we determine where and how many DOGs to place by performing a *global optimization* that is based on developables, i.e., the problem at hand, rather than on more generic mesh segmentation methods.

We demonstrate how the results of our algorithm are at least comparable with or better than previous methods in terms of approximation error while requiring fewer patches, which eases fabrication (see Fig. 3). We compare our method with previous works in more detail in Section 4.

We detail our algorithm in Section 3 and summarize it below, as illustrated in Fig. 1. The key to our algorithm is to wrap the input model tightly with DOGs, benefiting from their flexibility. Finding a good initial guess of where and how many DOGs to fit is not a trivial step. We initialize the global placement by creating an overcomplete set of geodesics, shown in Fig. 1 (*place*), from which we select a subset using a global multi-labeling graph-cut optimization. The selected geodesics guide the placement of our DOG-fitting optimization process (Fig. 1 *wrap*). Our DOG fitting is devised to prevent over-constraining, which we achieve by first using the geodesics alone as positional constraints and subsequently carefully adding more constraints to wrap and extend over a larger portion of the surface. Thanks to the DOGs' flexibility and our surface fitting process, we achieve a better coverage compared to torsal patches, as shown in Fig. 10. After wrapping the mesh with DOGs, we non-linearly project the parts of the input mesh onto the DOGs to obtain a valid, manifold, piecewise developable representation (Fig. 1 *result*), which can be fabricated (Fig. 1 *paper craft*).

Contributions. The contribution of this paper is a novel *automatic* tool that approximates general curved surfaces by piecewise developable ones. Since we wrap the input mesh with developable surfaces, rather than deforming it during the wrapping process, our algorithm does not depend on information contained in the specific tessellation of the shape and is thus robust to the underlying meshing, as we show in Fig. 20.

Our specific contributions, illustrated in Fig. 2, are as follows: We contribute a method for fitting DOGs to general surfaces (Fig. 2a). First, while the flexibility of DOGs has been demonstrated in the context of editing systems with few, predefined position constraints, we contribute a robust method to apply DOGs to automatic surface approximation. Secondly, our method allows for granularity control due to the global patch placement optimization (Fig. 2b). Users can specify their desired granularity in terms of number of patches, effectively trading approximation error for ease of assembly. Thanks to our global patch placement being based on developables, our algorithm results in an approximation under the given bounds.

2 RELATED WORK

We give an overview of the representative literature on modeling and approximation with developable surfaces.

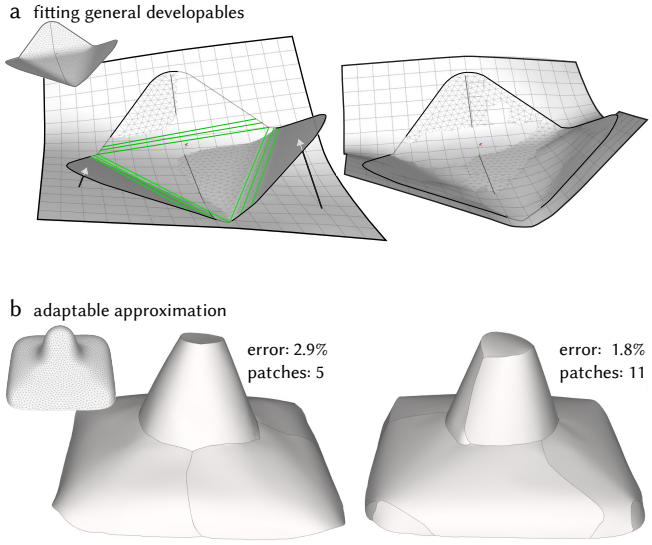

Fig. 2. We contribute an approximation method that (a) uses general developable surfaces, i.e., discrete orthogonal geodesic nets (DOGs). These can model surfaces that torsal patches cannot; in this example we show 3 cylindrically bent parts (signified by the green rulings) connected by a planar part. (b) The approximation power of our method can be steered to achieve variable granularity. Here, the bumpy shape is represented with 5 or 11 patches, consequently leading to a decrease in error, measured as the Hausdorff distance w.r.t. the bounding box diagonal.

Modeling with developable surfaces. The mathematical study of smooth developable surfaces is over two hundred years old [Lawrence 2011]. The past several decades have seen a significant body of work on *freeform modeling* of developable surfaces, a task that requires determining a specific representation for them that is amenable to user control of the shape. There are several such models for developable surfaces, each originating from different equivalent characterizations of developables and then possibly discretized. The representations are based, e.g., on vanishing Gaussian curvature [Wang and Tang 2004], isometry to the plane [Grinspun et al. 2003], or the special parameterizations admitted solely by developable surfaces. The latter parameterizations include conjugate ruled nets [Liu et al. 2006; Bo and Wang 2007; Solomon et al. 2012; Tang et al. 2016; Stein et al. 2018], orthogonal or parallel geodesic nets [Rabinovich et al. 2018a; Wang et al. 2019], or isometric quadrangulation of a planar domain to achieve discrete-isometric mappings [Jiang et al. 2020]. Sellán et al. [2020] focus on modeling developability of height fields and cast it as a convex rank minimization problem.

Different models have different strengths and weaknesses, and for the task at hand we choose different representations for the various stages of our algorithm.

Approximating curved surfaces with developables. The methods in [Wang and Tang 2004; Stein et al. 2018] iteratively deform an input shape until it becomes a discrete developable surface. The instability of the method of [Wang and Tang 2004], which minimizes a vertex-based angle deficit objective (a measure of discrete Gaussian

curvature) is noted in [Stein et al. 2018; Zhao and Xu 2006] and serves as a motivation for the more constrained, ruling-based developable model of [Stein et al. 2018].

In [Stein et al. 2018], the rulings are essentially encoded in the mesh and adapted to reduce the number of developable patches. The geometrical mesh structure emerges in their developable flow process. Other methods that are more concerned with editing rather than approximating with developable surfaces allow ruling directions to vary freely within an a priori defined, global combinatorial mesh decomposition [Solomon et al. 2012; Bo and Wang 2007]. In contrast, Schreck et al. [2015] present an interactive system that recomputes the composition in a pre-defined fixed domain.

Decomposing shapes into torsal developable surfaces, i.e., developable surfaces with non-vanishing mean curvature everywhere (no planar parts), is a popular approach. Mitani and Suzuki [2004] and Liu et al. [2009] use planar quad (PQ) strips, Shatz et al. [2006] fit circular cones, Massarwi et al. [2007] expand to generalized cones. Note that Shatz et al. allow for cone singularities (apexes), as visible in Fig. 3, which creates smoothness artifacts and may hinder some fabrication processes. All these methods perform the decomposition in a process that relies on a prior segmentation and a subsequent parameterization. The work of [Julius et al. 2005] is similar in spirit but decomposes the input into quasi-developable patches as opposed to strips, and these patches are used to make shapes out of paper and fabric.

Similar to our method, the works of [Chen et al. 1999; Pottmann and Wallner 1999] employ developables to wrap an input surface. However, in contrast to our approach, these works do not consider the global nature of the problem and rather use a single ruled patch, essentially relying on the existence of a segmentation of the input mesh if one needs to approximate general geometries and topology.

The global placement of developable patches is nontrivial, therefore a number of previous works rely on user input for guidance. Schüller et al. [2018] decompose shapes into a single spiralling ribbon based on user-guided segmentation. Rose et al. [2007] interpolate arbitrary sketched boundary curves with developable surface strips. Tang et al. [2016] demonstrate a user guided design tool to approximate input surfaces with piecewise spline developables, where users manually determine the initial location of each patch, unlike our work where this process is automated.

Position of this work. We leverage the larger body of work on developable surfaces, as we detail in Section 3, which yields preferable results of our novel approximation algorithm compared to prior methods. Fig. 3 shows that our method produces lower error than the methods of Mitani and Suzuki [2004] and Shatz et al. [2006] *and* fewer patches. In comparison to Stein et al. [2018], our method results in comparable error and *fabricable* parts. Furthermore, our method allows for steering the number of resulting patches. We report the root mean square (RMS) error as the metric that previous works used.

3 METHOD

Our input is a triangle mesh, which we call the *target shape* for the rest of this paper and denote by $\mathcal{S}_t = (V_t, F)$. Our output is a piecewise developable surface that approximates the target and is represented as a triangle mesh with the same connectivity, denoted by $\mathcal{S}_d = (V_d, F)$; we also output its segmentation to developable patches. Each patch can be fabricated by simply bending a thin sheet of material, and the entire shape can be built by joining the patches together (Fig. 1 *paper craft*).

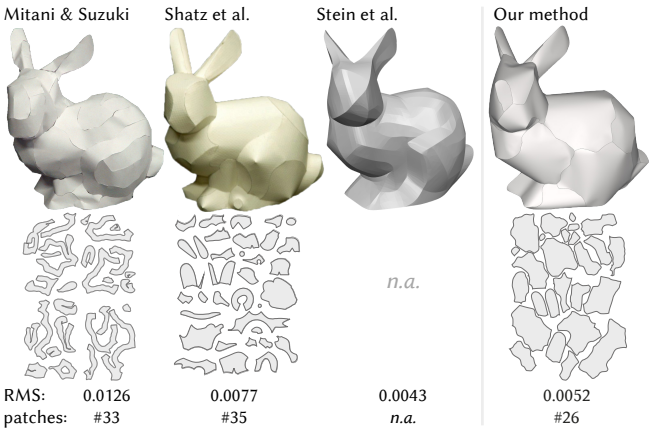

Fig. 3. Compared to previous methods, our algorithm produces comparable or lower error and fewer patches. *Left to right:* Mitani and Suzuki [2004], Shatz et al. [2006], Stein et al. [2018], and our method. The root mean square (RMS) error is computed as cited in the original papers [Cignoni et al. 1998]. We generated Stein's bunny based on their pre-scripted example.

As illustrated in Fig. 4, our algorithm consists of two main steps:

(1) Approximate the target with multiple developable patches.
(2) Non-linearly project the target shape onto the patches.

At the first stage we wrap our target shape with multiple discrete developable patches that approximate different parts of the surface. This results in a set of disconnected, intersecting developable patches (Fig. 4b), which we denote by D_i. These patches alone are not yet useful for fabrication purposes. At the second stage we compute our final output $\mathcal{S}_d$ by a special non-linear projection of $\mathcal{S}_t$ onto the collection of developable patches (Fig. 4c). We describe our wrapping algorithm by first explaining how we fit a single patch to the target surface and then show how to compute the placement of multiple patches to approximate the entire target shape.

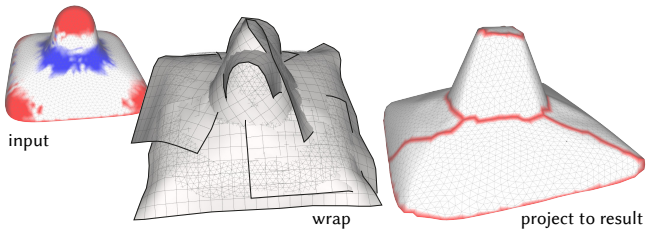

Fig. 4. Our algorithm receives the given target shape to approximate (left). After wrapping it with multiple DOG patches, in this case 5 (middle), we perform a non-linear projection step, which outputs a piecewise developable mesh that approximates the input (right). The colors indicate the Gaussian curvature K, with blue being negative and red positive K.

3.1 Locally fitting a single developable patch

The process of partially wrapping a target surface with a developable surface necessitates a good model for deforming developable surfaces: one that can freely bend in arbitrary directions without being limited by a predetermined set of rulings. Likewise, we would like to use a model that can stretch while staying developable and is not fixed to a predefined isometric planar shape. We use the model of discrete orthogonal geodesic nets (DOGs) [Rabinovich et al. 2018a], as they allow for modeling of developable surfaces without rulings. A DOG is a mesh with regular quadrilateral grid connectivity, where for each vertex of valence greater than 2, all angles in the vertex star are equal.

We wrap the target locally by starting with a flat DOG sheet and minimizing a fitting objective. Fig. 5 demonstrates that this is in itself a nontrivial task. Modeling with DOGs, or any other discrete model of developable surfaces, requires solving optimization problems with many non-linear and non-convex constraints. Constraining the positions of too many vertices of a DOG typically conflicts with its angle constraints and results in an over-constrained problem, as there might not be a discrete developable surface passing through these points. As Fig. 5 shows, over-constraining a DOG typically results in a failure in optimization. We therefore aim to find a stable way to wrap a given target surface.

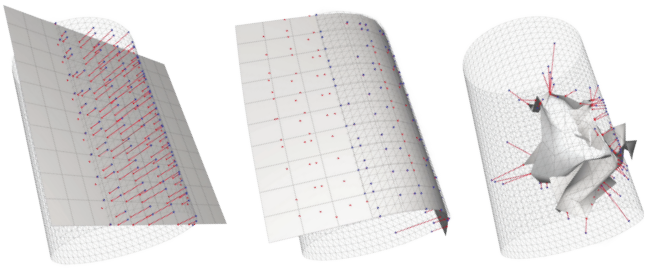

Fig. 5. Optimizing a single DOG while setting positional constraints at all vertices at once is unstable and typically fails, resulting in a mesh that is crumpled and also does not satisfy the DOG constraints. The dots mark the positional constraints' target (blue) and current (red) locations.

3.1.1 Stable patch initialization. Both the theory of smooth developable surfaces and of DOGs shows that one can always constrain a curve on a planar sheet while deforming it isometrically [do Carmo 1976; Rabinovich et al. 2018a,b]. Moreover, if the target curve has non-zero curvature, then this constraint is maximal, in the sense that generally one cannot isometrically deform the sheet and constrain much more than the points on a single curve. Motivated by this, we use the curve-constraining flows of [Rabinovich et al. 2018b] to first locally force a coordinate curve on a DOG patch to lie on some isometric curve along the target mesh. Unlike the case of directly constraining all vertices of a DOG, this can be done in a safe and stable manner. By construction, a coordinate curve of a DOG can be seen as a geodesic on it. Instead of fitting this geodesic to an arbitrary isometric curve on the target, we fit it to a *geodesic* with non-vanishing curvature on the target mesh (Fig. 6).

By matching a geodesic on the DOG to a geodesic on the target we guarantee the following properties:

(1) Along the matched geodesics, the normals of the DOG patch align with the normals of the target surface.
(2) If the target mesh is developable and the chosen geodesic on it has non-vanishing curvature, the DOG wraps the target perfectly.

Fig. 6 provides an illustration. Both of the above properties are mesh-independent. The first property is a consequence of geodesics having non-zero curvature, while the second is shown in [Fuchs and Tabachnikov 1999]. Geodesics with vanishing curvature do not have well defined normals, so these properties do not apply.

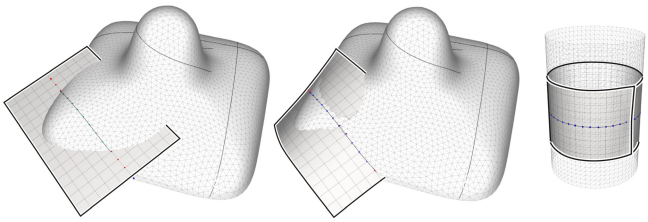

Fig. 6. To avoid over-constraining the initially flat DOG, we fit a geodesic in its center (initially a straight line) to a geodesic on the target mesh. Along the matched geodesics, the normals of the DOG align with the normals of the target surface. If the target is developable and the geodesic has non-vanishing curvature, the DOG patch wraps it perfectly.

3.1.2 Approximating the surface. Fitting a DOG geodesic to a specific target geodesic serves as a stable initialization. On non-developable surfaces, this initialization can often be improved, since the fitted patch still has some degrees of freedom. One of the biggest advantages in using DOGs is the ability to model developable surfaces with planar parts and multiple torsal patches, without being bounded by a pre-determined torsal decomposition [Rabinovich et al. 2018a]. To leverage these advantages, we continue to optimize the DOG patch, using the following strategy.

To approximate the surface, our algorithm adds position constraints in the least squares sense in order to gently pull the initialized patch closer to the target surface. To do so, we start from the well approximated part, i.e., the geodesic on the center line, and iteratively include constraints by growing outwards on the DOG face neighborhood after each optimization step. As Fig. 7 illustrates, this deforms a DOG patch that initially cuts through the target surface to be tangential to it.

The soft positional constraints C are constructed by projecting the target mesh vertices V_t onto the DOG patch. We formulate the soft constraints as a quadratic objective

$$E_{\text{fit}}(\mathcal{S}_t) = \sum_{i \in V_t} \delta_i \left\| C(i) - V_t(i) \right\|^2,$$

where $V_t(i)$ is a vertex on $\mathcal{S}_t$ and $C(i)$ is the closest point on the current DOG patch, expressed as a barycentric combination of DOG vertices. Again, to avoid over-constraining the patch, our algorithm only considers position constraints that are sufficiently close to the

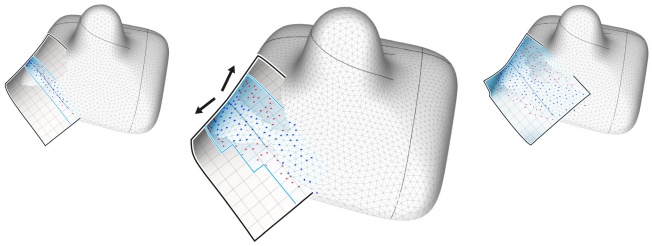

Fig. 7. We optimize a single DOG patch further after the initial geodesic fitting, since it often still has some degrees of freedom. To do so, we iteratively add soft position constraints from the centerline geodesic outwards. This leads to a better target surface approximation compared to the initialization, where the patch was initially cutting the target surface.

current patch shape. The exclusion of outliers is performed by δ_i, which is

$$\delta_i = \begin{cases} 1 & \text{if } \|C(i) - V_t(i)\| \leq \delta_{\text{distance}} \\ & \& \langle N(C(i)),\ N(V_t(i))\rangle \geq \delta_{\text{normal}}\,, \\ 0 & \text{otherwise,} \end{cases}$$

where $N(C(i))$ is the DOG normal at point $C(i)$ and $N(V_t(i))$ is the normal of the target vertex $V_t(i)$. If several target vertices are projected onto the same face of the DOG patch, we pick one that fulfills the above criteria and set the δ_i of the others to zero to avoid conflicting constraints.

3.1.3 Covering a larger surface. The length of our DOG patch is governed by the length of the geodesic, however, its width is not specified. When we construct the patch, we set its width to a default value, which is half the largest dimension of the target shape. However, the DOG patch might be able to cover a larger surface than its initial size if it is allowed to stretch (while remaining a discrete developable), as shown in Fig. 8. To achieve this, we relax the outlier threshold δ_{distance}. Additionally, we apply an energy term that allows the DOG's edge lengths to change while maintaining regularity of the parameterization, as described below.

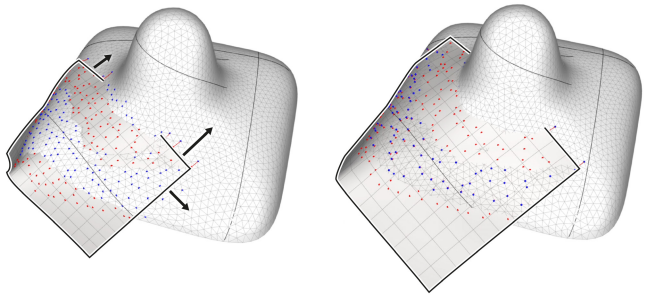

Fig. 8. To allow the patch to cover a larger surface, we relax the outlier threshold δ_{distance}. This leads to a larger set of position constraints in E_{fit}.

3.1.4 Optimization. Throughout the approximation, we optimize the DOG patch using equality-constrained SQP as in [Rabinovich et al. 2019]. We define the objective similar to [Rabinovich et al. 2018b] as a weighted sum of the terms

$$\begin{aligned} \min \quad & w_{\text{H}} E_{\text{H}} + w_{\text{edge}} E_{\text{edge}} + w_{\text{fit}} E_{\text{fit}}\,, \\ \text{s.t.} \quad & \text{DOG constraints,} \end{aligned}$$

where E_{H} is the bending energy to keep the DOG patch smooth and E_{fit} are the soft positional constraints described above. The term E_{edge} regularizes the edge lengths within the DOG patch. In the patch initialization and surface approximation step, we maintain isometric edges using the $E_{\text{edge}} = E_{\text{iso}}$ from [Rabinovich et al. 2018b] (see Eq.(14), page 13). In the stretching step, we exchange the isometry objective with $E_{\text{edge}} = E_{\text{uni-opp}}$, a term that allows the DOG's edge lengths to change, while keeping opposing edges of the same length to control the parametrization regularity [Rabinovich et al. 2018b] (see Eq.(15)). For all examples in this paper, we used fixed weights of $w_{\text{H}} = 6.0$, $w_{\text{edge}} = 1.0$, $w_{\text{fit}} = 1.5$.

3.2 Global approximation with multiple developables

Now that we know how to fit a patch to a given part of the target surface initialized by a geodesic, we need to find out *where* on the surface we should place patches. General shapes most often are only piecewise developable and therefore require multiple patches to be fitted, unless they are already entirely developable. Our goal is to *cover* our target surface $\mathcal{S}_t$ with developable patches D_i. More precisely, we aim to find a set of developable patches D_i such that the Euclidean distance from each vertex of the target $\mathcal{S}_t$ to one of the D_i's is smaller than some approximation threshold $\varepsilon > 0$. Hence, the challenge is two-fold: we need to be able to *locally* wrap an area of the target mesh with a single developable as detailed in Section 3.1, while also *globally* determining the areas each patch should cover.

Motivated by the previous section, we compute an estimate to this problem. We sample a large set of geodesics on the target surface and quickly estimate approximate developable surfaces for them. We use these approximate developables to select a smaller subset that still covers the surface up to the given approximation threshold (Fig. 9). The corresponding geodesics then serve as the initialization for our DOG patch fitting. We detail in the following how we compute this subset efficiently.

3.2.1 Sampling geodesics. We sample a set of geodesics by randomly selecting vertices as their starting points and tracing them on the surface in a direction of our choice. We choose the maximum principal curvature direction [Panozzo et al. 2010], since it leads to geodesics with higher curvature on cylindrical or conical parts compared to the minimum curvature direction. In fact, a geodesic in direction of minimum curvature on developable shapes is the ruling, i.e., has vanishing curvature and is therefore not a good candidate to initialize a DOG patch, as discussed in Section 3.1.1. Starting from a given vertex, we intersect a straight line in the max principle direction with a triangle edge. Similar to the line-of-sight algorithm for geodesics on polyhedral surfaces [Balasubramanian et al. 2009; Polthier and Schmies 2006], we isometrically flatten the neighboring triangle incident on the edge. We continue our line on the neighboring triangle, maintaining the direction. By repeating

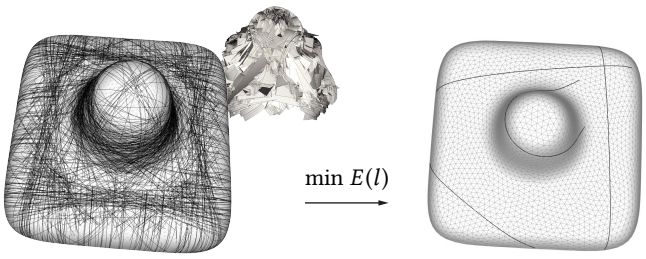

Fig. 9. To find where on the target we should fit DOG patches, we (left) sample a large set of geodesics, for which we (center) procedurally construct ruled developables. (right) We select a subset of these developables to globally cover the target. Their generating geodesics are the initializations for our DOG patches. Here, we randomly sample 20% of the 2477 vertices, and select 4 geodesics as our initializers.

this step, we can create geodesics of arbitrary length, which means that we need to find a meaningful stopping criterion.

Since we only sample the geodesics as stable initializers for the DOG patches, we aim to trace geodesics that would represent *smooth* developable surfaces. As such, our first criterion is to stop tracing a geodesic once its second order normal differences deviate too much. We detect this by an adaptive mean-shift variation, where we keep track of the normal differences of each path segment and evaluate whether a new normal difference exceeds a standard deviation of 3σ, i.e., is an outlier, in which case we stop tracing. This ensures that we do not trace geodesics past creases or other large dihedral angles. Secondly, we also prevent geodesics from winding around smaller features (e.g., the bunny's ears) by stopping when the angle sum of the path is $> 2\pi$. Multiple windings do not offer any additional benefit for our DOG patches and complicate the approximation.

3.2.2 Finding a subset of initial geodesics on the target. As a next step, we choose a smaller set of geodesics to serve as the initializers for our DOG patches D_i. We aim to select geodesics that are well spaced across the target, such that they globally optimize for coverage by DOGs. Since this is merely an initialization, we only need a rough estimate of the DOG patches to assess the coverage quality. Therefore, we efficiently create ruled developable surfaces by passing a spline through each sampled geodesic. Using the analytic derivates of the spline, we compute the ruled rectifying developable that interpolates the spline [Bo and Wang 2007]. We compute the ruling direction as the Darboux vector $r = \tau T + \kappa B$ with T being the unit-length tangent and B the unit binormal of the spline. Since our splines often have inflection points, for which rulings are undefined [do Carmo 1976], we compute the curve of regression [Pottmann and Wallner 2001] and disregard very short rulings, as they indicate flat parts on the curve. The resulting ruled surfaces are a very simple representation and only serve the purpose of helping the selection of geodesics.

To select a subset of geodesics that covers the target, we utilize a multi-label graph cut algorithm [Boykov et al. 2001]. We opt for this global optimization procedure since this method has been demonstrated to be effective for similar problem definitions [Herholz et al. 2015]. Let our randomly selected set of geodesics correspond to the labels l_i, for which we wish to find a minimizer. Our data term $d(i, l_i)$ evaluates the distance from a vertex on the target $i \in V_t$ to the closest point on the ruled surface generated by the geodesic with label l_i. The smoothness term s_e compares neighboring labels and penalizes the assignment of different labels. To do so, the algorithm traverses the edges $\mathcal{E}$ of the target $\mathcal{S}_t$ and compares the labels of the two edge vertices. Consequently, we formulate our energy as

$$E(l) = \sum_{i \in V_t} d(i, l_i) + \sum_{e \in \mathcal{E}} s(e)$$

$$\text{where} \quad s(e) = \begin{cases} 0 & \text{if } l(e_0) = l(e_1) \\ \lambda & \text{otherwise.} \end{cases}$$

3.2.3 Fitting DOG patches. The result of the graph cut algorithm gives us initial locations onto which we can place our DOG patches. We then follow the single DOG patch fitting algorithm as described in Section 3.1, by first fitting geodesics curves (Fig. 6) and then further optimizing and improving our approximation. Fitting DOGs, as they are general developable surfaces, improves the coverage compared to simple torsal patches with the *same geodesics* as generator curves. We show in Fig. 10 how the error is distributed on the surface. We compute the error simply as the distances between the target $\mathcal{S}_t$ and the respective covering developables.

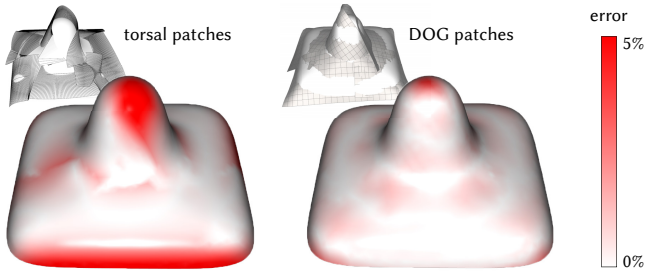

Fig. 10. We confirm that fitting DOGs (right) results in a significantly better coverage than with torsal patches (left). Here, we compare patches using the same geodesics as initial positional constraints for the DOGs and as generator curves for the torsal patches. The maximum distance error is 7.5% with torsal patches and 4.2% with DOGs. The latter decreases to 3.7% as we add another DOG to cover the top (not shown in image).

3.2.4 Cover any remaining holes. After our algorithm computes an approximating DOG for each geodesic, we evaluate the coverage of $\mathcal{S}_t$. For every vertex in V_t, we measure the shortest distance to any patch, resulting in our approximation error ε. We visualize this error in Fig. 10. Our algorithm considers every vertex with an approximation error $\varepsilon_i > \varepsilon_{\max}$ to be uncovered, where $\varepsilon_{\max}$ is a user-defined threshold. We define uncovered areas as connected components of uncovered vertices. As shown in Fig. 11, our algorithm approximates additional patches for uncovered areas using the same steps as before, i.e., it retrieves a geodesic and fits a DOG for each area.

3.3 Non-linear projection of the target onto the patches

Our wrapping algorithm returns a set of intersecting developables that do approximate the target, but also include extraneous parts that need to be culled. Simply cutting these areas might still leave

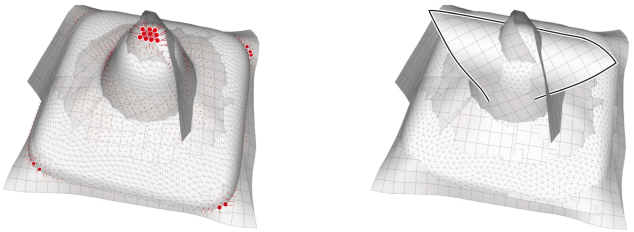

Fig. 11. After fitting multiple geodesics, our algorithm evaluates uncovered areas marked by red circles in this image. Here, the top of the *Bumpy* model remained uncovered. Our algorithm adds a patch to achieve good coverage of the target.

us with small holes, multiple connected components and a resulting mesh with different topology than the target. Instead of introducting and subsequently repairing these topological defects, we take a different approach: We non-linearly project the target mesh onto the wrapping developables. Because we use the original mesh and iteratively project it onto the DOGs, we generate a mesh that remains a valid manifold, and has the same combinatorics and topology as the target mesh, but is close to the developable patches (Fig. 4c). All our results shown in Section 4 are valid manifolds, we didn't encounter degenerate meshes as our output.

As a first step we label each face of the target mesh $\mathcal{S}_t$ with the index of a close-by DOG patch. Such a labeling induces a decomposition of the mesh into connected components associated with DOG patches. Direct projection, i.e. using the patch that is closest to face barycenters, does not yield satifsfactory results as intersections of DOG patches might lead to many isolated components. Therefore we employ a graph cut based approach, similar to the one described in Section 3.2, to obtain a labeling that is both smooth and associated faces to spatially close DOGs. We split the faces of $\mathcal{S}_t$ into connected components forming a disconnected mesh $\mathcal{S}_d$. As some vertices will appear in triangles of different labels, they will be duplicated. The non-linear projection step evolves this initial result $\mathcal{S}_d$ towards a piecewise developable surface. To this end we try to find vertex positions for $\mathcal{S}_d$ such that the following properties hold:

DOG Projection: each vertex should be close to its associated DOG surface.

Developability: the angle defect for each interior vertex should be very small. We use a threshold of $5 \cdot 10^{-3}$ for the maximal angle defect.

Smoothness: the surface should be smooth, exhibiting the least amount of wrinkling possible.

Seam Smoothness: for aesthetic reasons and easier manufacturability we would like seams between developable patches to be as smooth as possible.

Connectedness: corresponding vertices at patch boundaries should be colocated.

Each of these objectives is represented as a weighted term in the compound objective function

$$E(\mathcal{S}_d) = w_{\text{angle}} E_{\text{angle}} + w_{\text{proj}} E^k_{\text{proj}} + w_{\text{L}} E_{\text{L}} + w_{\text{S}} E_{\text{S}} + w_{\text{stitch}} E_{\text{stitch}},$$

where E_{angle} sums the squared angle defects at inner vertices normalized by vertex area and E^k_{proj} sums the squared distances of each vertex to the closest point on the DOG corresponding to its label. Over the course of the optimization we reproject vertices onto the DOGs, which allows them to slide over the patches; however, vertices remain always associated with the same DOG. The term E_{stitch} sums squared distances of corresponding vertices, the pairing of which is determined at the labelling stage. The term E_{L} contains a bi-Laplacian energy, encouraging a piecewise smooth solution. Finally, the energy E_{S} penalizes non-smooth seams. This energy applies 1D smoothing on the seams by employing the combinatorial Laplace operator restricted to the graph of boundary vertices $\mathcal{N}_i$. Two boundary vertices, x_i and x_j, are connected by an edge in this graph if they are connected by an edge in the original mesh $\mathcal{S}_t$. We formulate this energy as

$$E_{\text{S}} = \sum_i \Big\| \sum_{j \in \mathcal{N}_i} (x_j - x_i) \Big\|^2 .$$

Since the objective $E(\mathcal{S}_d)$ can be represented as a non-linear least-squares energy, we employ the Gauss-Newton algorithm to optimize it. To enforce developability, we use the quadratic penalty method [Nocedal and Wright 2006], i.e., we increase the weight w_{angle} over the course of the iterations. It is well known that optimizing for vanishing discrete Gaussian curvature is generally unstable [Stein et al. 2018; Zhao and Xu 2006]. However, we increase the weight w_{angle} after the optimization has already moved the vertices close to the DOGs, such that they describe a surface that is almost piecewise developable.

4 RESULTS

We showcase the applicability of our algorithm on various examples. We cover a representative set of applications, including developable and doubly curved primitives as well as complex shapes. We indicate the developability of our results using the discrete Gaussian curvature K on the *inner vertices* normalized by their vertex area. For each result, we report the mean K_{mean} and the absolute maximum $|K|_{\text{max}}$. We indicate the shape similarity by the Hausdorff distance d_H, with respect to the length of the bounding box diagonal.

Developable shapes. First, we show in Fig. 13 (left) that our algorithm approximates trivial developable shapes, such as a cylinder and cone, smoothly. The cylinder is approximated using a single DOG patch. Although the patch is initially only half the height of the cylinder, the position constraints that allow for stretch enable the approximation with one patch. The cone is represented using two DOG patches, due to the geodesics construction. Yet, as evident in Fig. 13 (right), both patches meet in a smooth manner, resulting in a very good representation.

Doubly curved primitives. Doubly curved surfaces are certainly more challenging and ambiguous to represent by developables. It is well known that no good solution exists to approximating a sphere with developable surfaces. As shown in Fig. 14 (left), the result of our algorithm is akin to a tennis ball. Our algorithm also succeeds at combinations of developable parts and doubly curved parts, as demonstrated in Fig. 14 (right). This example also showcases the

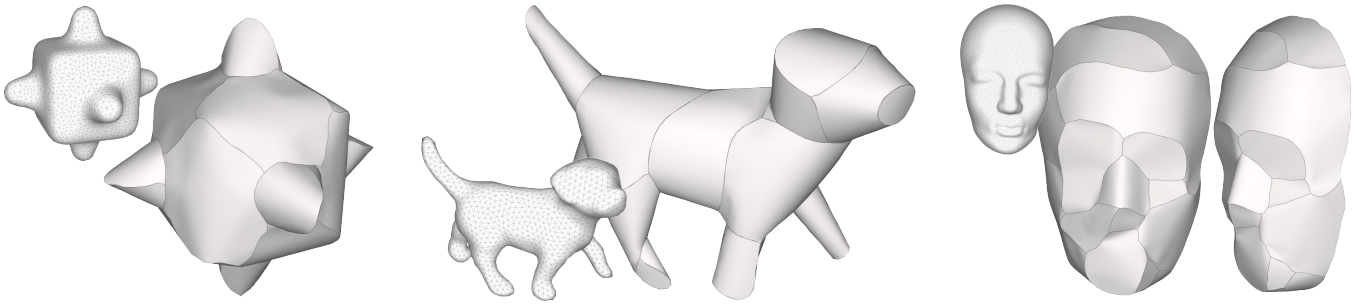

Fig. 12. Approximating complex shapes by piecewise developable surfaces results in a sensible abstraction, as showcased by (left) the bumpy cube, (center) the puppy, and (right) the face. (bumpy cube: 24 patches, $d_H = 1.7\%$, $|K|_{\max} = 2.9 \cdot 10^{-4}$, $K_{\text{mean}} = -1.8 \cdot 10^{-7}$; puppy: 23 patches, $d_H = 3.3\%$, $|K|_{\max} = 2.6 \cdot 10^{-3}$, $K_{\text{mean}} = 3.0 \cdot 10^{-5}$; face: 16 patches, $d_H = 2.2\%$, $|K|_{\max} = 9.6 \cdot 10^{-4}$, $K_{\text{mean}} = 1.2 \cdot 10^{-5}$)

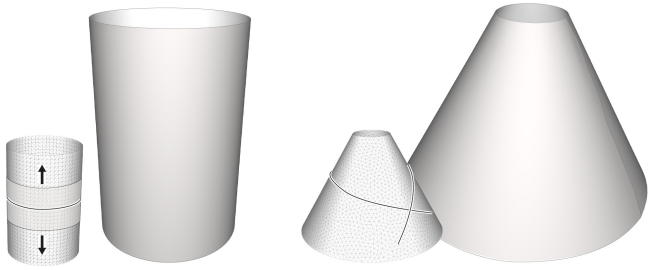

Fig. 13. Our algorithm creates smooth cylinders with one patch thanks to our stretching step in the patch fitting procedure (left). Our smooth cone consists of two patches (right), since we there are no geodesics on the cone that would cover it due to the nature of our geodesic construction. The seams meet smoothly. (cylinder: 1 patch, $d_H = 0.4\%$; cone: 2 patches, $d_H = 1.0\%$)

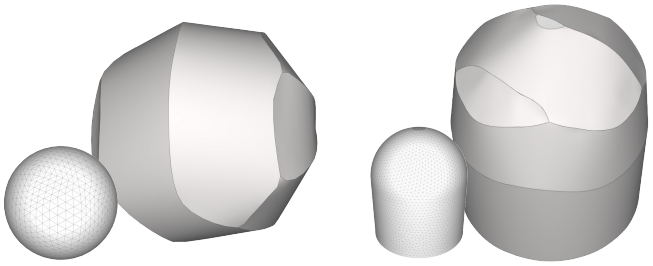

Fig. 14. We can approximate spheres (left) as well as combinations of developable and doubly curved shapes (right), here a cylinder with a hemisphere. The topology of the input, as demonstrated by the hole on top, is preserved by our non-linear projection. (sphere: 9 patches, $d_H = 4.1\%$, $|K|_{\max} = 8.3 \cdot 10^{-4}$, $K_{\text{mean}} = 3.3 \cdot 10^{-6}$; dome: 6 patches, $d_H = 1.8\%$, $|K|_{\max} = 1.8 \cdot 10^{-4}$, $K_{\text{mean}} = -1.3 \cdot 10^{-6}$)

benefit of our non-linear projection of the original mesh, in that the hole at the top is preserved.

Complex shapes. We demonstrate the performance of our algorithm on more complex examples, shown in Fig. 12. We show the result of our algorithm for the entire bumpy cube in Fig. 12 (left). We used one side of it to illustrate our method throughout Section 3.

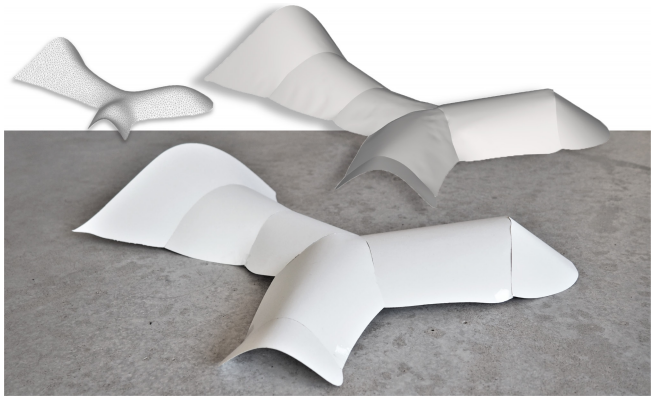

Fig. 15. This architectural shell showcases the use of developable surfaces for architectural design; we fabricated it from paper. The input shape consists of 3551 vertices and results in 7 patches, $d_H = 1.8\%$, $|K|_{\max} = 4.18 \cdot 10^{-3}$, $K_{\text{mean}} = 6.77 \cdot 10^{-6}$.

The bumpy cube is not a trivial example, yet our algorithm produces a good approximation automatically. Similarly, our algorithm approximates the puppy (Fig. 12 center) and face (Fig. 12 right) models automatically while staying close to the target shape. Developable surfaces have applications, e.g. in architecture. We demonstrate an architectural shell example in Fig. 15, which we also fabricated from paper.

Steering the approximation tradeoff. Our algorithm is automatic, yet the tradeoff between approximation error and number of patches can be steered by the user. The goal of any developable approximation highly depends on the task or preferences: while for flank milling a very small approximation error might be prioritized, for large-scale or assembly-intensive tasks, such as fabrication from sheet material (e.g., in architecture or ship building), a smaller number of patches might be desirable. Our algorithm allows steering this tradeoff, mainly governed by the parameter λ, which is the penalty in the smoothness term of our geodesics selection (see Section 3.2.2). Since we globally optimize the patch placement, increasing this penalty leads to fewer patches being selected, yet their distribution is still optimized. We demonstrate the effects of the λ penalty in

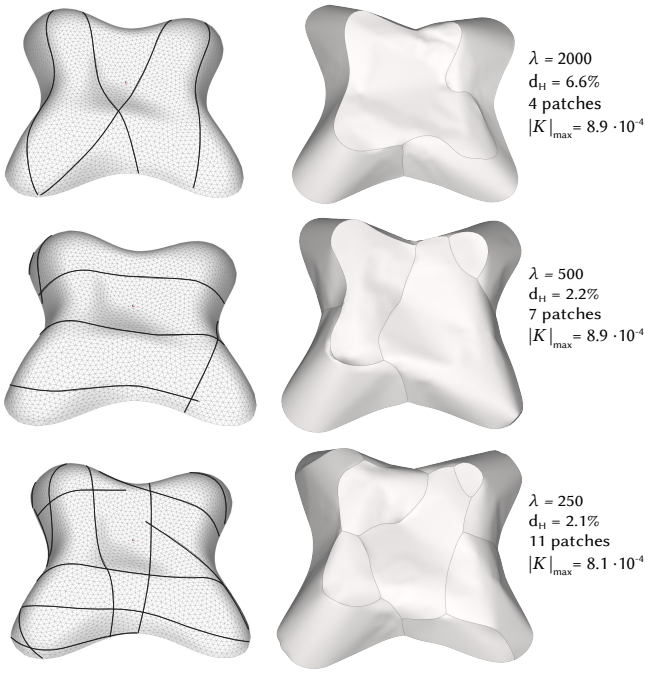

Fig. 16. Our algorithm allows steering the approximation tradeoff via the patch placement optimization. We show how increasing the penalty λ reduces the number of patches at the cost of higher error.

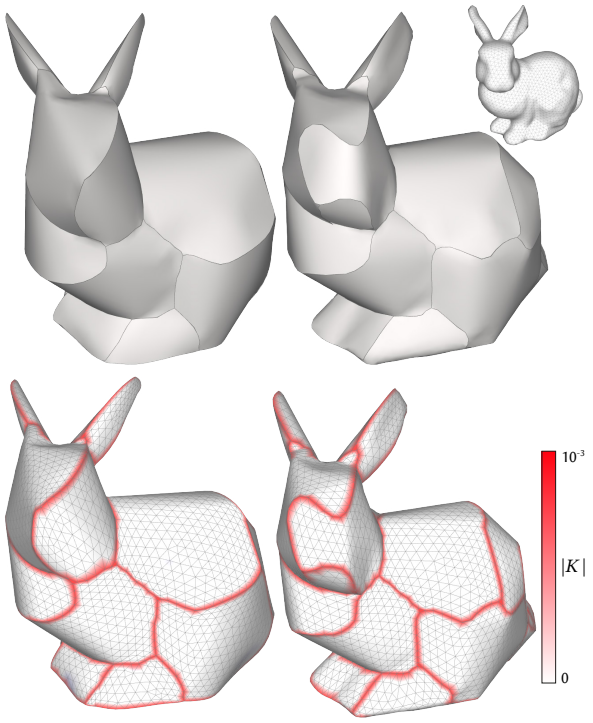

Fig. 17. (Top row) We show the bunny in two different granularities. The left model has 18 patches and $d_H = 4.1\%$ with $|K|_{\max} = 2.5 \cdot 10^{-3}$ and $K_{\text{mean}} = 2.5 \cdot 10^{-5}$. The right result consists of 26 patches and $d_H = 2.8\%$, with $|K|_{\max} = 1.4 \cdot 10^{-3}$, and $K_{\text{mean}} = 7.2 \cdot 10^{-6}$. (Bottom row) The Gaussian curvature K concentrates onto the resulting creases. We show the absolute Gaussian curvature $|K|$ in red.

Fig. 16. This confirms that when opting for fewer patches, our algorithm produces sensible abstractions. Fig. 17 showcases that the granularity adaption performs as expected on the complex models as well, such as the bunny.

In our experiments shown in Fig. 18, we confirm that the random set of geodesics, if sufficiently large, does not affect the geodesic selection substantially. For shapes with a clear best fit, like the cone, the three different random sets lead to similar selected geodesics and developable representation. The Lilium model has no clear best developable fit, which results in different geodesics being selected. Nevertheless, the selected geodesics satisfy our objective of covering the target with the given granularity λ.

4.1 Implementation

We implemented our algorithm in C++ using libigl [Jacobson et al. 2018] for geometry processing capabilities and Pardiso [De Coninck et al. 2016; Verbosio et al. 2017; Kourounis et al. 2018] for solving our linear systems. The results are generated on a 2.5 GHz Intel Core i7-7660U CPU laptop with 16 GB RAM. Computation times for the aforementioned examples lie between 2 and 9 minutes. As this range already implies, the computation speed mainly depends on the chosen granularity, i.e., the number of DOG patches to be fitted. The target mesh resolution, i.e., the number of vertices, does not significantly affect the geodesics and the DOG fitting, but it does impact the non-linear projection step.

Note that the initialization procedure is very efficient and constitutes only a small percentage of the overall timing. Tracing the

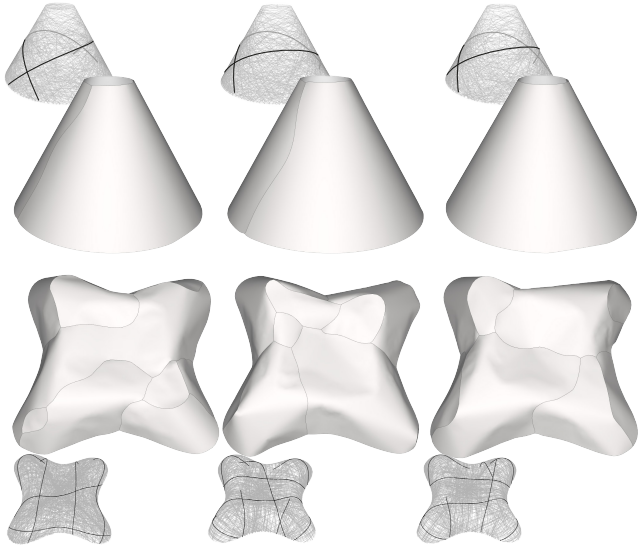

Fig. 18. Each result stems form a different set of randomly sampled geodesics. For both examples, we randomly sampled 20% of the target's vertices, resulting in 518 for the cone and 677 for the Lilium. We show the random geodesics in light gray and the selected geodesics in black. We used $\lambda = 2000$ for the cone (top) and $\lambda = 300$ for the Lilium (bottom).

geodesics and building approximate ruled developable surfaces is fast, since we compute analytic rulings from the splines passing through the geodesics. The computation time for the geodesics selection via the multi-label graph cut algorithm depends on the chosen geodesics sample size. Note that our prototype implementation currently does not utilize multi-threading. However, fitting the DOG patches as well as tracing the geodesics is trivially parallelizable.

4.2 Discussion

Limitations. Our method is designed to approximate arbitrary shapes, but it also exhibits some limitations. DOGs are designed to approximate smooth surfaces and, in our algorithm, sharp creases emerge from intersections between them. Our method is not specifically optimized towards preserving sharp creases, which is evident in the fandisk example in Fig. 19. While sharp creases do emerge, and the cube and fandisk are approximated with little error overall, locally some corners and creases in the fandisk example end up being rounded off. For example, the left edge of the center piece is smoothed because the DOG fitted to both sides of the crease. Since such mechanical shapes are typically easy to segment based on creases, we acknowledge that other approaches that build on prior segmentation (e.g., [Mitani and Suzuki 2004]) likely lead to more feature-preserving solutions. Such segmentation can be integrated with our method at the placement finding stage in the future.

Our method is robust with respect to the tessellation of the target shape, as we show in Fig. 20. We note, however, that we do achieve the most success with close-to-uniform tessellations. Irregular tessellations (Fig. 20a) or noisy surfaces (Fig. 20b) can be approximated using our algorithm. Noisy surfaces lead to a larger number of geodesics, as the stopping condition is based on their curvature. Sparse meshes, if simple enough, can succeed with our algorithm,

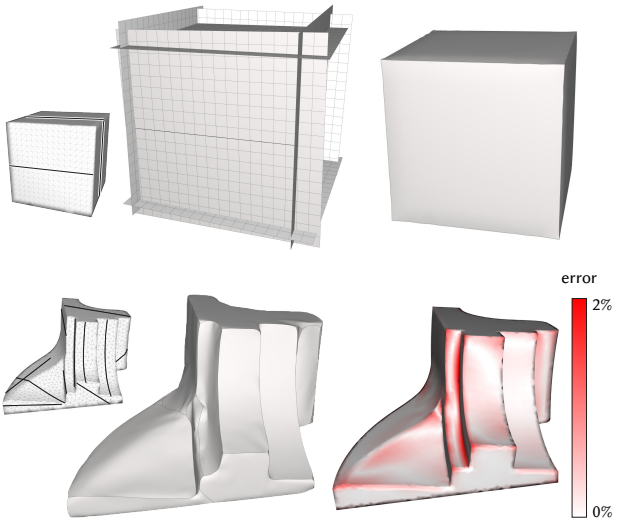

Fig. 19. Our method can approximate models with sharp creases, yet it is not optimized for perserving such features specifically. (Top) the cube is approximated with 6 patches ($d_H = 1.2\%$), and the fandisk (bottom) with 17 patches, ($d_H = 1.7\%$).

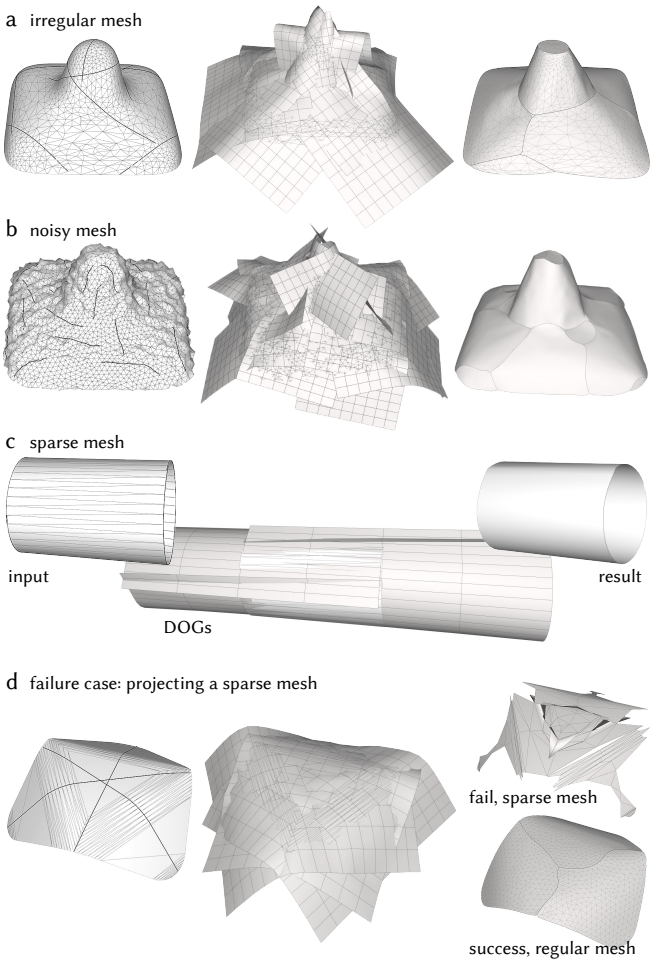

Fig. 20. Our algorithm is not strictly limited to uniform tessellation and can approximate meshes that are valid manifolds with sufficient number of vertices. Our algorithm succeeds with (a) irregularly tessellated meshes, (b) noisy surfaces and (c) on simple shapes, even if very sparsely tessellated. While the wrapping stage is less prone to challenging tessellations, (d) the non-linear projection fails, yet it succeeds given a more uniform mesh.

as shown in Fig. 20c. Our algorithm does find two geodesics on each rim of the cylinder and wraps it with two DOG patches accordingly. In the wrapping stage, our algorithm uses the target mesh vertices selectively as position constraints, with δ_{distance} determining how far the vertices may be from the current patch. Our algorithm initializes δ_{distance} based on the average edge length of the target model and applies a multiplier of 1.25. This default value seems to perform well on simple shapes such as the cylinder, but might fail for more complex shapes. As we show in Fig. 20d (left), a very sparse tessellation can lead to failure in our non-linear projection stage, even after successful wrapping. Using a more uniform tessellation can mitigate this problem, as we preliminarily show in Fig. 20d (right). Such a uniform remeshing step can be easily integrated into our pipeline in the future.

Comparison with Stein et al. [2018] . We already compared our results with previous methods on the bunny example in Fig. 3. In Fig. 21, we extend our comparison with Stein et al. [2018], as it is the most recent representative work in this area. We use their open source code and run several of our models with their method, taking care to select the best possible parameter settings for each example, such as energies and optimizer strategies. Our method generally results in smoother approximations. The authors do acknowledge that their "final design is largely guided by the input tessellation" leaving the user to provide a suitable, curvature aligned mesh. We, on the other hand, use meshes that are common in 3D reconstruction or modeling software. In terms of fabrication, their method lends itself to flank milling, as they acknowledge in their paper. While the orginal work does not provide any patch decomposition, we applied the automatic segmentation and parameterization method [Sorkine et al. 2002] that the authors mention in their paper. We show the obtained patches in Fig. 22. However, since this parameterization method is generic and not specially tailored to piecewise developable surfaces, the resulting patches may run across creases. Our results consist of well-defined flattenable patches that can readily be fabricated from sheet material. One limitation worth noting here is that the boundary of the open shapes is not smooth in *both* methods, pointing to more research questions.

Future user interfaces. In this paper, our focus is on building a stable algorithm with only high-level parameters, resulting in high quality, automatic developable approximations. In the future, our algorithm can be augmented with elaborate user interaction for greater real-world impact. Such interaction design requires analysis of users' requirements, current workflow and context, including how much influence they would like to have in such design tasks. Since our algorithm is automatic, it is very suitable for suggestive user interfaces, effectively making multiple suggestions to users, which they can mix selectively. This can be implemented within our global placement routine, which accepts pre-initializations and fixed selections. Alternatively, we can use the previously fitted DOGs and let them slide towards poorly covered areas. If users wish to define a rough layout of the patches, a possible interaction would be to allow them to paint onto the surface. We could trace the closest geodesic and generate preview patches in realtime. These user-selected patches can be used again as pre-selected geodesics in our global placement optimization, which then covers the remaining areas. Similarly, users may wish to specify different degrees of granularity *locally*. This selection could be integrated into our placement optimization and offered as an interactive brushing interaction for users. Lastly, symmetry is an important visual attribute that contributes to the perception of aesthetics. Our algorithm can be extended by adding symmetry detection and allowing users to select symmetry axes in their design.

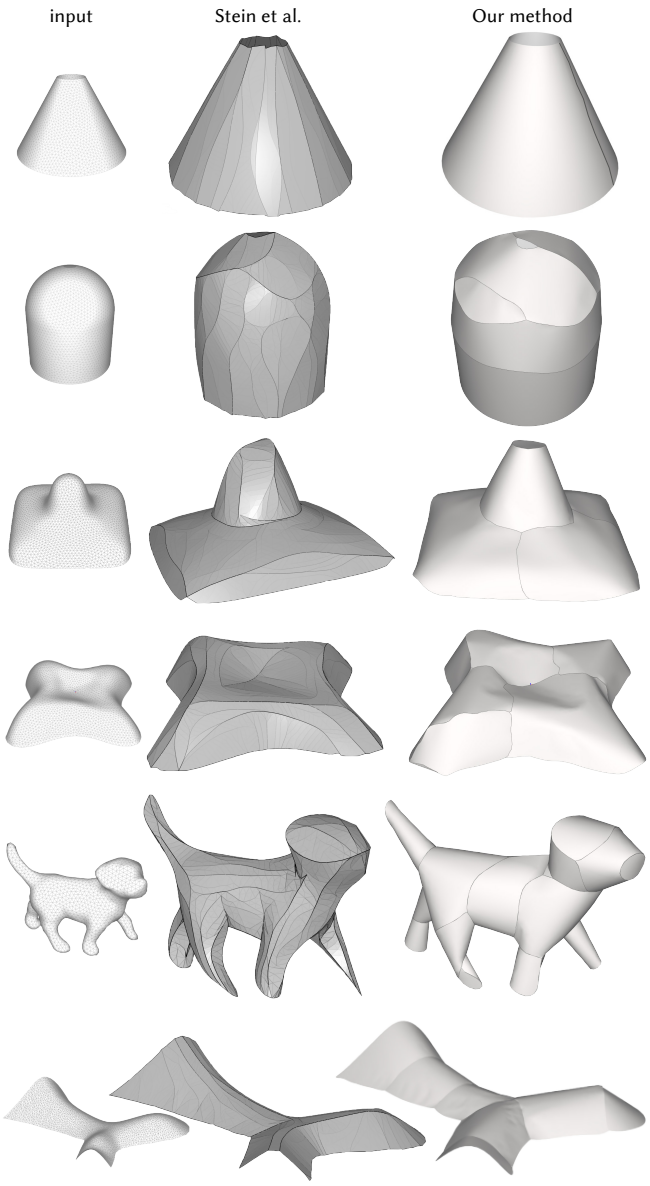

Fig. 21. We compare several of our results to the method of Stein et al. [2018], as the most recent representative of this line of work.

5 CONCLUSION

We presented an automatic algorithm that approximates a given target shape by piecewise developable surfaces. Our method preserves the overall target shape and its topology. The key is that we do not deform the original mesh to compute the approximation, but instead wrap the input with developable patches that we control and optimize. While our method is automatic, we enable users to specify the granularity of the resulting developable representation, effectively steering the tradeoff between assembly effort and approximation error. We demonstrated the effectiveness of our algorithm on several digital and fabricated examples.

In the future, we plan to investigate aesthetic aspects, such as controlling the seam, as well as offering more high-level control for users. We also aim to extend our method to cater to manufacturing needs, e.g., by optimizing developable surfaces with stability, scale and material properties in mind.

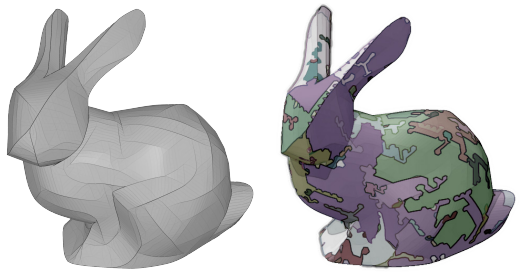

Fig. 22. The method of Stein et al. [2018] does not readily export flattenable patches. We applied the method that they cite as a possible automatic segmentation tool [Sorkine et al. 2002], which produces unfavorable patches. Here, 229 patches are found. We used 1.005 as the distortion threshold and 10^6 for the perimeter/area ratio.

ACKNOWLEDGMENTS

We would like to thank David Lindlbauer for feedback, support and help with paper model fabrication and video production. This work was in part supported by the Swiss National Science Foundation (NCCR Digital Fabrication Agreement #51NF40-182887) and by a DAAD FIT fellowship.

REFERENCES

Q. Alessio. 2012. *Membrane locking in discrete shell theories.* Ph.D. Dissertation. Niedersächsische Staats-und Universitätsbibliothek Göttingen.

M. Balasubramanian, J. R. Polimeni, and E. L. Schwartz. 2009. Exact Geodesics and Shortest Paths on Polyhedral Surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 6 (2009), 1006–1016. https://doi.org/10.1109/TPAMI.2008.213

P. Bo and W. Wang. 2007. Geodesic-Controlled Developable Surfaces for Modeling Paper Bending. *Computer Graphics Forum* 26, 3 (2007), 365–374. https://doi.org/10.1111/j.1467-8659.2007.01059.x

Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23, 11 (Nov 2001), 1222–1239. https://doi.org/10.1109/34.969114

D. Chapelle and K.-J. Bathe. 1998. Fundamental considerations for the finite element analysis of shell structures. *Computers & Structures* 66, 1 (1998), 19–36.

H.-Y. Chen, I.-K. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. 1999. On Surface Approximation Using Developable Surfaces. *Graphical Models and Image Processing* 61, 2 (1999), 110 – 124. https://doi.org/10.1006/gmip.1999.0487

P. Cignoni, C. Rocchini, and R. Scopigno. 1998. Metro: measuring error on simplified surfaces. *Computer graphics forum* 17, 2 (1998), 167–174.

A. De Coninck, B. De Baets, D. Kourounis, F. Verbosio, O. Schenk, S. Maenhout, and J. Fostier. 2016. Needles: Toward Large-Scale Genomic Prediction with Marker-by-Environment Interaction. *Genetics* 203, 1 (2016), 543–555. https://doi.org/10.1534/genetics.115.179887

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

D. Fuchs and S. Tabachnikov. 1999. More on paperfolding. *The American Mathematical Monthly* 106, 1 (1999), 27–35.

E. Grinspun, A. N. Hirani, M. Desbrun, and P. Schröder. 2003. Discrete Shells. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* 62–67. http://dl.acm.org/citation.cfm?id=846276.846284

P. Herholz, W. Matusik, and M. Alexa. 2015. Approximating Free-form Geometry with Height Fields for Manufacturing. *Computer Graphics Forum* 34, 2 (2015), 239–251. https://doi.org/10.1111/cgf.12556

A. Jacobson, D. Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. (2018). https://libigl.github.io/.

C. Jiang, C. Wang, F. Rist, J. Wallner, and H. Pottmann. 2020. Quad-Mesh Based Isometric Mappings and Developable Surfaces. *ACM Transactions on Graphics* 39, 4 (2020). https://doi.org/10.1145/3386569.3392430

D. Julius, V. Kraevoy, and A. Sheffer. 2005. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum* 24, 3 (2005), 581–590.

D. Kourounis, A. Fuchs, and O. Schenk. 2018. Towards the Next Generation of Multiperiod Optimal Power Flow Solvers. *IEEE Transactions on Power Systems* PP, 99 (2018), 1–10. https://doi.org/10.1109/TPWRS.2017.2789187

S. Lawrence. 2011. Developable Surfaces: Their History and Application. *Nexus Network Journal* 13, 3 (2011), 701–714. https://doi.org/10.1007/s00004-011-0087-z

Y. Liu, Y. Lai, and S. Hu. 2009. Stripification of Free-Form Surfaces With Global Error Bounds for Developable Approximation. *IEEE Transactions on Automation Science and Engineering* 6, 4 (2009), 700–709.

Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, and W. Wang. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Transactions on Graphics* 25, 3 (2006).

F. Massarwi, C. Gotsman, and G. Elber. 2007. Papercraft models using generalized cylinders. In *15th Pacific Conference on Computer Graphics and Applications (PG'07).* IEEE, 148–157.

J. Mitani and H. Suzuki. 2004. Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transactions on Graphics* 23, 3 (2004), 259–263.

J. Nocedal and S. Wright. 2006. *Numerical optimization.* Springer Science & Business Media.

D. Panozzo, E. Puppo, and L. Rocca. 2010. Efficient multi-scale curvature and crease estimation. *Proceedings of Computer Graphics, Computer Vision and Mathematics (Brno, Czech Rapubic* 1, 6 (2010).

K. Polthier and M. Schmies. 2006. Straightest geodesics on polyhedral surfaces. In *ACM SIGGRAPH 2006 Courses.* 30–38.

H. Pottmann and J. Wallner. 1999. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design* 16, 6 (1999), 539 – 556. https://doi.org/10.1016/S0167-8396(99)00012-6

H. Pottmann and J. Wallner. 2001. *Computational Line Geometry.* Springer-Verlag, Berlin, Heidelberg.

M. Rabinovich, T. Hoffmann, and O. Sorkine-Hornung. 2018a. Discrete Geodesic Nets for Modeling Developable Surfaces. *ACM Transactions on Graphics* 37, 2 (2018).

M. Rabinovich, T. Hoffmann, and O. Sorkine-Hornung. 2018b. The Shape Space of Discrete Orthogonal Geodesic Nets. *ACM Transactions on Graphics* 37, 6 (2018).

M. Rabinovich, T. Hoffmann, and O. Sorkine-Hornung. 2019. Modeling Curved Folding with Freeform Deformations. *ACM Transactions on Graphics* 38, 6 (2019).

K. Rose, A. Sheffer, J. Wither, M.-P. Cani, and B. Thibert. 2007. Developable Surfaces from Arbitrary Sketched Boundaries. In *Proc. Symposium on Geometry Processing.* 163–172. http://dl.acm.org/citation.cfm?id=1281991.1282014

C. Schreck, D. Rohmer, S. Hahmann, M.-P. Cani, S. Jin, C. C. Wang, and J.-F. Bloch. 2015. Nonsmooth developable geometry for interactively animating paper crumpling. *ACM Transactions on Graphics (TOG)* 35, 1 (2015), 1–18.

C. Schüller, R. Poranne, and O. Sorkine-Hornung. 2018. Shape representation by zippables. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 78.

S. Sellán, N. Aigerman, and A. Jacobson. 2020. Developability of Heightfields via Rank Minimization. *ACM Transactions on Graphics* 39, 4 (2020). https://doi.org/10.1145/3386569.3392419

I. Shatz, A. Tal, and G. Leifman. 2006. Paper Craft Models from Meshes. *Visual Computer* 22, 9 (Sept. 2006), 825–834. https://doi.org/10.1007/s00371-006-0067-6

J. Solomon, E. Vouga, M. Wardetzky, and E. Grinspun. 2012. Flexible developable surfaces. *Computer Graphics Forum* 31, 5 (2012), 1567–1576.

O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. 2002. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization.* IEEE Computer Society, 355–362.

O. Stein, E. Grinspun, and K. Crane. 2018. Developability of triangle meshes. *ACM Transactions on Graphics* 37, 4 (2018).

C. Tang, P. Bo, J. Wallner, and H. Pottmann. 2016. Interactive design of developable surfaces. *ACM Transactions on Graphics* 35, 2, Article 12 (2016), 12 pages.

F. Verbosio, A. D. Coninck, D. Kourounis, and O. Schenk. 2017. Enhancing the scalability of selected inversion factorization algorithms in genomic prediction. *Journal of Computational Science* 22, Supplement C (2017), 99 – 108. https://doi.org/10.1016/j.jocs.2017.08.013

C. C. Wang and K. Tang. 2004. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer* 20, 8-9 (2004), 521–539.

H. Wang, D. Pellis, F. Rist, H. Pottmann, and C. Müller. 2019. Discrete Geodesic Parallel Coordinates. *ACM Transactions on Graphics* 38, 6, Article 173 (Nov. 2019), 13 pages. https://doi.org/10.1145/3355089.3356541

H. Zhao and G. Xu. 2006. Triangular surface mesh fairing via Gaussian curvature flow. *J. Comput. Appl. Math.* 195, 1-2 (2006), 300–311.