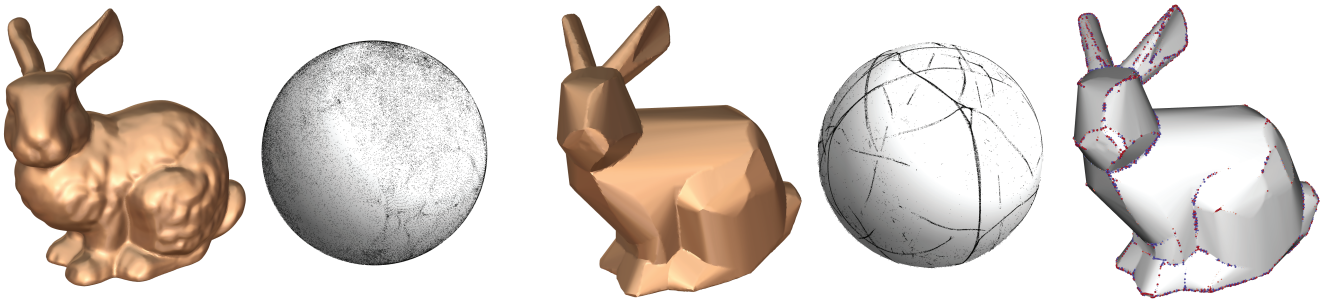
DOI: 10.1111/cgf.14374
Eurographics Symposium on Geometry Processing 2021
K. Crane and J. Digne
(Guest Editors)

*Volume 40 (2021), Number 5*

# Developable Approximation via Gauss Image Thinning

Alexandre Binninger* Floor Verhoeven* Philipp Herholz Olga Sorkine-Hornung

ETH Zurich, Switzerland



Input mesh and its Gauss image | Piecewise developable mesh with thinned Gauss image | Gauss curvature

**Figure 1:** *We present a method for approximating an input mesh with a piecewise developable surface by thinning its Gauss image. Using an iterative algorithm we are able to concentrate Gauss curvature on creases that naturally emerge over the course of the iterations.*

**Abstract**

*Approximating 3D shapes with piecewise developable surfaces is an active research topic, driven by the benefits of developable geometry in fabrication. Piecewise developable surfaces are characterized by having a Gauss image that is a 1D object – a collection of curves on the Gauss sphere. We present a method for developable approximation that makes use of this classic definition from differential geometry. Our algorithm is an iterative process that alternates between thinning the Gauss image of the surface and deforming the surface itself to make its normals comply with the Gauss image. The simple, local-global structure of our algorithm makes it easy to implement and optimize. We validate our method on developable shapes with added noise and demonstrate its effectiveness on a variety of non-developable inputs. Compared to the state of the art, our method is more general, tessellation independent, and preserves the input mesh connectivity.*

**CCS Concepts**

• ***Computing methodologies*** → ***Shape modeling;*** *Mesh geometry models;*

## 1. Introduction

Developable surfaces are an important class of shapes in geometric modeling, as they can be manufactured by pure bending of sheet materials, without stretching or shearing. As such, developable surfaces are often used in architectural design, where surfaces are composed of panels of stiff materials, as well as in industrial design of products that can be manufactured with a cylindrical CNC milling tool. Designing developable surfaces can be challenging due to the highly constrained nature of the shape space, and in practice, typically only relatively simplistic smooth developable surfaces are used, or piecewise developable surfaces consisting of a low number of pieces. A recent surge in research focusing on approximating general 3D shapes with piecewise developable surfaces highlights the desire to design developable surfaces without needing to worry about their constrained nature during the design phase. We discuss past and current research in this area in Sec. 2.

In this paper, we propose a general method for piecewise developable approximation that is based on a well known characterization: the Gauss image (i.e., the set of normals) of a piecewise developable surface is one-dimensional: it may consist of several spherical, not necessarily simple curves and isolated points. Finding a piecewise developable approximation of a shape hence amounts to finding a surface whose geometry is close to the input and whose Gauss

*joint first authors

image is "thin". Our approach does not rely on a segmentation of the given shape into pieces that should be individually approximated by developables, but rather optimizes the developability of the shape as a whole. Our method alternates between a local step that optimizes the surface normals to form a curve network-like Gauss image, and a global step that deforms the surface to approach the target normals. We preserve the connectivity of the input mesh, thereby allowing to carry over attributes of the input if desired, such as spatially varying material properties or texture maps. Our algorithm does not explicitly rely on principal curvature information in its modeling of developability, and the results are tessellation independent to a significant degree.

We show the effectiveness of our algorithm in Sec. 4, where we apply our method to various non-developable surfaces. We compare our results with the state-of-the-art, and we also validate that our method succeeds to recover developable surfaces from ground-truth developables with added noise. The algorithm is concise and simple to implement; we will release our implementation publicly to foster further research and practice in this area.

## 2. Related work

There is a substantial and diverse body of literature on developable surfaces in mathematics and geometric modeling. We touch on the most relevant related works here and refer the interested reader to the detailed surveys in [Rab20, SAJ20, IRHS20].

### 2.1. Representation and modeling of developable surfaces

Freeform modeling of developable surfaces is a task that received a lot of research attention over the last decades. The main goal is to find a suitable representation of developables that incorporates their constrained nature in a straightforward manner, while also providing flexibility for the design process. Attention has been devoted to *smooth* representations of developables, e.g., via B-spline surfaces [TBWP16, GSP19] or via parametric curves with associated envelopes of rectifying planes [BW07] or tangent planes [BR93, PW99] – as well as *discrete* representations, which can afford greater flexibility. Prominent discretization examples include strips of planar quadrilaterals (PQ strips) [LPW*06, VVHSH21], ruled meshes [RSW*07, SVWG12], discrete orthogonal or parallel geodesic nets [RHSH18, WPR*19], as well as discrete isometries of planar checkerboard patterns [JWR*20].

### 2.2. Developable approximation

The stream of works above enables modeling developable surfaces from scratch in a flexible and interactive manner. However, in some scenarios it is desirable to approximate an already existing, but not necessarily developable shape with developable pieces, e.g., for the purpose of rationalization for fabrication. The approximation approaches can be categorized into methods that "wrap", or fit smooth or discretized developable models to the input geometry, and methods that modify or deform the input geometry to optimize a developability criterion. The former approach guarantees that the output is (piecewise) developable in the sense of the used developable representation, but it typically requires segmenting the input surface into patches that are already close to being developable. The latter approach enables a global optimization of the entire surface, where the seams between developable patches and cone apexes may emerge naturally instead of by pre-segmentation heuristic. Our method belongs to this category. The optimization may not reach a global optimum, or the proximity to the input may be traded for developability in some cases, or for the amount of seams, which is also the case with the "wrapping" approaches.

**Approximation by developable wrapping.** Liu et al. [LLH09] approximate rectangular freeform parametric surfaces with a collection of developable strips fitted to strips on the input surface that are obtained via slicing by geodesics. This approach is similar to an earlier method by [Hos98], where the input is restricted to surfaces of revolution and the pre-segmentation into strips is done along the direction that follows the rotation axis. Triangle and PQ strips as discrete developable surfaces are employed in [MS04, SPSH18]; explicit fitting of cones is used in [STL06, MGE07]. Generating a developable strip that approximates the input can be done by fitting a curve (i.e., a 1D object) in the projective space of tangent planes [Pet04]. Peternell [Pet04] uses the moving least squares thinning procedure by Lee [Lee00] on the points in this space, and then reconstructs the developable patch by computing the analytical rulings from the curve, as in [BR93]. The method is applicable only to dense scans of developable surfaces comprised of a single torsal patch, as the fitted curve must be a single connected component without bifurcations.

More recently the developable approximation of more general surfaces, and with a wider variety of developable pieces has received attention. Ion et al. [IRHS20] approximate general input meshes by wrapping them with developable patches, represented as discrete orthogonal geodesic nets (DOGs) [RHSH18]. They initialize the DOGs by selecting a sparse set of geodesic curves and aligning the DOGs coordinate curves to them. After wrapping with disparate developable patches, they employ a graph-cut algorithm to assign each input mesh vertex to a specific patch, followed by a nonlinear projection onto the patch collection to yield a piecewise developable approximation. The method is not greatly sensitive to the input tessellation and generates developable pieces of general geometry as opposed to single torsal patches, but the technique is quite complex, consisting of multiple stages and relying on segmentation and clustering heuristics.

**Approximation by optimizing developability.** Different developability criteria can be optimized to make the input surface more developable. Wang and Tang [WT04] propose to minimize Gauss curvature (angle deficit), which works well when the input surface is already almost developable, but suffers from instability on general inputs [SGC18, IRHS20]. A number of works use a common property of cones, cylinders and planar patches, i.e., developables with constant slope: their surface normals have a constant angle with a certain axis vector [JKS05, DJW*06, JHR*15, GSP19]. This criterion excludes the more general tangent developable surfaces, but they can be locally approximated with constant slope developables to second order [GSP19]. Julius et al. [JKS05] use this criterion for segmentation into near-developable patches that can be flattened with low distortion, whereas Decaudin et al. [DJW*06] and Jung

et al. [JHR*15] improve the developability of a garment piece by employing this criterion in optimization. They first locally fit a cone or a plane to each surface point and then deform the mesh to match the normals found in the first step by solving a Poisson equation, similarly to our approach. However, in their context of garment modeling, the input surfaces are a priori close to developable, and the boundaries and seams are predefined. In contrast, our inputs are general and we do not assume a segmentation or cuts, which necessitates a more elaborate, iterative optimization to enable the seams to emerge automatically and to reach a high degree of developability.

Stein et al. [SGC18] introduce a novel triangle mesh developability criterion: every vertex should have an in- and outgoing edge that together divide the vertex star into two sets of triangles, such that the triangles in each set share a normal (or the entire star is planar). The divider edges are colinear and model the local ruling. This developability criterion is optimized via a surface flow, where the rulings and seams arise naturally, yielding piecewise developable surfaces with highly smooth parts in between the sharp seams. The limitation of this method is the dependence on the input meshing, which should provide the topological ability to align the edges to eventual rulings and hence biases the outcome.

When the input is restricted to height fields, the developability can be characterized by the height field Hessian being rank deficient. Sellán et al. [SAJ20] use this criterion to design a developability optimization. Their method is independent of input tessellation and resolution and also allows for the interpolation of a set of height constraints with a developable height field.

Gavriil et al. [GSP19] increase the developability of freeform architectural surfaces by nonlinear optimization of a similar criterion to [JKS05, DJW*06, JHR*15], namely by adapting the Gauss image to be locally near-planar. In contrast to our method, the method in [GSP19] is restricted to B-spline surfaces as input, which do not contain the levels of noise, sharp features and general topology, as in our inputs. Additionally, they exclude surfaces with wrinkles or folds and as such also surfaces that are piecewise developable, focusing on single smooth developable patches.

## 3. Method

A smooth developable surface is a ruled surface with the surface normal being constant along each ruling. Consequently, such a surface is characterized by the Gauss image being *one-dimensional*: it is a network of curves on the Gauss sphere, where the "junction points" correspond to planar parts on the surface and the curves correspond to so-called torsal patches (developable patches with non-vanishing mean curvature). Piecewise developable surfaces are shapes that consist of smooth developable patches connected to each other via crease curves. Piecewise developables, as well as developables containing creases or curved folds, also have 1D Gauss images that consist of multiple connected components and may include isolated points corresponding to planar patches delineated by crease curves (see Fig. 1). If a surface has a 1D Gauss image, it is piecewise developable.

Fitting a piecewise developable surface to a given shape can be formulated as finding a sufficiently close surface whose Gauss image is one-dimensional. A desirable fit approximates the input

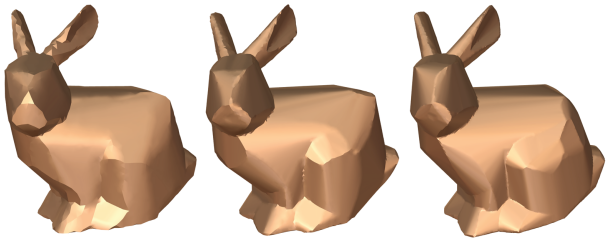



**Figure 2:** *Our method generates comparable results regardless of input resolution. Higher resolutions lead to smoother surfaces and crisper seams.*

shape not just in terms of Euclidean distance but also in terms of higher order properties, such as normals. Therefore it is reasonable to ask that the 1D Gauss image of the approximating developable is close to the original (two-dimensional) Gauss image in some sense. It is tempting to focus solely on "thinning" the input Gauss image, but this poses two challenges: first, a surface in 3D cannot be reconstructed from a set of normals alone, and second, Gauss images can have very complex structures, such that finding a suitable, globally approximating 1D structure is not easily defined.

Our proposal is to gradually deform the input surface towards a piecewise developable shape by guiding the deformation through *local* thinning of the Gauss image. We consider local neighborhoods on the Gauss image and approximate each neighborhood by an arc, moving the center point of the neighborhood onto the arc. This idea is reminiscent of explicit surface smoothing, where points are iteratively moved to a locally optimal location: even though these operations are very local, they reliably lead to a globally smoother shape. Similarly, we observe a globally smooth and one-dimensional structure emerging from our local operations.

To evolve the input surface according to the progressively updated normals, we follow the local-global approach, similar to the as-rigid-as-possible surface modeling method [SA07]. We alternate between the following steps:

1. Locally optimize the Gauss image to become more curve network-like.
2. Globally deform the surface to comply with the optimized Gauss image.

In the local step (Gauss image thinning) we locally approximate the Gauss image by arcs. In the global step (surface deformation), we deform the shape from its current state to one that strives to match the surface normals to the optimized normals found in the previous local step. In the following sections we give details on these two components of our algorithm.

In this paper we work with discrete surfaces, specifically with triangle meshes. The input mesh is denoted by $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ with vertex coordinates $\mathbf{V}$, and our output $\mathcal{M}'$ has the same connectivity and new vertex positions $\mathbf{V}'$. For simplicity, we center the input mesh $\mathcal{M}$ and normalize it to fit inside the unit-diameter sphere.

### 3.1. Gauss image thinning

The local step seeks to find a contracted version of the Gauss image that locally drives the Gauss image to be closer to a 1D object. We initialize the Gauss image using the face normals of $\mathcal{M}$, computed as the cross product of triangle edges.

For each face $f \in \mathcal{F}$ with normal $\mathbf{n}_f$, we compute an updated normal $\mathbf{n}'_f$ as follows. Denoting $\mathbf{b}_f$ as the barycenter of face $f$, we employ a breadth-first search to collect a connected neighborhood of faces $\mathcal{N}_f$ such that

$$\forall g \in \mathcal{N}_f, \quad \|\mathbf{b}_f - \mathbf{b}_g\| \leq r \;\text{ and }\; \mathbf{n}_f \cdot \mathbf{n}_g \geq \cos\omega. \tag{1}$$

The combined criterion above defines a neighborhood of faces that is small both on the surface as well as on the Gauss image: its radius on the mesh is bounded by $r$ and on the Gauss sphere by $\omega$. Notice that $r$ is a bound on the Euclidean distance in $\mathbb{R}^3$, while $\omega$ bounds the geodesic distance $\sphericalangle(\mathbf{n}_f, \mathbf{n}_g)$ on the Gauss sphere. See Fig. 5 for an illustration of the selected neighborhoods.

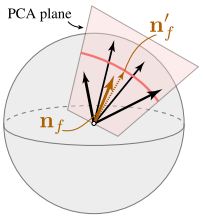
The normals indexed by the set $\mathcal{N}_f$ are used to estimate a best-fit plane whose intersection with the Gauss sphere defines a 1D approximation of the normal set. To this end we perform weighted principal component analysis (PCA) on the Gauss image of $\mathcal{N}_f$. Specifically, we assemble the normal vectors of faces $g \in \mathcal{N}_f$ into a matrix $\mathbf{X}_f \in \mathbb{R}^{3\times|\mathcal{N}_f|}$. In order to find a robust fit, we weight the selected normals based on their geodesic distance on the Gauss sphere to the normal $\mathbf{n}_f$:

$$w_g = \exp\left[-\left(\frac{\sphericalangle(\mathbf{n}_f, \mathbf{n}_g)}{\omega\sigma}\right)^2\right], \tag{2}$$

where $\sigma$ controls how fast the weights fall off with distance. Throughout our experiments we use $\sigma = 2$. The plane with the smallest weighted distance in the least squares sense to the given normal vectors is computed by considering the singular value decomposition of the weighted scatter matrix

$$\mathbf{A}_f = \sum_{g\in\mathcal{N}_f} w_g \mathbf{n}_g \mathbf{n}_g^\mathsf{T} = \mathbf{X}_f \mathbf{W} \mathbf{X}_f^\mathsf{T},$$

where $\mathbf{W}$ is a diagonal matrix of the weights $w_g$. The first two right singular vectors span the plane and define an arc on the Gauss sphere (see inset). We project $\mathbf{n}_f$ onto that arc by projecting it onto the plane and renormalizing, leading to the updated normal $\mathbf{n}'_f$.



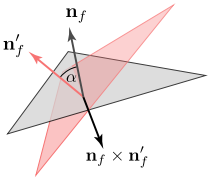

The angle threshold $\omega$ controls the normal cone of the neighborhood $\mathcal{N}_f$, such that our neighborhood selection process is similar to bilateral filtering. The size of the normal cone determines how aggressively we average normals and therfore decides about how large the developable pieces approximating the surface become. At the same time, input meshes might exhibit noise or very small features spanning a large normal cone. For this reason we progressively tighten the normal cone. Starting with a cone angle $\omega_{\text{start}} = 25°$, we reduce it by a constant factor at each iteration until we reach the angle $\omega_{\text{min}}$, which is a parameter of our method. The cone angle for the $k$-th iteration is computed according to the formula

$$\omega = \max\{\omega_{\text{start}} \cdot 0.95^k,\ \omega_{\text{min}}\}. \tag{3}$$

Increasing either, $\omega_{\text{start}}$ or $\omega_{\text{min}}$, or choosing a larger falloff factor leads to smoother results consisting of a smaller number of developable pieces, however, we found it sufficient to only modify $\omega_{\text{min}}$ in order to control the characteristics of the result (see Sec. 4.1).

### 3.2. Surface deformation

The local Gauss image optimization step provides us with updated face normals $\mathbf{n}'_f$. The global surface deformation step finds vertex positions $\mathbf{V}' = \{\mathbf{v}'_1, \ldots, \mathbf{v}'_{|\mathcal{V}|}\}$ such that the mesh faces approximately comply with these normals. To this end, we employ Poisson mesh deformation [YZX*04, BS08], similarly to the global step of ARAP [SA07, LZX*08]. Specifically, we rotate each current mesh triangle $f \in \mathcal{F}$ separately by a rotation $\mathbf{R}_f$ such that its normal becomes $\mathbf{n}'_f$, and we solve a Poisson system to stitch the disparate triangles together. We pick $\mathbf{R}_f$ as the rotation about the axis $\mathbf{n}_f \times \mathbf{n}'_f$ by the angle $\alpha = \sphericalangle(\mathbf{n}_f, \mathbf{n}'_f)$, i.e., the shortest-path rotation.




We opt for rotating individual triangles, as opposed to overlapping vertex neighborhoods as in [SCOL*04, SA07], in order to afford the formation of sharp creases. While generally, non-overlapping neighborhoods may lead to less smooth deformations [JBK*12], in our case the effect is mediated by having smooth prescribed normals (except across creases).

The stitching deformation objective of the global step is therefore

$$E_{\text{deform}}(\mathbf{V}') = \frac{1}{2}\sum_{f\in\mathcal{F}}\sum_{(i,j)\in f} \hat{w}_{ij}\|(\mathbf{v}'_i - \mathbf{v}'_j) - \mathbf{R}_f(\mathbf{v}_i - \mathbf{v}_j)\|^2. \tag{4}$$

We sum over the edges of each face $f = \{(i_1, j_1), (i_2, j_2), (i_3, j_3)\}$, with properly oriented edges. The weight $\hat{w}_{ij}$ is the cotan weight of the oriented edge $(i, j)$ with respect to the triangle $f$ containing it: $\hat{w}_{ij} = \frac{1}{2}\cot\varphi_k$, where $\varphi_k$ is the angle lying opposite of edge $(i, j)$ in the triangle. The objective $E_{\text{deform}}(\mathbf{V}')$ is quadratic in $\mathbf{V}'$.

In order to regularize our developability deformation process, we add a quadratic data term

$$E_{\text{pos}}(\mathbf{V}') = \frac{1}{2}\|\mathbf{M}(\mathbf{V}' - \mathbf{V})\|_F^2 \tag{5}$$

and a quadratic fairness term

$$E_{\text{fair}} = \frac{1}{2}\|\mathbf{L}\mathbf{V}'\|_F^2, \tag{6}$$

where $\mathbf{M}$ is the barycentric mass matrix and $\mathbf{L}$ is the cotan Laplacian matrix. The total deformation objective therefore becomes

$$E(\mathbf{V}') = E_{\text{deform}}(\mathbf{V}') + \lambda_{\text{pos}} E_{\text{pos}}(\mathbf{V}') + \lambda_{\text{fair}} E_{\text{fair}}(\mathbf{V}'), \tag{7}$$

which can be minimized by solving the sparse linear system

$$(\mathbf{L} + \lambda_{\text{pos}}\mathbf{M} + \lambda_{\text{fair}}\mathbf{L}^\mathsf{T}\mathbf{L})\mathbf{V}' = \mathbf{B} + \lambda_{\text{pos}}\mathbf{M}\mathbf{V}, \tag{8}$$

where $\mathbf{B}$ contains expressions in $\mathbf{R}_f$'s and $\mathbf{V}$ stemming from the constant term in the gradient of $E_{\text{deform}}$ and $E_{\text{pos}}$.

We use the cotan Laplacian $\mathbf{L}$ and the mass matrix $\mathbf{M}$ of the initial mesh, which enables us to pre-factor the system matrix. In principle

it is possible to recompute the matrix at every iteration, however, we have found no significant influence of recomputation on the results of the deformation procedure, and maintaining the reference matrix fixed may in fact stabilize the iterative deformation flow [KSBC12]. We find the parameters $\lambda_{\text{pos}}$ and $\lambda_{\text{fair}}$ to work consistently well over a large range of values without the need to pick specific values to guarantee a satisfying result. However, these parameters can be used to tune the result in the obvious way – making it smoother or keeping it closer to the original geometry (see Sec. 4).

Our local-global algorithm is summarized in Algorithm 1. We provide the code in the supplemental material.

**Algorithm 1:** Developability-increasing flow

Center $\mathcal{M}$ at origin and normalize to unit bounding sphere
Initialize $\mathbf{V}' \leftarrow$ input mesh vertex positions
$k \leftarrow 0$
**repeat**
  $\mathbf{V} \leftarrow \mathbf{V}'$
  $\mathbf{N} \leftarrow \text{ComputeFaceNormals}(\mathbf{V}, \mathcal{F})$
  **for** $f \in \mathcal{F}$ **do**
    $\mathcal{N}_f \leftarrow \text{Neighborhood}(f, \mathbf{V}, \mathbf{N}, r, \omega) \cup \{f\}$
    $\mathbf{n}'_f \leftarrow \text{ProjectNormal}(\mathbf{n}_f, \mathbf{N}, \mathcal{N}_f, \omega)$
    $\mathbf{R}_f \leftarrow \text{ShortestPathRotation}(\mathbf{n}_f, \mathbf{n}'_f)$
  **end**
  $\mathbf{V}' \leftarrow \text{DeformSurface}(\{\mathbf{R}_1, \ldots, \mathbf{R}_{|\mathcal{F}|}\}, \mathbf{V})$
  $k \leftarrow k+1$
**until** $k =$ *maxIterations* ***or*** $\max_{i \in \mathcal{V}} \|\mathbf{v}'_i - \mathbf{v}_i\| < 10^{-3}$;
Output $\mathcal{M}' = (\mathbf{V}', \mathcal{F})$

## 4. Results

We implement our algorithm using libigl [JP*21] and run it on a variety of inputs on a standard laptop set up with an Intel i7-1165G7 processor. We test various input geometries, ranging from almost developable to doubly-curved, see Fig. 13. The developability of the results can be estimated by their Gauss image and curvature, while the approximation quality is specified by the Hausdorff distance to the input mesh. We investigate the impact of the algorithm parameters, namely the thresholds used in the neighborhood collection, and we also explore the dependence of the results on the input meshing. In most cases our algorithm performs reliably for standard parameters: 100 iterations, $\omega_{\text{start}} = 25°$, $\omega_{\text{min}} = 2.5°$, $\lambda_{\text{pos}} = 10^{-3}$, $\lambda_{\text{fair}} = 10^{-5}$, $r = 0.1$, and we keep them mostly fixed throughout for our experiments. However, in order to achieve specific effects, such as higher abstraction or fidelity to the input surface, these parameters can be varied.

### 4.1. Algorithm parameters

The problem of computing developable approximations is characterized by several tradeoffs: approximation tightness vs. number of creases and developability. The parameters in our method can be used to steer these tradeoffs, with the minimum cone angle $\omega_{\text{min}}$ being the central parameter.

The parameter $\omega_{\text{min}}$ controls the selection of neighboring points on the Gauss map. A larger value leads to larger neighborhoods with greater normal variance and hence stronger averaging, resulting in fewer developable patches, as shown in Fig. 4. As the value of $\omega_{\text{min}}$ increases, the result become more abstract, featuring larger developable parts . Since normals are very susceptible to noise, we have to make sure the parameter $\omega_{\text{start}}$ is large enough for noisy meshes (e.g., Fig. 8) to have a stronger initial smoothing effect. If this parameter is too small, we risk single triangles to be excluded from local smoothing. In principle, the cone angle could be adapted by an arbitrary function, however, it should be monotonically decreasing, and we found the simple formula in Eq. (3) sufficient for all our examples. The the value of the radius $r$ can be safely chosen smaller than 0.1 for meshes of higher resolution (more than 50k vertices). For significantly lower resolutions, one might need to increase $r$ to ensure that a sufficient number of normals is collected for each $\mathcal{N}_f$ for a stable PCA fit.

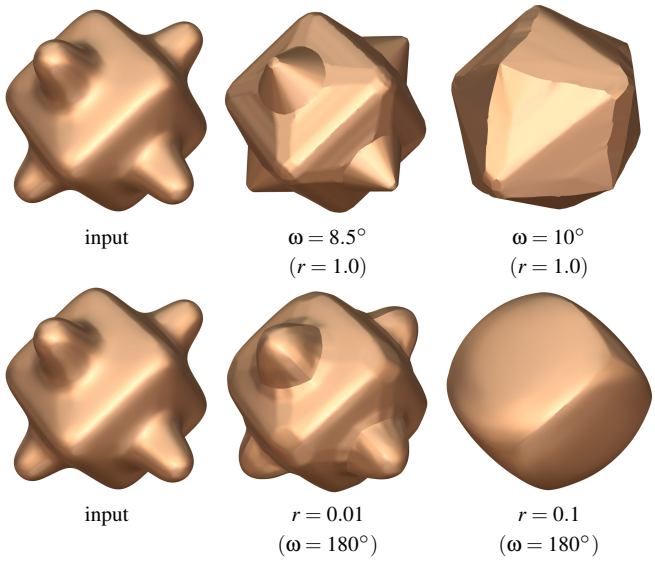



**Figure 3:** *Our two neighborhood parameters ω and r are both needed to prevent oversmoothing and to enable seams to arise.*

To investigate parameter influence, we run an ablation study on the neighborhood Euclidean radius $r$ and Gauss sphere radius $\omega$. We vary one of the parameters and set the other parameter high enough so that its effect on the neighborhood selection is eliminated. Specifically, we set $r = 1$ to eliminate the effect of $r$, since the mesh is scaled to a sphere of unit diameter, and we set $\omega = 180°$ to eliminate selection based on angle between normals. To ease interpretation, we do not apply progressive decreasing of $\omega$ in this experiment, i.e., we do not use Eq. (3). As demonstrated in Fig. 3, when the neighborhood is determined by the angle between normals $\omega$ alone, a lower $\omega$ does not produce developable patches separated by well-defined seams, while a larger $\omega$ struggles to recover the underlying geometry. When the neighborhood size is steered by $r$ alone, a large radius produces too coarse an approximation, while a smaller $r$ sticks more closely to the original geometry with a large amount of developable pieces. This experiment stresses the significance of combining both criteria in neighborhood computation: the

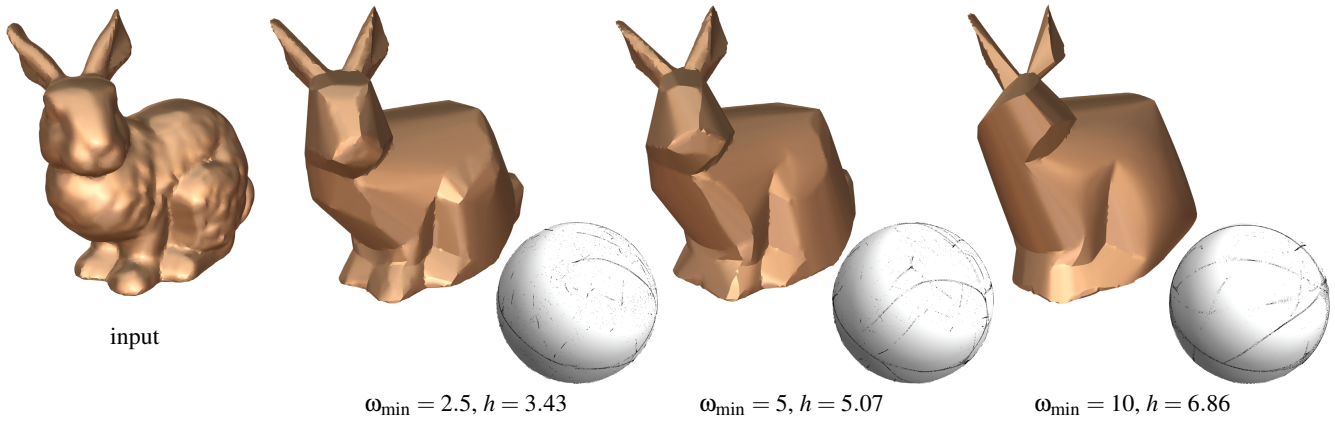



**Figure 4:** *Increasing the parameter $\omega_{\min}$ results in a coarser segmentation into developable pieces and a larger Hausdorff distance (reported in percentage of the bounding box diagonal).*

radius in the spatial domain and on the Gauss sphere, in the spirit of bilateral filtering.

### 4.2. Validation and comparisons

The Gauss images of our results indicate that our method succeeds to deform the input surfaces such that the Gauss image of an initially non-developable input transforms to one that is more akin to a collection of curves corresponding to a piecewise developable shape. Illustrations of these original input Gauss images and our output can be seen in Figures 1, 12, and 13. On surfaces that are already very close to developable, such as the *Fandisk* in Fig. 13, we manage to thin the curves on the Gauss map even further. The rightmost column in Fig. 13 highlights that Gauss curvature is pushed towards the creases between the developable patches and vanishes on the patches themselves.

Note that when small groups of points appear on the Gauss image which are separate from the curve network, this is due to non-developable creases. This is in agreement with the goal of piecewise developable approximation: creating a surface where Gauss curvature is concentrated on the creases between the developable pieces that do have vanishing Gauss curvature.

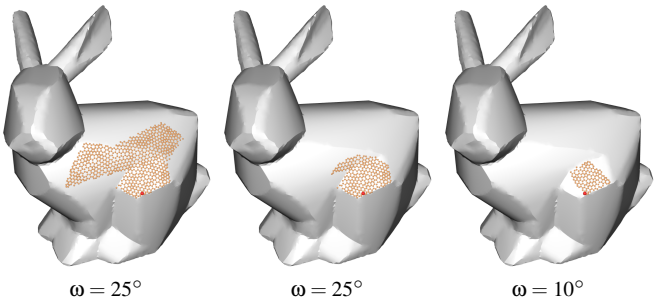



**Figure 5:** *Our collection method selects neighboring faces (orange) that are close to the reference face (red) in terms of Euclidean and connectivity distance r without crossing sharp features, as ensured by a threshold $\omega$ on the angles between corresponding normals.*

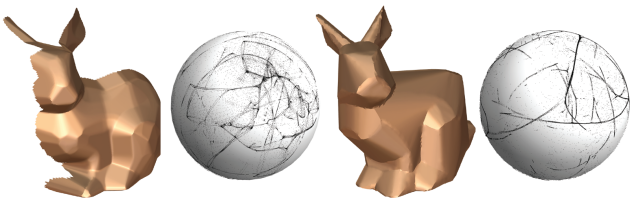



**Figure 6:** *The method of Sellán et al. [SAJ20] works on height fields. We compare a result of their method with ours at a similar resolution.*

In Figures 12 and 6 we show a comparison with the most recent works on developable surface approximation: Stein et al. [SGC18], Ion et al. [IRHS20] and Sellán et al. [SAJ20] (the latter works only on height fields). Our results are sometimes smoother and tend to reflect the symmetries of the input geometry better than Ion et al. [IRHS20], because we entirely avoid their segmentation step, which may introduce random symmetry breaking. Additionally, similar to Stein et al. [SGC18], our method is able to produce a network of seams and open seam curves, unlike only closed seams that delineate disk topology patches as in [IRHS20]. The sharp features in our output may not be as crisp as those of the previous works because we keep the original connectivity. Still, these features are apparent enough, and clearly expressed by Gauss curvature, such that feature-enhancing remeshing should be possible. Compared to [SGC18] our method usually produces fewer seams and avoids flattening the surface between rulings, because the mesh edges do not need to align to rulings in the characterization of developability we choose to work with.

We show in Fig. 7 how our method lowers the overall distribution of discrete Gauss curvature on the mesh. We also show the corresponding histograms for the related works by Stein et al. [SGC18] and Sellán et al. [SAJ20].

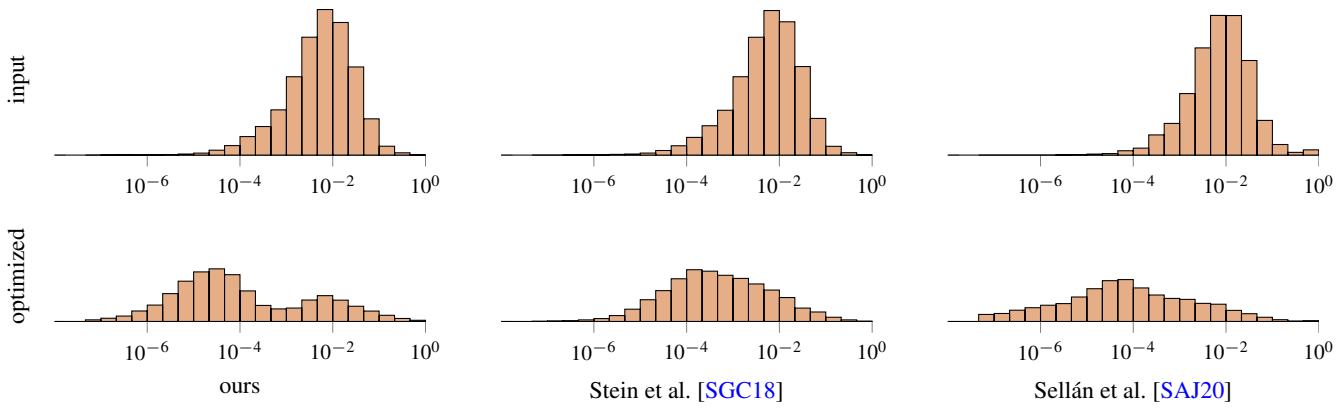



**Figure 7:** *Compared to the results of Stein et al. [SGC18] (middle) and Sellán et al. [SAJ20] (right), our method (left) shifts the discrete Gauss curvature more towards zero. The bimodal distribution is caused by the Gauss curvature of seam versus interior vertices. Discrete Gauss curvature is measured as the angle defect at vertices and plotted on a log-scale. The histograms correspond to the Stanford bunnies in Fig. 6 and Figure 27 of [SGC18] (third row, first column).*

The method of Sellán et al. [SAJ20] produces remarkably smooth surfaces and crease curves. They benefit from working with regular grids stemming from their high resolution height field input; for our method we also observe smoother results on higher-resolution and highly regular meshes, see Fig. 6. For a fair comparison we choose results from both methods that roughly match in resolution. The benefit of our approach is its generality: we are not restricted to height fields and can process arbitrary, irregular input meshes without having to pre-segment them into height field patches. The Gauss image of our result features a much thinner curve network, however, the result by Sellán et al. [SAJ20] is merely a rendering of a triangulated height field, which might have an influence on their Gauss image.

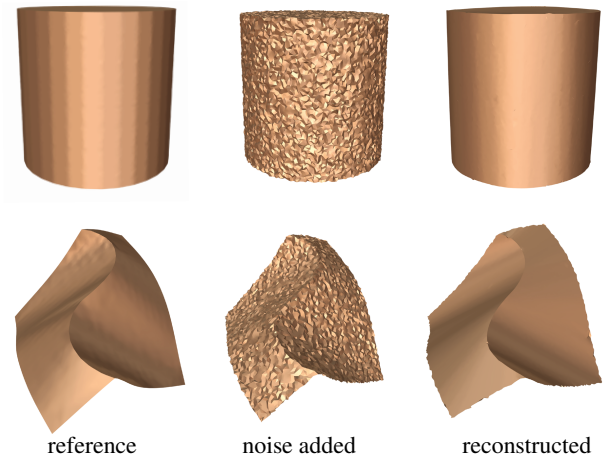



**Figure 8:** *Our method manages to recover underlying analytical developable surfaces when random noise is applied to the input. For these examples we used $\omega_{start} = 45°$. The original curved crease model was provided by Rabinovich et al. [RHSH19].*

### 4.3. Mesh dependence

Our method is based on the averaging of the normals over a collection of neighbors locally computed according to the geodesic distance on the Gauss map and the Euclidean distance of the corresponding faces on the mesh. Combining these two criteria grants a degree of robustness against tessellation dependence and noise. Since the surface update is based on Poisson mesh deformation using local rotations, the deformation is regularized by the input mesh and inherits the relative mesh independence and robustness of gradient domain deformations [BS08]. We validate this statement by comparing our method on meshes featuring noise, different resolutions as well as irregular meshings in terms of triangle sizes and quality.

**Noise.** Normals are highly sensitive to noise, thus a low angle criterion $\omega$ could hinder the computation of the neighborhood. If the amplitude of the noise is small compared to the radius threshold $r$, our method can compute the neighborhood of normals based on the underlying structure of the mesh. The approximation of noisy meshes is illustrated in Fig. 8, where we choose a large starting angle threshold $\omega_{\text{start}} = 45°$ with a decrease factor of 0.9 at each iteration. The meshes represent piecewise developable surfaces and our method manages to restore them even after adding a significant amount of noise.

**Resolution.** Changes in resolution do not greatly affect the distances on the mesh and on the Gauss map. Therefore, the neighborhood collection scales accordingly to the resolution. Our method provides very similar results for inputs of differing resolutions, as shown in Fig. 2.

**Tessellation.** Since our method is not too sensitive with respect to mesh resolution, we can robustly handle varying triangle sizes within the same model (Fig. 9, top row). To test our method on a mesh with anisotropic triangles, we randomly flip one half of all edges. Even

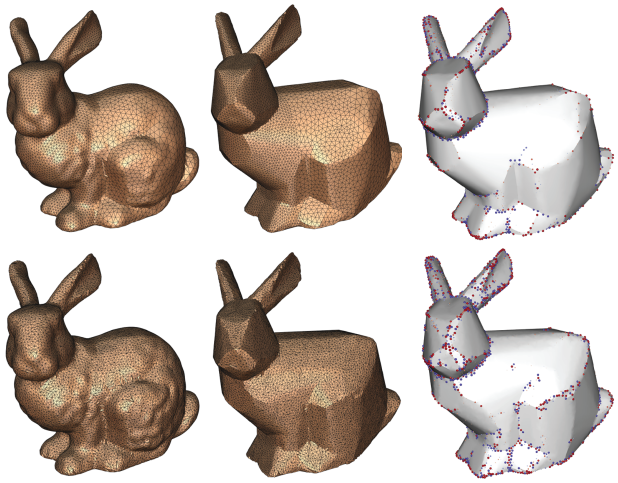


**Figure 9:** *Although our method preserves the connectivity of the original mesh, we obtain good results regardless of the input tessellation. We show our method applied to a meshing of the bunny with varying density (top row) and an irregular tessellation containing skinny triangles (bottom row). The results have very similar shapes and also have comparable Gauss curvature.*

this extreme meshing does not affect our result significantly. We are using the triangle based cotan Laplacian operator for reconstructing the mesh and expect the intrinsic variant [BS07] to lead to even fewer artifacts.

When applying our method to different tessellations of a sphere, it becomes clear that the mesh connectivity affects the final result, see Fig. 10. Nevertheless, all tested tessellations lead to intuitive piecewise developable approximations of a sphere. The sphere is a special case, since every point is umbilical, hence the symmetry of the approximation can only be broken due to meshing. Our method is fairly tessellation independent on meshes with anisotropic curvature, as depicted in Fig. 9.

### 4.4. Runtime performance

To measure the runtime performance of our method, we perform 100 iterations on *Bunny* meshes of different resolutions with our default parameter except for a varying minimum angle $\omega_{\min}$ of $2.5, 5$ and $10$. We report the average runtime per iteration in Table 1, broken down into the neighborhood collection step, the Gauss image thinning (normal update), and the global surface deformation step. The last column (total) also accounts for additional negligible operations, such as the recomputation of the Gauss map. It is evident that the most time is spent on the neighborhood collection, which depends on the angle $\omega$ and the resolution of the mesh, since the mesh is normalized and the selection radius $r_{\max}$ is constant. The thinning of the Gauss image is the second most expensive step, as it also depends on the neighborhood size. Since the local step computes the target normals independently for each face, we provide a performance analysis on top of an implementation relying on parallelization. Further optimization is possible, such as caching of the topological search for the neighborhood collection. Updating the surface is currently negligible in comparison to the local step, since we precompute the factorization of the system matrix and only need to perform right-hand-side assembly and back-substitution in each iteration.

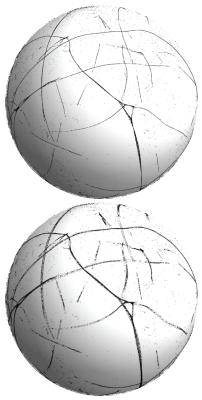
| $\lvert\mathcal{V}\rvert$ | $\omega_{\min}$ | Neighborhood | Normals | Surface | Total |
|---|---|---|---|---|---|
| 2,294 | 2.5 | 0.002 | 0.001 | 0.001 | 0.006 |
| | 5.0 | 0.002 | 0.001 | 0.001 | 0.007 |
| | 10.0 | 0.003 | 0.002 | 0.001 | 0.008 |
| 14,290 | 2.5 | 0.026 | 0.015 | 0.006 | 0.050 |
| | 5.0 | 0.032 | 0.018 | 0.008 | 0.062 |
| | 10.0 | 0.043 | 0.026 | 0.007 | 0.079 |
| 57,154 | 2.5 | 0.495 | 0.193 | 0.055 | 0.757 |
| | 5.0 | 0.613 | 0.253 | 0.055 | 0.937 |
| | 10.0 | 0.960 | 0.419 | 0.056 | 1.451 |

**Table 1:** *Average running time per iteration (in seconds) over 100 iterations on the Bunny mesh of different resolutions. Experiments performed on two AMD Ryzen Threadripper 1950X @ 3.40 GHz, total 32 cores.*

Our runtime performance is comparable to that of Stein et al. [SGC18], who report that their method runs in the range of seconds to a few minutes on inputs of 1k to 57k vertices. Our method scales less well than the method proposed by Sellán et al. [SAJ20], who report linear runtime scaling. Nevertheless, for input resolutions typically used in this paper (roughly 2k-10k vertices) our method is faster, and for resolutions around 57k the runtimes are comparable (estimated 70 seconds for [SAJ20], 75-150 seconds for our method). Our method slightly outperforms the work by Ion et al. [IRHS20], who report a runtime of 2 to 9 minutes on meshes similar to the ones used in this paper.

### 4.5. Target Gauss image

Our algorithm reliably converges towards a very thin Gauss image, however, we still observe points on the Gauss sphere forming lines with a small but visible width. By inspecting the the target normals computed in the 500th iteration of the mesh featured in Fig. 1, we observe that they are actually much closer to a 1D structure (see inset). This effect is due to the reconstruction step which models the deformation from one iteration to the next and therefore constraints the normal fitting. It could be beneficial to consider more general transformations or a global non-linear optimization as a post processsing step in order to generate meshes that are even closer to the goal of having a 1D Gauss image.



### 4.6. Garment design and fabrication

A possible application of our algorithm is the fabrication of shapes from planar materials like sheet metal or textiles. Garment fabri-

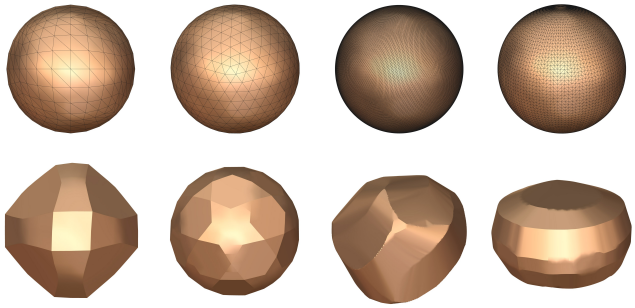


**Figure 10:** *Depending on the tessellation and resolution of the sphere, we obtain different piecewise developable approximations. We use $r = 0.1$ and $\omega_{\min} = 5$ for all four results.*

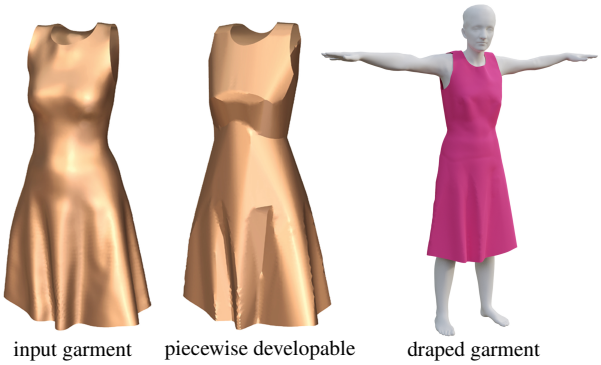



**Figure 11:** *The dress in this example (left) is designed to fit a specific person using [WHZ*21]. We show the result of our method (middle) and the re-simulated draped shape (right).*

cation in particular is largely built around the concept of sewing patterns and creating a garment that drapes on a 3D body shape from planar cloth pieces. Recent novel approaches to garment design propose sketching the garment shape directly in 3D and computing the sewing pattern in a post-process (see [WHZ*21] and the references therein). Improving the developability of the 3D garment model can help produce a more fabrication-ready design for such approaches, see a conceptual example in Fig. 11.

## 5. Discussion, limitations and conclusions

We presented a method to compute a piecewise developable approximation of an input triangle mesh by interleaving local "thinning" of the surface Gauss image and global fitting of the surface geometry to the normals. The key ingredients of our method are robust normal smoothing in the spirit of bilateral filtering, and a surface deformation to match the normals that is regularized by keeping the local deformations rigid and avoiding element collapses. Our algorithm is very simple and depends on a small number of intuitively tunable parameters that control the results. Our method does not require a segmentation of the input into patches (unlike [IRHS20]) – the seams and creases emerge automatically as part of the deformation process. The algorithm performs stably for a range of parameter settings, making a fully automatic mode possible. The implementation

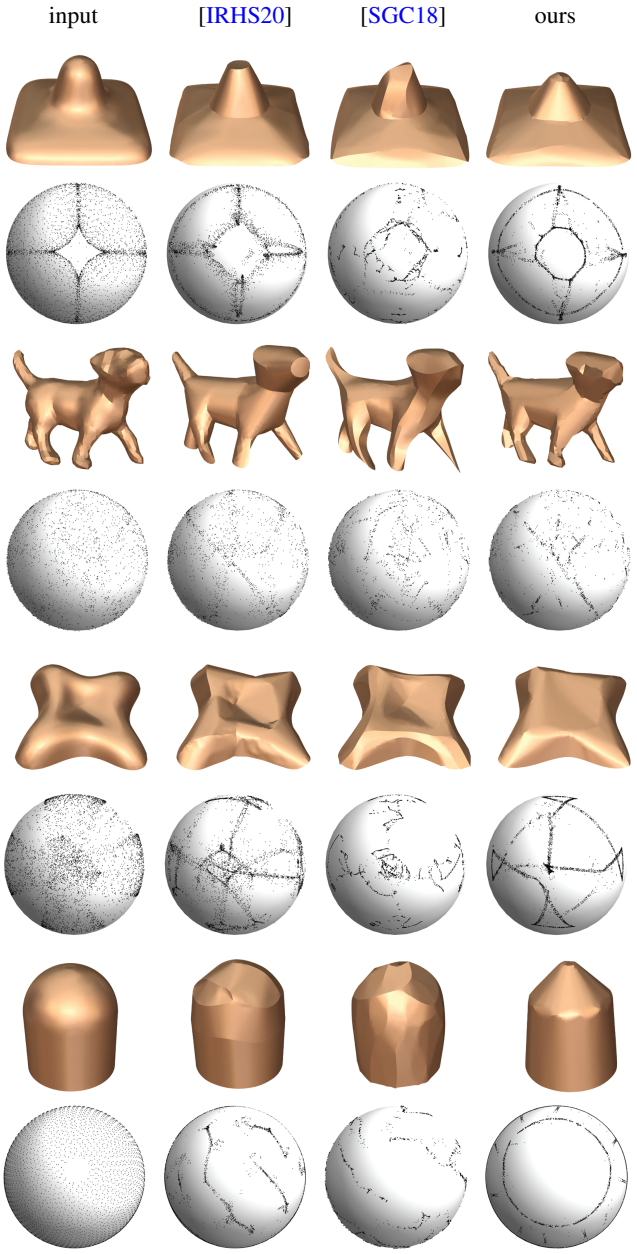



**Figure 12:** *We compare several of our results with the methods of Stein et al. [SGC18] and Ion et al. [IRHS20].*

is straightforward on top of a geometry processing library (we used libigl [JP*21]); we provide our prototype as supplemental material, with only 250 lines of code.

We use the $K = 0$ characterization of smooth developables expressed as 1D Gauss image. This means that our approach does not explicitly depend on rulings and does not rely on the ability of the mesh edges to model them (unlike e.g. [SVWG12, SGC18]), making it relatively tessellation-independent. In our current formulation we preserve the input mesh connectivity as is, and yet creases and seams

are revealed naturally, as expected. The resulting sharp features are sometimes not perfectly crisp due to the fixed connectivity. It would be possible to perform remeshing on the fly (as in e.g. [SRH*15]) to better align the mesh edges to principal directions and sharp features. We have already observed a benefit by performing simple local edge flips, but opted not to include this option to keep the algorithm as simple as possible.

Like other deformation flow based developable approximation methods [SGC18,SAJ20], we do not generate an explicit partitioning of the piecewise developable result into developable patches; the seam lines are not computed directly, and even after ridge detection they might not produce a segmentation of the surface such that each developable patch can be individually flattened, so additional surface cutting may be required [SGC18]. An exciting direction for future work is to incorporate manufacturing considerations into the developability flow, such that usable patches (from the standpoint of fabrication) with smooth seams are encouraged to arise. In this regard, allowing interactive user input to help define the seams would be a helpful feature. With a complete segmentation into developable patches available, it would also be possible to optimize the surface for developability even further using [VVHSH21]. Additionally, it is interesting to explore more elaborate formulations of the local step, which is currently a mere arc fitting. There is a potential tradeoff between a tighter local fitting of the Gauss image and the simplicity and robustness of the overall scheme.

The current running time of our method is dominated by the neighborhood search and PCA computations. Even though we implemented our algorithm in a parallel fashion, it would additionally benefit from spatial indexing and caching. We do observe that the performance scales with mesh resolution because runtime is dominated by neighbourhood queries. The method runs robustly without requiring interactive intervention. Our optimization is a first order, gradient descent based approach and tends to make progress quickly in the beginning and subsequently keep a constant pace; it is conceivable to switch to a second order method after the initial bulk of progress is achieved (e.g., Newton, as proposed in [GSP19]).

**Acknowledgements**

We thank the anonymous reviewers for their remarks. We also thank Amir Vaxman and Tim Hoffmann for the insightful discussions, which laid the foundation for this project. We are grateful to Silvia Sellán for providing us meshes for test and comparison purposes. This work was partially supported by the Personalized Health and Related Technologies (PHRT) SwissHeart grant and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101003104).

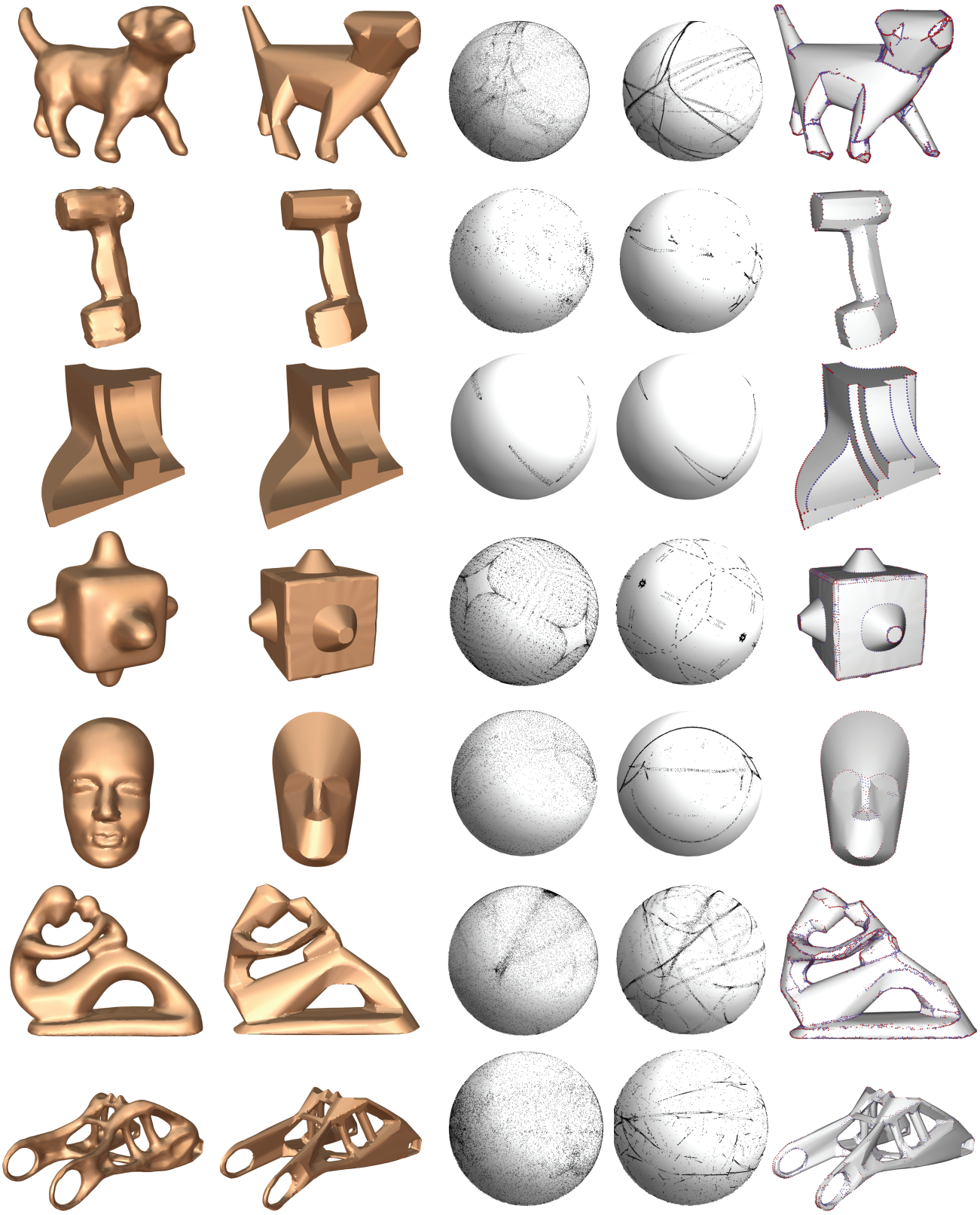


**Figure 13:** *Results computed using our algorithm featuring meshes with different characteristics. The columns show (left to right) the original mesh, the optimized mesh, original and optimized Gauss image as well as Gauss curvature.*

## References


[BR93] BODDULURI R., RAVANI B.: Design of developable surfaces using duality between plane and point geometries. *Computer-aided design 25*, 10 (1993), 621–632. 2

[BS07] BOBENKO A. I., SPRINGBORN B. A.: A discrete laplace—beltrami operator for simplicial surfaces. *Discrete & Computational Geometry 38*, 4 (2007), 740–756. 8

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 213–230. 4, 7

[BW07] BO P., WANG W.: Geodesic-controlled developable surfaces for modeling paper bending. *Computer Graphics Forum 26*, 3 (2007), 365–374. 2

[DJW*06] DECAUDIN P., JULIUS D., WITHER J., BOISSIEUX L., SHEFFER A., CANI M.-P.: Virtual garments: A fully geometric approach for clothing design. *Computer Graphics Forum 25*, 3 (2006), 625–634. 2, 3

[GSP19] GAVRIIL K., SCHIFTNER A., POTTMANN H.: Optimizing b-spline surfaces for developability and paneling architectural freeform surfaces. *Computer-Aided Design 111* (2019), 29–43. 2, 3, 10

[Hos98] HOSCHEK J.: Approximation of surfaces of revolution by developable surfaces. *Computer-Aided Design 30*, 10 (1998), 757–763. 2

[IRHS20] ION A., RABINOVICH M., HERHOLZ P., SORKINE-HORNUNG O.: Shape approximation by developable wrapping. *ACM Trans. Graph. 39*, 6 (2020). 2, 6, 8, 9

[JBK*12] JACOBSON A., BARAN I., KAVAN L., POPOVIĆ J., SORKINE O.: Fast automatic skinning transformations. *ACM Trans. Graph. 31*, 4 (2012), 77:1–77:10. 4

[JHR*15] JUNG A., HAHMANN S., ROHMER D., BEGAULT A., BOISSIEUX L., CANI M.-P.: Sketching folds: Developable surfaces from non-planar silhouettes. *ACM Trans. Graph. 34*, 5 (2015). 2, 3

[JKS05] JULIUS D., KRAEVOY V., SHEFFER A.: D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum 24*, 3 (2005), 581–590. 2, 3

[JP*21] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2021. URL: https://libigl.github.io/. 5, 9

[JWR*20] JIANG C., WANG C., RIST F., WALLNER J., POTTMANN H.: Quad-mesh based isometric mappings and developable surfaces. *ACM Trans. Graph. 39*, 4 (July 2020). 2

[KSBC12] KAZHDAN M., SOLOMON J., BEN-CHEN M.: Can mean-curvature flow be modified to be non-singular? *Computer Graphics Forum 31*, 5 (2012), 1745–1754. 5

[Lee00] LEE I.-K.: Curve reconstruction from unorganized points. *Computer aided geometric design 17*, 2 (2000), 161–177. 2

[LLH09] LIU Y., LAI Y., HU S.: Stripification of free-form surfaces with global error bounds for developable approximation. *IEEE Transactions on Automation Science and Engineering 6*, 4 (2009), 700–709. 2

[LPW*06] LIU Y., POTTMANN H., WALLNER J., YANG Y.-L., WANG W.: Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph. 25*, 3 (2006), 681–689. 2

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. *Computer Graphics Forum 27*, 5 (2008), 1495–1504. 4

[MGE07] MASSARWI F., GOTSMAN C., ELBER G.: Papercraft models using generalized cylinders. In *Proc. Pacific Graphics* (2007), pp. 148–157. 2

[MS04] MITANI J., SUZUKI H.: Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Trans. Graph. 23*, 3 (2004), 259–263. 2

[Pet04] PETERNELL M.: Developable surface fitting to point clouds. *Computer Aided Geometric Design 21*, 8 (2004), 785–803. 2

[PW99] POTTMANN H., WALLNER J.: Approximation algorithms for developable surfaces. *Computer Aided Geometric Design 16*, 6 (1999), 539–556. 2

[Rab20] RABINOVICH M.: *Modeling Developable Surfaces with Discrete Orthogonal Geodesic Nets*. PhD thesis, ETH Zurich, Zurich, 2020. doi:10.3929/ethz-b-000427802. 2

[RHSH18] RABINOVICH M., HOFFMANN T., SORKINE-HORNUNG O.: Discrete geodesic nets for modeling developable surfaces. *ACM Transactions on Graphics 37*, 2 (2018). 2

[RHSH19] RABINOVICH M., HOFFMANN T., SORKINE-HORNUNG O.: Modeling curved folding with freeform deformations. *ACM Trans. Graph. 38*, 6 (2019). 7

[RSW*07] ROSE K., SHEFFER A., WITHER J., CANI M.-P., THIBERT B.: Developable surfaces from arbitrary sketched boundaries. In *Proc. Symposium on Geometry Processing* (2007), p. 163–172. 2

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proc. Symposium on Geometry Processing* (2007), p. 109–116. 3, 4

[SAJ20] SELLÁN S., AIGERMAN N., JACOBSON A.: Developability of heightfields via rank minimization. *ACM Trans. Graph. 39*, 4 (2020). 2, 3, 6, 7, 8, 10

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proc. Symposium on Geometry Processing* (2004), pp. 179–188. 4

[SGC18] STEIN O., GRINSPUN E., CRANE K.: Developability of triangle meshes. *ACM Trans. Graph. 37*, 4 (2018). 2, 3, 6, 7, 8, 9, 10

[SPSH18] SCHÜLLER C., PORANNE R., SORKINE-HORNUNG O.: Shape representation by zippables. *ACM Trans. Graph. 37*, 4 (2018). 2

[SRH*15] SCHRECK C., ROHMER D., HAHMANN S., CANI M.-P., JIN S., WANG C. C., BLOCH J.-F.: Nonsmooth developable geometry for interactively animating paper crumpling. *ACM Trans. Graph. 35*, 1 (2015). 10

[STL06] SHATZ I., TAL A., LEIFMAN G.: Paper craft models from meshes. *Visual Computer 22*, 9 (2006), 825–834. 2

[SVWG12] SOLOMON J., VOUGA E., WARDETZKY M., GRINSPUN E.: Flexible developable surfaces. *Computer Graphics Forum 31*, 5 (2012), 1567–1576. 2, 9

[TBWP16] TANG C., BO P., WALLNER J., POTTMANN H.: Interactive design of developable surfaces. *ACM Trans. Graph. 35*, 2 (2016). 2

[VVHSH21] VERHOEVEN F., VAXMAN A., HOFFMANN T., SORKINE-HORNUNG O.: Dev2PQ: Planar quadrilateral strip remeshing of developable surfaces, 2021. arXiv:2103.00239. 2, 10

[WHZ*21] WOLFF K., HERHOLZ P., ZIEGLER V., LINK F., BRÜGEL N., SORKINE-HORNUNG O.: 3d custom fit garment design with body movement, 2021. arXiv:2102.05462. 9

[WPR*19] WANG H., PELLIS D., RIST F., POTTMANN H., MÜLLER C.: Discrete geodesic parallel coordinates. *ACM Trans. Graph. 38*, 6 (Nov. 2019). 2

[WT04] WANG C. C., TANG K.: Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer 20*, 8 (2004), 521–539. 2

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with Poisson-based gradient field manipulation. *ACM Trans. Graph. 23*, 3 (2004), 644–651. 4