

Robust Curvature-Driven Mesh Optimization

L. Fricke¹ , U. Schwanecke² , M. Botsch^{1,3} 

¹TU Dortmund University, Dortmund, Germany

²RheinMain University of Applied Sciences, Wiesbaden, Germany

³Lamarr Institute for Machine Learning and Artificial Intelligence, Dortmund, Germany

Abstract

Curvature is fundamental to the design and optimization of surfaces. Unlike modeling approaches based on the Laplacian or mean curvature, we achieve more precise control by manipulating surface meshes through principal curvatures. Compared to prior work on curvature-driven surface modeling, our proposed discretization of principal curvatures yields increased accuracy, enhanced robustness, and simplified derivative computations. Furthermore, we extend our curvature optimization to general polygon meshes. Beyond prescribing target curvatures, our framework also enables the modeling and optimization of general curvature functionals on surface meshes, allowing for more advanced mesh curvature processing applications.

CCS Concepts

• *Mathematics of computing* → *Discrete mathematics*; • *Computing methodologies* → *Mesh geometry models*;

1. Introduction

Curvature plays a significant role in geometric processing and surface optimization. Many approaches employ the Laplacian or the mean curvature for mesh processing tasks, such as mesh smoothing, fairing, or modeling and deformation [BKP*10]. In contrast, our focus is on surface modeling driven by the two principal curvatures, a topic previously explored in the work of Eigensatz et al. [ESP08, EP09]. Prescribing principal curvatures opens up a wide range of applications, such as curvature-driven surface editing, smoothing in the curvature domain, feature enhancement, and curvature clamping. In addition, it allows to formulate functionals based on curvature variation, which are especially useful for designing smooth surface blends and hole fills [XZ07].

The method by Eigensatz et al. [ESP08, EP09] – while being flexible and powerful – suffers from a couple of limitations: Due to the complex derivative calculations required for the optimization of the nonlinear curvature energies it requires a significant implementation effort, and since no public implementation is available, it is not as widely used. In addition, their curvature discretization has shortcomings in terms of accuracy for obtuse triangulations, and their numerical optimization can suffer from robustness issues for large mesh deformations.

To address these issues, we improve the principal curvature optimization framework by making it both more robust and more accurate. First, we propose a different curvature discretization than Eigensatz et al. [ESP08]. Instead of using the eigenvalues of the curvature tensor [CSM03] to compute per-vertex principal curvatures, we derive them from discretizations of mean and Gaussian

curvature, where we also use a more robust formulation for computing the involved dihedral angles. Furthermore, we employ mixed Voronoi areas instead of the barycentric areas [MDSB03], which leads to more accurate curvature estimation and less distortion of triangle shapes in the resulting mesh. To improve numerical robustness for more significant deformations, we propose an extended line-search procedure that avoids excessively large step sizes.

We further improve curvature control by including the geodesic curvature of the surface boundary as optimization target in our framework. Moreover, we extend our curvature optimization framework from pure triangle meshes to general polygon meshes. Finally, we demonstrate that our framework is not limited to prescribed principal curvature targets, but can also minimize curvature functionals like total curvature variation.

The source code of our method is publicly available at https://github.com/leofricke/curvature_optimizer.

2. Related Work

We aim for a surface deformation method driven by target principal curvatures. The optimization of surface curvature has a wide variety of applications in surface modeling, surface editing, and smoothing.

Mean Curvature Optimization Some methods manipulate surfaces by prescribing target mean curvatures. The approach by Nealen et al. [NISA07] minimizes the difference between the per-vertex Laplacian and a target Laplacian, which is determined from

the target mean curvature and the vertex normal. Their method then optimizes the vertex positions and normals in an alternating manner. Similarly, Hildebrandt and Polthier [HP04] present a prescribed mean curvature flow that they use to smooth surfaces while controlling surface area and curvature magnitude. In addition, Jin et al. [JZW14] include mean curvature prescription in their deformation framework using a scale-based formulation. Fang et al. [FZL*20] propose a method to modify either the mean or Gaussian curvature while keeping the other fixed. Kineri et al. [KEM14] deform surfaces by prescribing curvature along isoparametric curves. Unlike our approach, none of the above methods provide full control over both principal curvatures.

Metric Preserving Deformation Our principal curvature optimization framework also aims to preserve the surface metric in order to control the deformation and resolve ambiguity in the target curvatures. Optimization of the surface metric is related to various surface parameterization tasks [HPS08]. There exist multiple relevant works [BCGB08, SSP08, SC17] in the field of parameterization that flatten a surface with curvature prescription while preserving conformality and face areas. Although these parameterization methods address curvature and metric energies similarly, they differ from our setting because they do not deform the surface in 3D. Further work addressing more general surface deformation using metric and conformal control is found in [Cor24, JZW14, FZL*20].

Bending Energies A much broader field of research exists in the simulation of thin shells [GHDS03, TPBF87] and cloth simulation [HP04, BMF03, BWH*06, WBH*07]. Various works address the optimization of nonlinear bending energies [TG13, GSH*04, BS99], which often use approximate forms of curvature to model physical bending behavior. While these works do not include actual curvatures, the optimization methods and robustness considerations of bending energies are similar to our method.

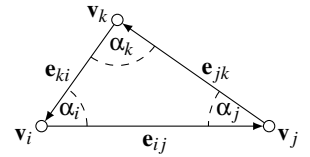
Principal Curvature Optimization The work by Eigensatz et al. [ESP08, EP09] is the first to optimize a surface mesh by prescribing both principal curvatures, while additionally employing position, length, and curvature constraints. Their method has been recently applied to enforce constant Gaussian curvature in patch layouts [KMB20]. However, the principal curvature optimization framework by Eigensatz et al. [ESP08] was never widely adopted since no implementation is publicly available and the derivative calculations needed for the optimization are rather involved. Also, the method may experience stability issues, which are common with nonlinear optimization problems.

Curvature Functionals Curvature-based functionals can be used for surface smoothing and fair surface design, minimizing area, curvature, or curvature variation [Tau95, DMSB99]. One of the most popular energy functionals in this area is the Willmore energy [BS05], which is often used in smoothing applications. Minimization of the Willmore energy has been previously performed using flow methods [BS05] and using nonlinear optimization [WBH*07]. Sellán et al. [SKSJ20] use a minimum curvature flow to generate surfaces with bounded minimum curvature, producing results similar to those of Eigensatz et al. [ESP08], who

clamp the target principal curvature distribution to a specific interval. Additionally, some early works minimize curvature variation [JS07, WW92, MS92], which is of particular interest in surface blending and hole filling applications. Some optimization methods that seek minimum mean curvature variation surfaces using gradient flow techniques are found in [XZ07, CSV24].

3. Curvature Deformation Framework

We aim to deform a given surface mesh such that its principal curvatures best match a given set of target curvatures. This section introduces the various energies involved in the framework of Eigensatz et al. [ESP08, EP09], which our work is based on. We denote the surface mesh by $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, with vertices $\mathcal{V}$, edges $\mathcal{E}$, and faces $\mathcal{F}$. In the following, all properties denoted with a bar are calculated on the initial mesh. Each vertex $v_i \in \mathcal{V}$ has a corresponding 3D position denoted by $\mathbf{v}_i$. For each triangle face $f = (v_i, v_j, v_k)$, we introduce the interior angles α_i, α_j , and α_k at the three vertices and the edge vectors $\mathbf{e}_{ij}, \mathbf{e}_{jk}$, and $\mathbf{e}_{ki}$ (see inset).



The curvature energy encodes the primary objective of the optimization, which is matching the per-vertex principal curvatures κ_1 and κ_2 to user-prescribed target curvatures $\bar{\kappa}_{1,v}$ and $\bar{\kappa}_{2,v}$:

$$E_{\text{curv}}(\mathcal{M}) = \sum_{v \in \mathcal{V}} \bar{A}_v \left((\kappa_{1,v} - \bar{\kappa}_{1,v})^2 + (\kappa_{2,v} - \bar{\kappa}_{2,v})^2 \right) \quad (1)$$

The per-vertex deviation is weighted by the dual Voronoi area $\bar{A}_v$ of vertex v on the initial mesh to make the energy scale-invariant.

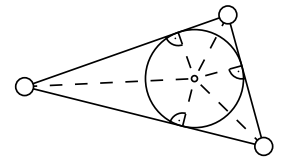
Two additional energy terms aim to preserve the metric of the initial surface. First, the area energy

$$E_{\text{area}}(f) = \frac{\bar{A}_f}{A_f} + \frac{A_f}{\bar{A}_f} \quad (2)$$

measures area deviation with respect to the initial surface, where A_f denotes the area of face f . Second, the conformal energy

$$E_{\text{conf}}(f) = \frac{\cot(\bar{\alpha}_i) \|\mathbf{e}_{jk}\|^2 + \cot(\bar{\alpha}_j) \|\mathbf{e}_{ki}\|^2 + \cot(\bar{\alpha}_k) \|\mathbf{e}_{ij}\|^2}{2A_f} \quad (3)$$

preserves the shape of a triangle face f . In order to avoid negative cotangent weights in Equation (3), Eigensatz et al. [EP09] apply a virtual subdivision, which splits each triangle into six virtual triangles using its incircle center (see inset). In this subdivision, the conformal energy is represented using six rational quadratic terms where all interior angles are ensured to be $\leq \frac{\pi}{2}$, thus leading to positive cotangent weights.



Both $E_{\text{area}}(f)$ and $E_{\text{conf}}(f)$ are then accumulated over the full surface as follows

$$\begin{aligned} E_{\text{area}}(\mathcal{M}) &= \frac{1}{\bar{A}_{\mathcal{M}}} \sum_{f \in \mathcal{F}} \bar{A}_f E_{\text{area}}(f) \\ E_{\text{conf}}(\mathcal{M}) &= \frac{1}{\bar{A}_{\mathcal{M}}} \sum_{f \in \mathcal{F}} \bar{A}_f E_{\text{conf}}(f) , \end{aligned} \quad (4)$$

where $\bar{A}_{\mathcal{M}} = \sum_{f \in \mathcal{F}} \bar{A}_f$ denotes the surface area of the initial mesh.

The two energies E_{area} and E_{conf} are needed to regularize the deformation and resolve ambiguities in the curvature energy. In some applications, the area distortion term may be optionally omitted if the additional constraints already prevent unnecessary surface area change. An additional position energy is added to keep the vertex positions close to their initial position

$$E_{\text{pos}}(\mathcal{M}) = \frac{1}{\bar{A}_{\mathcal{M}}^2} \sum_{v \in \mathcal{V}} \bar{A}_v \|\mathbf{v} - \bar{\mathbf{v}}\|^2 . \quad (5)$$

The position energy is needed to spatially control the deformation because all other energies are invariant to rigid transformations [EP09]. Finally, the complete energy is assembled by combining the four energy terms as a weighted sum

$$\begin{aligned} E(\mathcal{M}) &= w_{\text{curv}} E_{\text{curv}}(\mathcal{M}) + w_{\text{area}} E_{\text{area}}(\mathcal{M}) \\ &\quad + w_{\text{conf}} E_{\text{conf}}(\mathcal{M}) + w_{\text{pos}} E_{\text{pos}}(\mathcal{M}) . \end{aligned} \quad (6)$$

The corresponding weights are set to one, unless otherwise specified. The weight of the position energy is always set to $w_{\text{pos}} = 10^{-3}$ to keep its effect minimal. This framework can easily be extended with additional energies, such as the curve length and patch area constraints from [EP09].

4. Discretization of Principal Curvatures

The work of Eigensatz et al. [ESP08] used the curvature tensor approach [CSM03] to compute per-vertex principal curvatures. Their curvature tensor is defined as a 3×3 matrix

$$\mathbf{T}_v = \frac{1}{2A_v} \sum_{e \in \mathcal{E}_v} \|\mathbf{e}\| \beta_e \hat{\mathbf{e}} \hat{\mathbf{e}}^\top , \quad (7)$$

where β_e is the signed dihedral angle between the two faces adjacent to edge e and $\mathcal{E}_v$ is the set of incident edges at vertex v . The hat symbol denotes normalized vectors: $\hat{\mathbf{e}} = \mathbf{e} / \|\mathbf{e}\|$. The principal curvatures $\kappa_{1,v}$ and $\kappa_{2,v}$ are found as the two eigenvalues of $\mathbf{T}_v$ that are largest in magnitude. However, the principal curvature derivatives are undefined if any of the three eigenvalues are duplicated (which occurs at cylindrical and umbilical points). Eigensatz et al. [ESP08] approximate derivatives using the mean curvature gradient in case of any duplicated eigenvalues. Also, the calculation of gradients involves forming tensor-valued expressions of third order and derivatives of eigenvalues with respect to vertex positions.

4.1. Principal Curvatures

In our discretization, we first compute the mean and Gaussian curvature H_v and K_v at each vertex v and then use those to retrieve the principal curvatures $\kappa_{1,v}$ and $\kappa_{2,v}$. In order to express

the mean curvature of a mesh, we use the Steiner approximation [Ste40,BPW10]. The integrated mean curvature of a single interior edge e results in $\frac{1}{2} \|\mathbf{e}\| \beta_e$. By distributing half of each edge's mean curvature to its two vertices, the mean curvature around the neighborhood of a vertex v is [Sul08]

$$H_v = \frac{1}{4A_v} \sum_{e \in \mathcal{E}_v} \|\mathbf{e}\| \beta_e . \quad (8)$$

Similarly, the Gaussian curvature is discretized using the popular angle deficit formula [MDSB03,CSM03]

$$K_v = \frac{1}{A_v} \left(2\pi - \sum_{f \in \mathcal{F}_v} \alpha_{f,v} \right) , \quad (9)$$

where $\alpha_{f,v}$ denotes the interior angle in face f at the respective vertex v and $\mathcal{F}_v$ is the set of faces around vertex v . Both quantities H_v and K_v are divided by the associated dual vertex area A_v , which transforms the integrated curvatures into pointwise measures. We can then use the mean and Gaussian curvatures from Equations (8) and (9) to discretize the two principal curvatures

$$\kappa_{1,v} = H_v + c \quad \kappa_{2,v} = H_v - c , \quad (10)$$

where $c = \sqrt{\max\{0, H_v^2 - K_v\}}$. In the discrete setting, $H_v^2 - K_v \geq 0$ must be ensured by clamping [MDSB03].

One benefit of our discretization is that the principal curvature gradients with respect to the vertex positions can be found as a combination of gradients of basic properties like areas, interior angles, and dihedral angles. In contrast to the curvature tensor discretization, our method contains no tensor-valued expressions or derivatives of eigenvalues. We outline the specific gradient derivation in Appendix B. Also, our method is naturally differentiable at cylindrical points. In case of an umbilical point $H_v^2 - K_v \leq 0$, our principal curvatures intuitively reduce to the mean curvature H_v .

4.2. Dual Vertex Areas

We now address the specific realization of the dual vertex area A_v . Eigensatz et al. [ESP08] used barycentric areas (Figure 1a) because the derivatives are found simply as a sum of face area gradients

$$A_{\text{bary},v} = \sum_{f \in \mathcal{F}_v} \frac{1}{\text{val}(f)} A_f , \quad (11)$$

where $\text{val}(f)$ is the valence of face f . We instead suggest the use of mixed Voronoi areas, which are known to outperform barycentric areas in terms of curvature accuracy [MDSB03]. The dual Voronoi area of a vertex v is calculated as

$$A_{\text{Voro},v} = \frac{1}{8} \sum_{e \in \mathcal{E}_v} (\cot(\alpha_{e,0}) + \cot(\alpha_{e,1})) \|\mathbf{e}\|^2 , \quad (12)$$

where $\alpha_{e,0}$ and $\alpha_{e,1}$ are the interior angles opposite of edge e . As Figure 1b shows, Voronoi cells can lie outside of the triangle when interior angles are obtuse. To make the dual area more robust, Meyer et al. [MDSB03] propose mixed Voronoi areas, which clamp the Voronoi cells so that they lie within the triangle (Figure 1c). These mixed areas are implemented as simple per-face case distinction between Voronoi area and (scaled) face area, depending on whether a triangle is obtuse or not. The additional computational

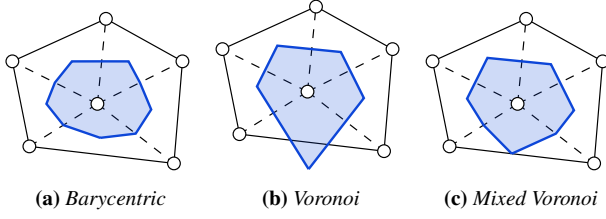


Figure 1: Different choices of dual areas for a vertex.

effort involved in the derivation thus exclusively comes from differentiating the Voronoi area in Equation (12).

Figure 2 demonstrates the advantage of mixed Voronoi areas at vertices with low valence, where barycentric areas yield larger errors in curvature estimation. Additionally, the example in Figure 4 shows that mixed Voronoi areas result in less distortion of triangle shapes, due to their more accurate representation of dual vertex area in case of obtuse triangles.

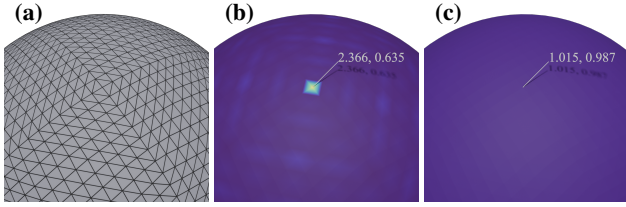


Figure 2: A sphere mesh with unit radius (a). Using barycentric areas for the dual vertex area (b) results in large errors for the low-valence vertex. Using mixed Voronoi areas (c) greatly improves the curvature estimation, bringing it closer to the target of (1, 1).

4.3. Robust Dihedral Angles

Numerical robustness also depends on the computation of dihedral angles β_e , which appear in the mean curvature discretization in Equation (8). The dihedral angle of a hinge is generally defined to be in the range $\beta_e \in (-\pi, \pi)$, avoiding the self-intersection at $\pm\pi$. However, when optimizing energies that include raw dihedral angles, as in the curvature tensor discretization [CSM03] used by Eigensatz et al. [ESP08], self-intersections are generally possible. This can lead to situations where the optimization gets stuck in a folded configuration (see Figure 3).

Various thin shell [GHDS03] and cloth simulation works [HP04, BMF03] have considered bending energies that approximate the local mean curvature of an edge using a function $\varphi(\beta_e)$ that depends on dihedral angles. We seek an energy that grows unbounded as $\beta_e \rightarrow \pm\pi$. We choose the tangent-based angle function that was previously used in [TG13, GSH*04, BS99] to express per-edge curvature. The raw dihedral angles in Equation (8) are replaced with the following function

$$\varphi(\beta_e) = 2 \tan\left(\frac{\beta_e}{2}\right). \quad (13)$$

This leads to a per-edge mean curvature of $\frac{1}{2} \|\mathbf{e}\| \varphi(\beta_e)$, extending the discretization from Equation (8). This notion of curvature can

also be understood as a result of Steiner's framework [Ste40] under a different definition of offset curves [BPW10, Sul08, BS99].

The benefit of our robust dihedral angle modification is demonstrated in Figure 3, where our formulation avoids the self-intersection that occur with the standard dihedral angle computation. Similar problems are also visible in Figures 4b and 4c, where multiple edges have extreme dihedral angles.

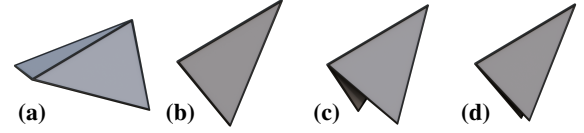


Figure 3: A hinge element (a) is optimized using target curvatures $\bar{\kappa}_1 = 10$, $\bar{\kappa}_2 = 0$. Using raw dihedral angles (b), the hinge folds into a self-intersecting state and the optimization gets stuck ($\kappa_1 = 3.8$). Using our robust dihedral angles (c), the expected result is achieved ($\kappa_1 = 10$). Even under more extreme conditions with target curvatures $\bar{\kappa}_1 = 50$, $\bar{\kappa}_2 = 0$ (d), the result remains coherent.

5. Numerical Optimization

We use the Gauss-Newton method [MNT04] to optimize the non-linear surface energy from Equation (6), which to this end it formulated as $E(\mathbf{x}) = \frac{1}{2} \mathbf{f}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$, where $\mathbf{f}(\mathbf{x})$ is the residual vector. The parameter vector $\mathbf{x}$ contains the flattened vertex positions of the surface mesh. With $\mathbf{J}(\mathbf{x})$ denoting the Jacobian of the residual vector, the energy gradient becomes $\nabla E(\mathbf{x}) = \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$. The derivative expressions required to construct the Jacobian $\mathbf{J}(\mathbf{x})$ are provided in Appendices A to D.

In each Gauss-Newton iteration, we solve the linear least squares system $\mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}) \delta = -\nabla E(\mathbf{x})$ for the step direction δ and update the current parameter vector $\mathbf{x} \leftarrow \mathbf{x} + h\delta$ using a step-size h .

This step-size h is found using a line-search procedure. We start with an initial step-size h_0 and then successively halve it until the Armijo condition $E(\mathbf{x} + h\delta) \leq E(\mathbf{x}) + 10^{-4} h \delta^\top \nabla E(\mathbf{x})$ [MNT04] is fulfilled. Many nonlinear optimizers, including the one used by Eigensatz et al. [EP09], simply start the line-search with $h_0 = 1$ and then reduce the step-size iteratively. Although this approach is sufficient for many problems, it can select an unreasonably large step-size, which can lead to instabilities, especially for larger deformations. We choose a more robust initial step-size strategy by Nocedal and Wright [NW06], which uses gradient scaling information to generate an initial guess. In the very first Gauss-Newton iteration, we set

$$h_0 = \min \left\{ 1, \frac{1}{\|\nabla E(\mathbf{x})\|_\infty} \right\}, \quad (14)$$

where $\|\cdot\|_\infty$ is the maximum norm. In all subsequent iterations, we use the energy value from the previous iteration $E(\mathbf{x}')$ to initialize

$$h_0 = \min \left\{ 1, \frac{2(E(\mathbf{x}) - E(\mathbf{x}'))}{\nabla E(\mathbf{x})^\top \delta} \right\}. \quad (15)$$

This initial step-size logic successfully prevents excessively large steps in the curvature optimization. Figure 4 compares our strategy

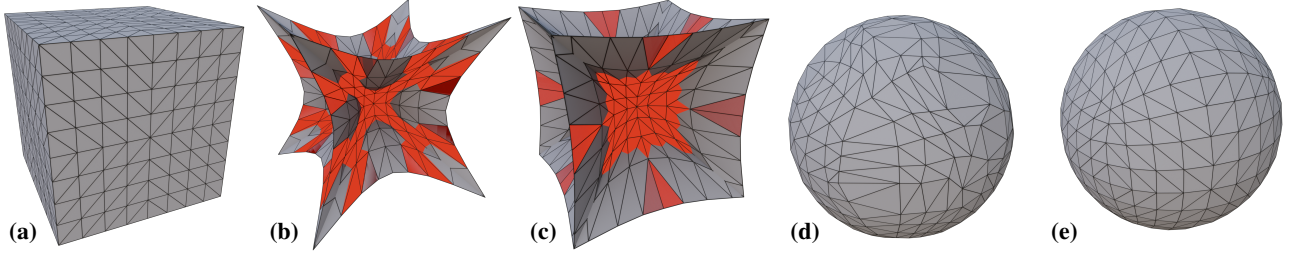


Figure 4: A cube mesh with unit side length (a) is deformed into a much smaller sphere using target curvatures $\tilde{\kappa}_1 = \tilde{\kappa}_2 = 10$. We perform the optimization with a small area term weight ($w_{\text{area}} = 10^{-5}$) to allow the surface to shrink. Without the improved step-size logic, the cube turns inside-out and breaks the optimization (b, c). The interior side of the mesh is colored red. Using the improved line-search, the optimization succeeds (d, e). Additionally, we display the difference between barycentric areas (b, d) and mixed Voronoi areas (c, e).

to the naive line-search starting at $h_0 = 1$, which is used by Eigensatz et al. [EP09]. The naive initialization takes an extremely large step in the first iteration, which causes the optimization to get stuck. In contrast, our method successfully achieves the desired surface, despite this significant deformation.

6. Geodesic Boundary Curvature

Both the curvature tensor method and our discretization using mean and Gaussian curvature provides inaccurate estimates of the principal curvatures at boundary vertices, because only a few interior edges contribute dihedral angles [Rus04].

To address this issue, we introduce the discrete geodesic curvature κ_g at the surface boundary [SSP08, SC17]. It indicates how the boundary curve bends within the tangent plane of the surface and is computed using an angle deficit formulation similar to the Gaussian curvature in Equation (9). Since the geodesic curvature κ_g is a curve property, it is divided by the dual length L_v of the boundary curve (see inset) to convert it to a pointwise measure

$$\kappa_{g,v} = \frac{1}{L_v} \left(\pi - \sum_{f \in \mathcal{F}_v} \alpha_{f,v} \right). \quad (16)$$

The optimization of boundary curvatures is easily integrated into our framework by adding the following energy to Equation (6)

$$E_{\text{boundary}} = \sum_{v \in \mathcal{B}} \bar{A}_v (\kappa_{g,v} - \tilde{\kappa}_{g,v})^2, \quad (17)$$

where $\mathcal{B}$ denotes the set of boundary vertices and $\tilde{\kappa}_{g,v}$ are the target geodesic curvatures. As shown in Figure 5, the addition of geodesic curvature improves control over the boundary shape when modifying the principal curvatures of a surface with a boundary.

7. Extension to Polygon Meshes

Triangle meshes are beneficial for modeling organic shapes, while quad meshes are better suited for technical models [BKP*10]. In the following, we demonstrate that our method is not limited to pure triangle meshes, but can be naturally extended to general polygon

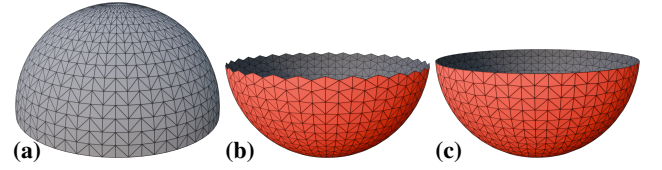
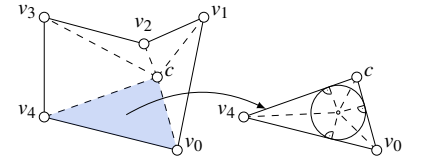


Figure 5: A hemisphere (a) is inverted using target curvatures $\tilde{\kappa}_1 = -\tilde{\kappa}_1$, $\tilde{\kappa}_2 = -\tilde{\kappa}_2$. Without constraining geodesic curvature, the boundary becomes ragged (b). By optimizing the geodesic boundary curvature $\tilde{\kappa}_g = 0$ these effects are successfully avoided (c). The interior side of the mesh is colored red.

meshes. In order to minimize our surface energy from Equation (6), we require derivatives of the discretized principal curvatures that generalize on polygons. We present analytic gradient expressions of interior and dihedral angles, face normals, and face areas on polygon meshes in Appendix A. From these, we derive the principal curvature gradients for general polygons in Appendix B.

The core challenge lies in implementing the conformal energy (Equation (3)) and the dual Voronoi area (Equation (12)). We opted for a pragmatic solution: subdividing each polygon of the initial mesh into virtual triangles. Inserting a virtual vertex using the method of Bunge et al. [BHKB20] creates a virtual triangle fan in the polygon face. We then proceed with the same calculations that we did for triangles and accumulate the contributions. For the conformal energy, we apply the incircle scheme by Eigensatz et al. [EP09] to each virtual triangle, creating six more virtual triangles. We outline this two-stage virtual subdivision in the inset. Our subdivision leads to a total of $6 \cdot \text{val}(f)$ Gauss-Newton residuals per polygon face for the conformal term. Both the virtual vertex weights from [BHKB20] and the barycentric coordinates in the per-triangle subdivision for the conformal term [EP09] are computed on the initial mesh and kept fixed during the optimization. During derivative computation, gradients with respect to the virtual vertex are distributed over the polygon vertices, weighted by the affine weights of Bunge et al. [BHKB20].



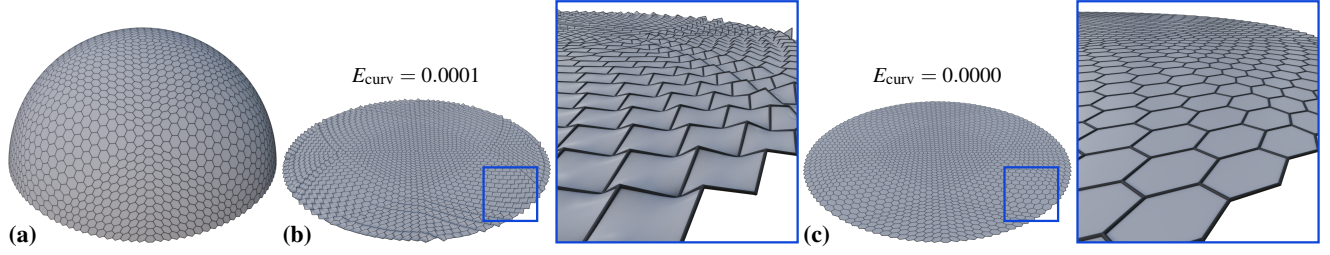


Figure 6: A polygon mesh of a hemisphere (a) is flattened using target curvatures $\tilde{\kappa}_1 = \tilde{\kappa}_2 = 0$. Without the planarity energy (b), the faces become crumpled. With the addition of the planarity energy, the mesh is successfully deformed into a plane (c). Note that, despite the mesh appearing heavily distorted without the planarity energy, the curvature error E_{curv} is rather low.

For general polygon meshes, we introduce an additional energy term to guide our optimization to produce planar faces. Similar to [dGBD20], we formulate an energy that penalizes the height of the polygon vertices relative to the plane defined by the face normal

$$E_{\text{planar}}(\mathcal{M}) = \frac{1}{\bar{A}_{\mathcal{M}}} \sum_{f \in \mathcal{F}} \sqrt{\bar{A}_f} \left(\sum_{v \in \mathcal{V}_f} |(\mathbf{v} - \mathbf{c}_f)^\top \hat{\mathbf{n}}_f| \right), \quad (18)$$

where $\mathbf{c}_f$ is the face barycenter and $\hat{\mathbf{n}}_f$ the unit-length face normal. Figure 6 demonstrates that this planarity term is necessary to achieve the expected polygon mesh and avoid unwanted distortion of polygonal faces.

We show another example of a hemisphere polygon mesh in Figure 7. Similar to the triangle mesh from Figure 5, our method produces the desired surface with inverted curvatures. With the addition of geodesic curvature, the boundary is smoothed.

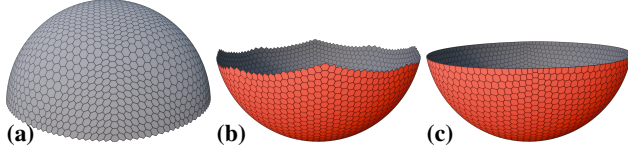


Figure 7: A hemisphere polygon mesh (a) is inverted using target curvatures $\tilde{\kappa}_1 = -\tilde{\kappa}_1$, $\tilde{\kappa}_2 = -\tilde{\kappa}_2$. Without constraining the geodesic curvature (b), the boundary is unnaturally deformed. With target geodesic curvature $\tilde{\kappa}_g = 0$ (c), the boundary is constrained to zero geodesic curvature. The interior side of the mesh is colored red.

8. Results

In this section, we present example applications of our principal curvature optimization framework. Our method is capable of many tasks, such as curvature-guided surface editing, curvature-domain smoothing, feature enhancement, and curvature clamping [ESP08].

In Figure 8, we showcase curvature-guided surface editing on a surface consisting of multiple smoothly connected cylinders. Doubling either the maximum or the minimum principal curvature yields the desired surface modifications.

Figure 9 shows an example in which the interior region of a technical mesh is first flattened by minimizing area using a fairing [BKP*10]. The target surface is then reconstructed from the original principal curvatures. Using the method of Eigensatz et

al. [ESP08], the optimization gets stuck in a state with a large curvature error and distorted triangle shapes. Our robust method can reconstruct the surface with a significantly lower curvature error while preserving triangle shapes.

Next, we demonstrate the application of curvature clamping, which is highly relevant in technical contexts. Figure 10 displays a technical part containing multiple convex and concave features. By clamping the curvature distribution to a specified interval, the surface can be dynamically modified. The deformations in Figures 10b and 10c reduce the curvature magnitude in the convex or concave regions. In Figure 10d, we completely remove all concavities.

Our method is not limited to optimizing prescribed target curvatures; it can also be used as a basis to minimize curvature variation functionals involving curvatures. In Figure 8d, we show how our optimization can minimize the Willmore energy $\int_{\mathcal{M}} H^2 dA$ [XZ07, BS05], where only the discretized mean curvature from Equation (8) is necessary.

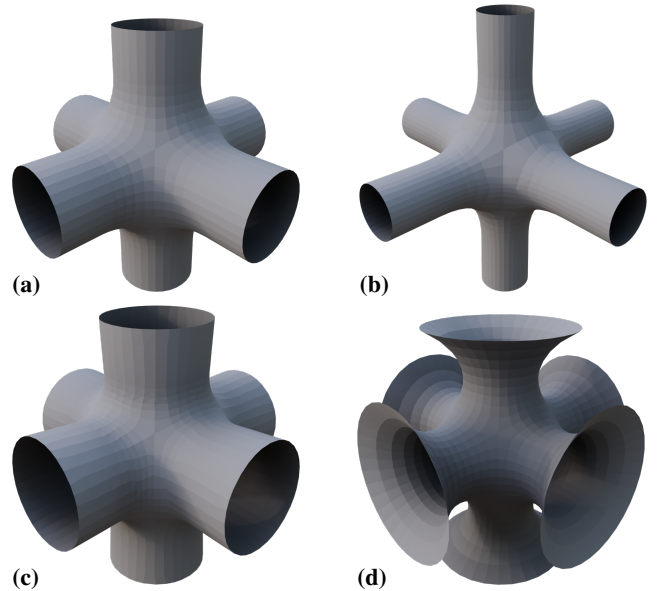


Figure 8: A mesh (a) is modified by doubling the maximum principal curvature $\tilde{\kappa}_1 = 2\tilde{\kappa}_1$ (b), doubling the minimum principal curvature $\tilde{\kappa}_2 = 2\tilde{\kappa}_2$ (c), and minimizing the Willmore energy (d).

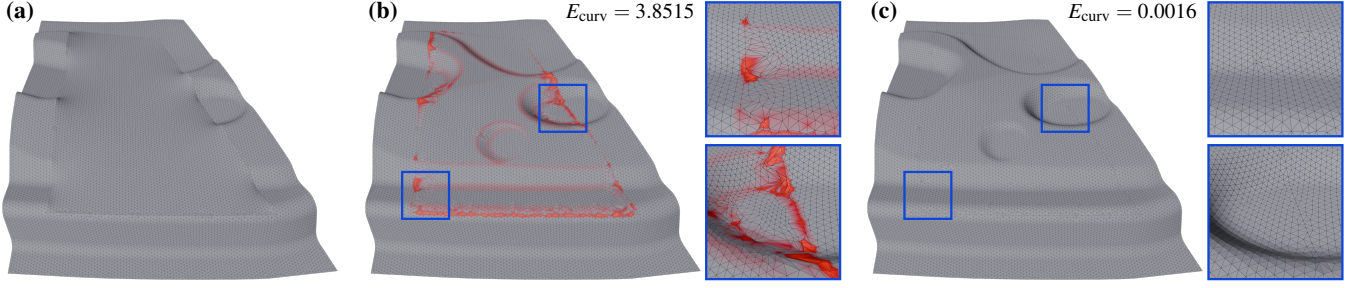


Figure 9: The interior region of a technical part is flattened by minimizing surface area (a). We reconstruct the surface from the initial principal curvatures. The method of Eigensatz et al. [ESP08] (b) produces distorted triangle shapes and gets stuck in a state with large curvature error. Our method (c) achieves a significantly lower curvature error while successfully preserving triangle shapes. Per-vertex curvature errors larger than 10^{-3} are shown in red.

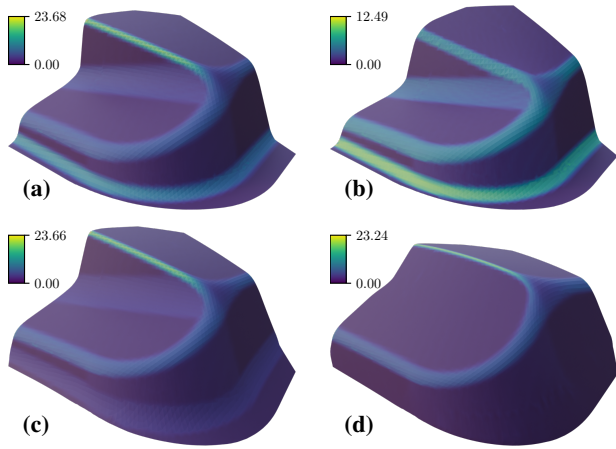


Figure 10: Modification of a technical part (a) using curvature clamping. The principal curvature distribution is clamped to the intervals $(-\infty, 5]$ (b) and $[-1, \infty)$ (c) which decreases curvature magnitude in either convex or concave regions. Clamping the distribution to the interval $[0, \infty)$ (d) removes all concave curvature features. Colors indicate the maximum absolute curvature.

Next, we introduce a curvature variation functional [MS92] using the two principal curvatures

$$E_{\text{var}}(\mathcal{M}) = \sum_{f \in \mathcal{F}} A_f \left(\|\nabla_f \kappa_1\|_p^p + \|\nabla_f \kappa_2\|_p^p \right), \quad (19)$$

where p controls the norm. For $p = 1$ this leads to the total curvature variation functional. In our optimization, the variation energy can be expressed using six individual Gauss-Newton residuals per triangle. The curvature gradients on the mesh are discretized using the standard formulation from differential geometry [BKP*10]

$$\nabla_f \kappa = \frac{1}{2A_f} (\kappa_{v_i} \hat{\mathbf{n}}_f \times \mathbf{e}_{jk} + \kappa_{v_j} \hat{\mathbf{n}}_f \times \mathbf{e}_{ki} + \kappa_{v_k} \hat{\mathbf{n}}_f \times \mathbf{e}_{ij}) \quad (20)$$

Figure 11 shows how curvature variation can be used for surface blending. We create an initial mesh to connect the two rings using an implicit surface fairing [BKP*10] by solving the tri-harmonic equation $\Delta^3 \mathbf{x} = \mathbf{0}$ while fixing the red boundary regions. Minimization of our total curvature variation energy produces a pipe surface

with almost constant radius. Figure 12 displays another application of our curvature variation functional. If we drive down p below one, the surface develops regions with piecewise constant curvatures. Sharp ruling lines appear that separate the regions.

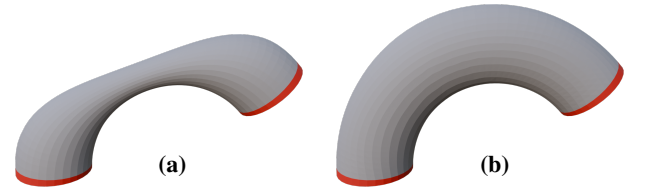


Figure 11: An initial surface (a) is created using a third-order fairing [DMSB99] connecting the two rings. Our total curvature variation functional produces a nearly constant-radius pipe surface (b).

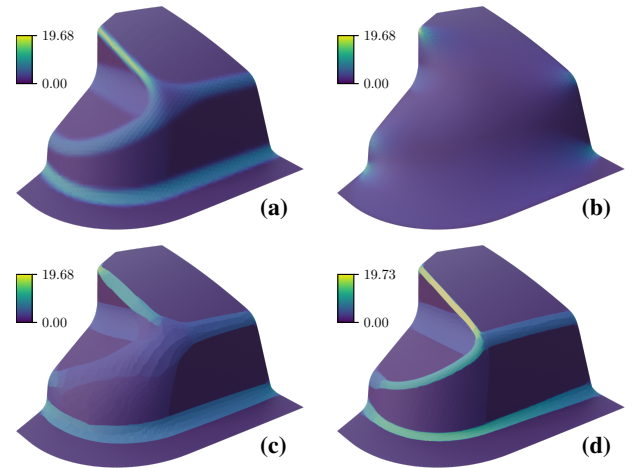
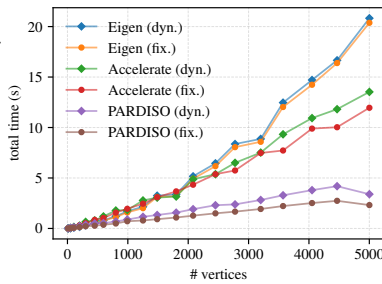


Figure 12: Optimization of a technical part (a) using our curvature variation functional. Two rings of vertices around the boundary are fixed and the area term is disabled ($w_{\text{area}} = 0$). Using total curvature variation (b), the surface flattens similar to a thin-plate solution. With $p = 0.5$ (c) and $p = 0.2$ (d), the surface evolves toward piecewise constant curvatures and develops sharp ruling lines. Colors indicate the maximum absolute curvature.

In practice, multiple implementations are available for solving the linear systems in our Gauss-Newton optimization. Inspired by PolySolve [SDG*19], for the sparse Cholesky decomposition we either use the standard Eigen solver [GJ*10], or the PARDISO solver [SGFS01] from the Intel MKL library, or the Accelerate framework [App] on Apple machines. Since the mesh topology remains constant during optimization, the sparsity pattern of our linear system is constant as well. We take advantage of this by performing the symbolic part of the decomposition only in the first iteration and reusing it in subsequent iterations.

We test our method by inverting the principal curvatures of a hemisphere at different mesh resolutions. The target curvatures are the same as in Figure 5. For our benchmark, we used a Linux workstation with an AMD 9800X3D CPU for Eigen and PARDISO, and an M2 MacBook Air for Accelerate. The inset displays the total computation time with increasing numbers of vertices. Both PARDISO and Accelerate outperform the solver from Eigen by a large margin and show a linear increase in time with growing numbers of vertices. Exploitation of the fixed matrix pattern (fix.) leads to a slight improvement compared to dynamically computing the symbolic factorization each iteration (dyn.).



9. Conclusions

This paper presents a versatile framework for robustly deforming surface meshes driven by principal curvatures. Our improved discretization of principal curvatures enhances both accuracy and robustness while simultaneously simplifying the computation of derivatives. By providing geodesic boundary curvature as an additional optimization target, we enable finer control over the deformation process. Furthermore, the extension of our method to general polygon meshes allows for more flexible and universal curvature-based surface optimization. The optimization of curvature functionals further advances the possible applications of our proposed framework. Our principal curvature variation functional with a variable norm can be applied to generate surfaces with piecewise constant curvatures.

References

- [App] APPLE: Accelerate. <https://developer.apple.com/documentation/accelerate>. 8
- [AW11] ALEXA M., WARDETZKY M.: Discrete Laplacians on general polygonal meshes. *ACM Transactions on Graphics* 30, 4 (2011), 102:1–102:10. 9
- [BCGB08] BEN-CHEN M., GOTSCHMAN C., BUNIN G.: Conformal flattening by curvature prescription and metric scaling. *Computer Graphics Forum* 27, 2 (2008), 449–458. 2
- [BHK20] BUNGE A., HERHOLZ P., KAZHDAN M., BOTSCH M.: Polygon Laplacian made simple. *Computer Graphics Forum* 39, 2 (2020), 303–313. 5, 10
- [BKP*10] BOTSCH M., KOBELT L., PAULY M., ALLIEZ P., LEVY B.: *Polygon Mesh Processing*. CRC Press, 2010. 1, 5, 6, 7
- [BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Symposium of Computer Animation* (2003), pp. 28–36. 2, 4, 9
- [BPW10] BAUER U., POLTHIER K., WARDETZKY M.: Uniform convergence of discrete curvatures from nets of curvature lines. *Discrete & Computational Geometry* 43, 4 (2010), 798–823. 3, 4
- [BS99] BOBENKO A. I., SURIS Y. B.: Discrete time Lagrangian mechanics on Lie groups, with an application to the Lagrange top. *Communications in Mathematical Physics* 204 (1999), 147–188. 2, 4
- [BS05] BOBENKO A. I., SCHRÖDER P.: Discrete Willmore flow. In *Symposium on Geometry Processing* (2005), pp. 101–110. 2, 6
- [BWH*06] BERGOU M., WARDETZKY M., HARMON D., ZORIN D., GRINSUN E.: A quadratic bending model for inextensible surfaces. In *Symposium on Geometry Processing* (2006), pp. 227–230. 2
- [Cor24] CORMAN E.: Curvature-driven conformal deformations. *ACM Transactions on Graphics* 43, 4 (2024), 139:1–139:16. 2
- [CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted Delaunay triangulations and normal cycle. In *Symposium on Computational Geometry* (2003), pp. 312–321. 1, 3, 4
- [CSV24] CAFFARELLI L., STINGA P., VIVAS H.: A PDE approach to the existence and regularity of surfaces of minimum mean curvature variation. *Archive for Rational Mechanics and Analysis* 248, 70 (2024). 2
- [dGBD20] DE GOES F., BUTTS A., DESBRUN M.: Discrete differential operators on polygonal meshes. *ACM Transactions on Graphics* 39, 4 (2020), 110:1–110:14. 6
- [DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. of ACM SIGGRAPH* (1999), pp. 317–324. 2, 7
- [EP09] EIGENSATZ M., PAULY M.: Positional, metric, and curvature control for constraint-based surface deformation. *Computer Graphics Forum* 28, 2 (2009), 551–558. 1, 2, 3, 4, 5
- [ESP08] EIGENSATZ M., SUMNER R., PAULY M.: Curvature-domain shape processing. *Computer Graphics Forum* 27, 2 (2008), 241–250. 1, 2, 3, 4, 6, 7, 10
- [FZL*20] FANG Q., ZHAO Z.-Y., LIU Z.-Y., LIU L., FU X.-M.: Metric first reconstruction for interactive curvature-aware modeling. *Computer-Aided Design* 126 (2020). 2
- [GHDS03] GRINSUN E., HIRANI A. N., DESBRUN M., SCHRÖDER P.: Discrete shells. In *Symposium of Computer Animation* (2003), pp. 62–67. 2, 4
- [GJ*10] GUENNEBAUD G., JACOB B., ET AL.: Eigen. <https://libeigen.gitlab.io>, 2010. 8
- [GSH*04] GINGOLD Y., SECORD A., HAN J., GRINSUN E., ZORIN D.: A discrete model for inelastic deformation of thin shells. *Technical Report, Courant Institute of Mathematical Sciences* (2004). 2, 4
- [HP04] HILDEBRANDT K., POLTHIER K.: Anisotropic filtering of non-linear surface features. *Computer Graphics Forum* 23, 3 (2004), 391–400. 2, 4
- [HPS08] HORMANN K., POLTHIER K., SHEFFER A.: Mesh parameterization: theory and practice. *ACM SIGGRAPH Courses* (2008). 2
- [JS07] JOSHI P., SEQUIN C.: Energy minimizers for curvature-based surface functionals. *Computer-Aided Design and Applications* 4, 5 (2007), 607–617. 2
- [JZW14] JIN S., ZHANG Y., WANG C. C.: Deformation with enforced metrics on length, area and volume. *Computer Graphics Forum* 33, 2 (2014), 429–438. 2
- [KEM14] KINER Y., ENDO S., MAEKAWA T.: Surface design based on direct curvature editing. *Computer-Aided Design* 55 (2014), 1–12. 2
- [KMB20] KAMMANN L., MENZEL S., BOTSCH M.: A compact patch-based representation for technical mesh models. In *Vision, Modeling, and Visualization* (2020), pp. 1–8. 2

- [MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. *Visualization and Mathematics III* (2003), 35–57. 1, 3, 10
- [MNT04] MADSEN K., NIELSEN H., TINGLEFF O.: *Methods for Non-Linear Least Squares Problems*. Informatics and Mathematical Modelling, Technical University of Denmark, 2004. 4
- [MS92] MORETON H. P., SÉQUIN C. H.: Functional optimization for fair surface design. In *Proc. of ACM SIGGRAPH* (1992), pp. 167–176. 2, 7
- [NISA07] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Fiber-Mesh: Designing freeform surfaces with 3D curves. *ACM Transactions on Graphics* 26, 3 (2007), 41:1–41:9. 1
- [NW06] NOCEDAL J., WRIGHT S.: *Numerical Optimization*. Springer, 2006. 4
- [Rus04] RUSINKIEWICZ S.: Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization and Transmission* (2004), pp. 486–493. 5
- [SC17] SAWHNEY R., CRANE K.: Boundary first flattening. *ACM Transactions on Graphics* 37, 1 (2017), 5:1–5:14. 2, 5
- [SDG*19] SCHNEIDER T., DUMAS J., GAO X., ZORIN D., PANOZZO D.: PolyFEM. <https://polyfem.github.io>, 2019. 8
- [SGFS01] SCHENK O., GÄRTNER K., FICHTNER W., STRICKER A.: PARDISO: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems* 18, 1 (2001), 69–78. 8
- [SKSJ20] SELLÁN S., KESTEN J., SHENG A. Y., JACOBSON A.: Opening and closing surfaces. *ACM Transactions on Graphics* 39, 6 (2020), 198:1–198:13. 2
- [SSP08] SPRINGBORN B., SCHRÖDER P., PINKALL U.: Conformal equivalence of triangle meshes. *ACM Transactions on Graphics* 27, 3 (2008), 77:1–77:11. 2, 5
- [Ste40] STEINER J.: Über parallele Flächen. *Monatsbericht der Akademie der Wissenschaften zu Berlin* (1840), 114–118. 3, 4
- [Sul08] SULLIVAN J. M.: Curvatures of smooth and discrete surfaces. *Discrete Differential Geometry* 38 (2008), 175–188. 3, 4
- [Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *Proc. of ACM SIGGRAPH* (1995), pp. 351–358. 2
- [TG13] TAMSTORF R., GRINSPUN E.: Discrete bending forces and their Jacobians. *Graphical Models* 75, 6 (2013), 362–370. 2, 4, 9
- [TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proc. of ACM SIGGRAPH* (1987), pp. 205–214. 2
- [WBH*07] WARDETZKY M., BERGOU M., HARMON D., ZORIN D., GRINSPUN E.: Discrete quadratic curvature energies. *Computer Aided Geometric Design* 24, 8 (2007), 499–518. 2
- [WW92] WELCH W., WITKIN A.: Variational surface modeling. In *Proc. of ACM SIGGRAPH* (1992), pp. 157–166. 2
- [XZ07] XU G., ZHANG Q.: G2 surface modeling using minimal mean-curvature-variation flow. *Computer-Aided Design* 39, 5 (2007), 342–351. 1, 2, 6

Appendix A: Derivatives of Basic Quantities

The minimization of the energy in Equation (6) using Gauss-Newton requires first-order derivatives of multiple mesh quantities. We define all derivatives so that they are compatible with general polygon meshes. To clarify the notation, we use the vertices v_i, v_j, v_k to denote specific vertices and v_* to denote an arbitrary vertex. The hat symbol denotes vector normalization: $\hat{\mathbf{e}} = \mathbf{e}/\|\mathbf{e}\|$. First, we

introduce the gradient expressions for the norm, the squared norm, and the normalized vector associated with some vector $\mathbf{p}$

$$\begin{aligned}\frac{\partial \|\mathbf{p}\|}{\partial \mathbf{p}} &= \frac{\mathbf{p}}{\|\mathbf{p}\|} \\ \frac{\partial \|\mathbf{p}\|^2}{\partial \mathbf{p}} &= 2\mathbf{p} \\ \frac{\partial \hat{\mathbf{p}}}{\partial \mathbf{p}} &= \frac{1}{\|\mathbf{p}\|} \left(\mathbf{I} - \hat{\mathbf{p}}\hat{\mathbf{p}}^\top \right).\end{aligned}\quad (21)$$

The gradient of an interior angle α_i influenced by the two edges e_{ij} and e_{ki} is given as

$$\begin{aligned}\frac{\partial \alpha_i}{\partial \mathbf{v}_j} &= \frac{\mathbf{e}_{ij} \times \hat{\mathbf{n}}_{ijk}}{\|\mathbf{e}_{ij}\|^2} \\ \frac{\partial \alpha_i}{\partial \mathbf{v}_k} &= \frac{\mathbf{e}_{ki} \times \hat{\mathbf{n}}_{ijk}}{\|\mathbf{e}_{ki}\|^2} \\ \frac{\partial \alpha_i}{\partial \mathbf{v}_i} &= -\frac{\partial \alpha_i}{\partial \mathbf{v}_j} - \frac{\partial \alpha_i}{\partial \mathbf{v}_k},\end{aligned}\quad (22)$$

where $\hat{\mathbf{n}}_{ijk} = \frac{\mathbf{e}_{ij} \times \mathbf{e}_{ik}}{\|\mathbf{e}_{ij} \times \mathbf{e}_{ik}\|}$.

An important quantity of a polygonal face is the vector area $\mathbf{a}_f = \frac{1}{2} \sum_{v_i \in f} \mathbf{v}_i \times \mathbf{v}_{i+1}$, where v_{i+1} denotes the next vertex in the polygon's cyclic vertex loop. For a triangle f , the vector area yields exactly its area and normal as $A_f = \|\mathbf{a}_f\|$ and $\mathbf{n}_f = \hat{\mathbf{a}}_f$, respectively; we use the same expressions to generalize area and normal to polygons [AW11]. This yields the gradients of face area

$$\frac{\partial A_f}{\partial \mathbf{v}_i} = \frac{1}{2} \left(\hat{\mathbf{a}}_f \times (\mathbf{v}_{i-1} - \mathbf{v}_{i+1}) \right), \quad (23)$$

where v_{i-1} and v_{i+1} again denote previous/next vertex within the polygon. The face normal's Jacobian is

$$\frac{\partial \hat{\mathbf{n}}_f}{\partial \mathbf{v}_i} = \frac{1}{\|\mathbf{a}_f\|} \left(\mathbf{I} - \hat{\mathbf{a}}_f \hat{\mathbf{a}}_f^\top \right) [\mathbf{v}_{i-1} - \mathbf{v}_{i+1}]_\times, \quad (24)$$

where $[\cdot]_\times$ is the skew-symmetric matrix such that $[\mathbf{p}]_\times \mathbf{q} = \mathbf{p} \times \mathbf{q}$ for two three-dimensional vectors $\mathbf{p}$ and $\mathbf{q}$.

Next, the necessary derivatives of the dihedral angle β_e of an edge e are introduced. We derive a gradient expression for dihedral bending on general polygon meshes, but refer to existing literature [BMF03, TG13] for more efficient formulations in case of triangle meshes. We write the dihedral angle of an edge e as

$$\begin{aligned}\beta_e &= \arctan \frac{\sin(\beta_e)}{\cos(\beta_e)} \\ \sin(\beta_e) &= \hat{\mathbf{e}}^\top (\hat{\mathbf{n}}_0 \times \hat{\mathbf{n}}_1) \\ \cos(\beta_e) &= \hat{\mathbf{n}}_0^\top \hat{\mathbf{n}}_1,\end{aligned}\quad (25)$$

where $\hat{\mathbf{n}}_0$ and $\hat{\mathbf{n}}_1$ are the face normals of the two adjacent faces. The respective dihedral angle gradient is found as

$$\frac{\partial \beta_e}{\partial \mathbf{v}_*} = \frac{\partial \sin(\beta_e)}{\partial \mathbf{v}_*} \cos(\beta_e) - \frac{\partial \cos(\beta_e)}{\partial \mathbf{v}_*} \sin(\beta_e). \quad (26)$$

The derivatives of $\sin(\beta_e)$ and $\cos(\beta_e)$ are calculated as follows

$$\begin{aligned} \frac{\partial \sin(\beta_e)}{\partial \mathbf{v}_*} &= \left([\hat{\mathbf{n}}_0]_\times \frac{\partial \hat{\mathbf{n}}_1}{\partial \mathbf{v}_*} - [\hat{\mathbf{n}}_1]_\times \frac{\partial \hat{\mathbf{n}}_0}{\partial \mathbf{v}_*} \right)^\top \hat{\mathbf{e}} + \left(\frac{\partial \hat{\mathbf{e}}}{\partial \mathbf{v}_*} \right)^\top (\hat{\mathbf{n}}_0 \times \hat{\mathbf{n}}_1) \\ \frac{\partial \cos(\beta_e)}{\partial \mathbf{v}_*} &= \left(\frac{\partial \hat{\mathbf{n}}_0}{\partial \mathbf{v}_*} \right)^\top \hat{\mathbf{n}}_1 + \left(\frac{\partial \hat{\mathbf{n}}_1}{\partial \mathbf{v}_*} \right)^\top \hat{\mathbf{n}}_0. \end{aligned} \quad (27)$$

The gradient of our robust dihedral angle $\varphi(\beta_e)$ from Equation (13) is then written using another chain rule

$$\frac{\partial \varphi(\beta_e)}{\partial \mathbf{v}_*} = \cos\left(\frac{\beta_e}{2}\right)^{-2} \frac{\partial \beta_e}{\partial \mathbf{v}_*}. \quad (28)$$

Appendix B: Derivatives of Principal Curvatures

Using the previous derivatives, the mean and Gaussian curvature gradients are obtained through straightforward application of product and quotient rules. For better readability, we first compute derivatives of the integrated curvature measures $H_{\text{int},v}$ and $K_{\text{int},v}$, omitting the division by dual vertex area in Equations (8) and (9). The integrated mean curvature is differentiated as follows

$$\frac{\partial H_{\text{int},v}}{\partial \mathbf{v}_*} = \frac{1}{4} \sum_{e \in \mathcal{E}_v} \frac{\partial \beta_e}{\partial \mathbf{v}_*} \|\mathbf{e}\| + \frac{\partial \|\mathbf{e}\|}{\partial \mathbf{v}_*} \beta_e. \quad (29)$$

Similarly, for the integrated Gaussian curvature (Equation (9))

$$\frac{\partial K_{\text{int},v}}{\partial \mathbf{v}_*} = - \sum_{f \in \mathcal{F}_v} \frac{\partial \alpha_f}{\partial \mathbf{v}_*}. \quad (30)$$

Conveniently, the gradient of the geodesic boundary curvature (Equation (16)) equals the Gaussian curvature gradient. The gradient of the pointwise mean curvature H_v is then calculated using the quotient rule

$$\frac{\partial H_v}{\partial \mathbf{v}_*} = \frac{1}{A_v} \frac{\partial H_{\text{int},v}}{\partial \mathbf{v}_*} - \frac{H_{\text{int},v}}{A_v^2} \frac{\partial A_v}{\partial \mathbf{v}_*}, \quad (31)$$

and analogously for the pointwise Gaussian curvature K_v .

Finally, we arrive at the principal curvatures using the relation from Equation (10) and write their gradients as

$$\frac{\partial \kappa_{1,2}}{\partial \mathbf{v}_*} = \frac{\partial H_v}{\partial \mathbf{v}_*} \pm \frac{1}{2\sqrt{H_v^2 - K_v}} \left(2H_v \frac{\partial H_v}{\partial \mathbf{v}_*} - \frac{\partial K_v}{\partial \mathbf{v}_*} \right). \quad (32)$$

Direct implementation using the gradient in Equation (32) is not feasible for our Gauss-Newton optimizer because the derivative of $\sqrt{H_v^2 - K_v}$ becomes unbounded at umbilical points. We replace the square root in Equation (10) with a standard robust formulation $\sqrt{x+\epsilon} - \sqrt{\epsilon}$. We choose $\epsilon = 10^{-8}$. Note that our principal curvature gradients naturally converge to the mean curvature gradient $\frac{\partial H_v}{\partial \mathbf{v}_*}$ at umbilics, modeling the same behavior as in [ESP08] without the need for hard conditional branching.

Appendix C: Derivatives of Dual Vertex Areas

The gradient of the barycentric area $A_{\text{bary},v}$ (Equation (11)) is easily found as a combination of the face area gradients from Equation (23) weighted by their contribution to the barycentric cell

$$\frac{\partial A_{\text{bary},v}}{\partial \mathbf{v}_*} = \sum_{f \in \mathcal{F}_v} \frac{1}{\text{val}(f)} \frac{\partial A_f}{\partial \mathbf{v}_*}. \quad (33)$$

In order to compute the Voronoi area gradient, we first derive the cotangent gradient of an interior angle α_i at the two edges e_{ij} and e_{ki} . For $\cot(\alpha_i) = \frac{\mathbf{e}_{ij}^\top \mathbf{e}_{ik}}{\|\mathbf{n}_{ijk}\|}$, with $\mathbf{n}_{ijk} = \mathbf{e}_{ij} \times \mathbf{e}_{ik}$, the gradients are given as

$$\begin{aligned} \frac{\partial \cot(\alpha_i)}{\partial \mathbf{v}_j} &= \frac{1}{\|\mathbf{n}_{ijk}\|} \left(\mathbf{e}_{ik} - \mathbf{e}_{ij}^\top \mathbf{e}_{ik} \frac{\hat{\mathbf{n}}_{ijk} \times \mathbf{e}_{ik}}{\|\mathbf{n}_{ijk}\|} \right) \\ \frac{\partial \cot(\alpha_i)}{\partial \mathbf{v}_k} &= \frac{1}{\|\mathbf{n}_{ijk}\|} \left(\mathbf{e}_{ij} - \mathbf{e}_{ij}^\top \mathbf{e}_{ik} \frac{\hat{\mathbf{n}}_{ijk} \times \mathbf{e}_{ij}}{\|\mathbf{n}_{ijk}\|} \right) \\ \frac{\partial \cot(\alpha_i)}{\partial \mathbf{v}_i} &= - \frac{\partial \cot(\alpha_i)}{\partial \mathbf{v}_j} - \frac{\partial \cot(\alpha_i)}{\partial \mathbf{v}_k}. \end{aligned} \quad (34)$$

The Voronoi area $A_{\text{Voro},v}$ (Equation (12)) gradient is then calculated by applying the product rule

$$\begin{aligned} \frac{\partial A_{\text{Voro},v}}{\partial \mathbf{v}_*} &= \frac{1}{8} \sum_{e \in \mathcal{E}_v} \left(\frac{\partial \cot(\alpha_0)}{\partial \mathbf{v}_*} + \frac{\partial \cot(\alpha_1)}{\partial \mathbf{v}_*} \right) \|\mathbf{e}\|^2 \\ &\quad + \frac{\partial \|\mathbf{e}\|^2}{\partial \mathbf{v}_*} (\cot(\alpha_0) + \cot(\alpha_1)). \end{aligned} \quad (35)$$

Note that if an edge e is a boundary edge, then only one of the opposite interior angles α_0 and α_1 is included. The mixed Voronoi area gradient [MDSB03] is found by using a per-face case distinction between Voronoi area and scaled triangle area in the above equations. For Voronoi areas on polygons, the above gradients are also computed with respect to the virtual vertex in the subdivided polygon. These gradients are then distributed over the polygon's vertices, weighted by the affine weights of the virtual vertex [BHKB20].

Appendix D: Derivatives of Curvature Variation

Finally, we require derivatives of the curvature variation energy from Equation (19). We differentiate the principal curvature gradient $\nabla_f \kappa$ from Equation (20) to obtain its Jacobian

$$\begin{aligned} \frac{\partial \nabla_f \kappa}{\partial \mathbf{v}_*} &= \frac{1}{2A_f} \left[\frac{\partial \kappa_{v_i}}{\partial \mathbf{v}_*} (\hat{\mathbf{n}}_f \times \mathbf{e}_{jk})^\top \right. \\ &\quad + \frac{\partial \kappa_{v_j}}{\partial \mathbf{v}_*} (\hat{\mathbf{n}}_f \times \mathbf{e}_{ki})^\top + \frac{\partial \kappa_{v_k}}{\partial \mathbf{v}_*} (\hat{\mathbf{n}}_f \times \mathbf{e}_{ij})^\top \\ &\quad + \kappa_{v_i} \left(-[\mathbf{e}_{jk}]_\times \frac{\partial \hat{\mathbf{n}}_f}{\partial \mathbf{v}_*} + [\hat{\mathbf{n}}_f]_\times \frac{\partial \mathbf{e}_{jk}}{\partial \mathbf{v}_*} - \frac{\partial A_f}{\partial \mathbf{v}_*} \frac{(\hat{\mathbf{n}}_f \times \mathbf{e}_{jk})^\top}{A_f} \right) \\ &\quad + \kappa_{v_j} \left(-[\mathbf{e}_{ki}]_\times \frac{\partial \hat{\mathbf{n}}_f}{\partial \mathbf{v}_*} + [\hat{\mathbf{n}}_f]_\times \frac{\partial \mathbf{e}_{ki}}{\partial \mathbf{v}_*} - \frac{\partial A_f}{\partial \mathbf{v}_*} \frac{(\hat{\mathbf{n}}_f \times \mathbf{e}_{ki})^\top}{A_f} \right) \\ &\quad \left. + \kappa_{v_k} \left(-[\mathbf{e}_{ij}]_\times \frac{\partial \hat{\mathbf{n}}_f}{\partial \mathbf{v}_*} + [\hat{\mathbf{n}}_f]_\times \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{v}_*} - \frac{\partial A_f}{\partial \mathbf{v}_*} \frac{(\hat{\mathbf{n}}_f \times \mathbf{e}_{ij})^\top}{A_f} \right) \right] \end{aligned} \quad (36)$$

where the face area gradient and face normal Jacobian are found in Equations (23) and (24).