Opening and Closing Surfaces

SILVIA SELLÁN, University of Oviedo, Fields Institute and University of Toronto
JACOB KESTEN, Fields Institute and University of Toronto
ANG YAN SHENG, Fields Institute and University of Toronto
ALEC JACOBSON, University of Toronto

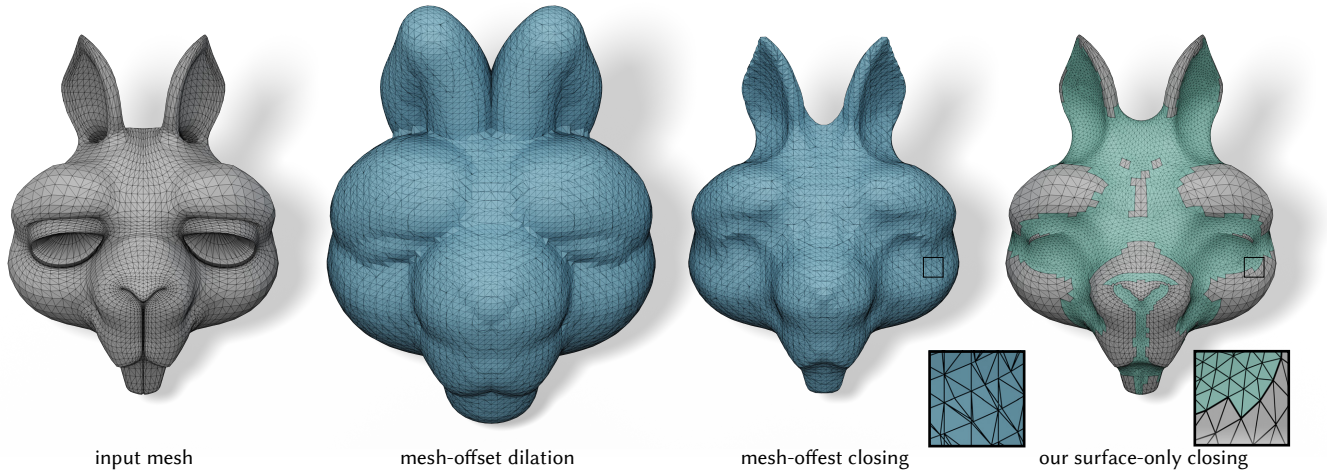


Fig. 1. This paper considers conducting closing and opening operations *without* composing dilations and erosions explicitly. Conducting a mesh-based offset for dilation changes the surface entirely. Eroding this with another (negative) offset to complete the closing operation retains much of the original geometry, but with a new discretization everywhere. Instead, our *surface-only* closing flow perfectly preserves the mesh in regions that the closing does not change (see blowups). The original quads can even be re-identified in those areas. 3D model by The Blender Foundation under CC-BY 3.0.

We propose a new type of curvature flow for curves in 2D and surfaces in 3D. The flow is inspired by the mathematical morphology *opening* and *closing* operations. These operations are classically defined by composition of dilation and erosion operations. In practice, existing methods implemented this way will result in re-discretizing the entire shape, even if some parts of the surface do not change. Instead, our surface-only curvature-based flow moves the surface selectively in areas that should be repositioned. In our triangle mesh discretization, vertices in regions unaffected by the opening or closing will remain exactly in place and do not affect our method's complexity, which is output-sensitive.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**; • **Mathematics of computing** → *Differential equations*.

Additional Key Words and Phrases: mathematical morphology

ACM Reference Format:
Silvia Sellán, Jacob Kesten, Ang Yan Sheng, and Alec Jacobson. 2020. Opening and Closing Surfaces. *ACM Trans. Graph.* 39, 6, Article 198 (December 2020), 13 pages. https://doi.org/10.1145/3414685.3417778

Address: Silvia Sellán, sgsellan@cs.toronto.edu.

0730-0301/2020/12-ART198 $15.00
https://doi.org/10.1145/3414685.3417778

1 INTRODUCTION

Mathematical morphology describes how a shape grows or shrinks according to a given *operation* and *structuring element* (or kernel). For example, the dilation operation is defined by placing the structuring element at each point in the input shape and conducting a union (also known as a Minkowski sum). As a result, the shape grows outward. Erosion is defined similarly on the shape's complement, and the shape shrinks inward.

These basic operations give way to more complex ones such as the *opening* and *closing* operations. For example, the opening of a solid shape returns the union of all points inside the shape that can be covered by placing the given structuring element somewhere strictly inside the shape (see Fig. 2). These powerful operations enjoy use in applications such as tool reachability in computational fabrication, cage design for computer animation, or shape abstraction in geometric modeling.

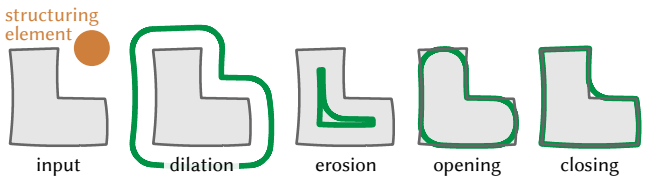


Fig. 2. Mathematical morphology grows or shrinks an input shape according to an *operation* (e.g., "opening") and a *structuring element*.

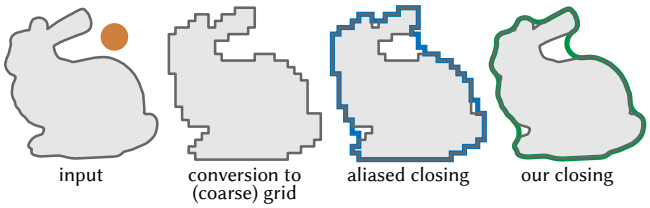


Fig. 3. Volumetric morphology operations introduce coordinate system bias and aliasing (exaggerated here). Our flows operate directly on the input discretization boundary.

Opening and closing operations are traditionally defined and implemented by composing erosions and dilations. For example, the opening is equivalent to erosion followed by dilation. While erosions and dilations are naturally defined for volumetric shape representations (e.g., voxel lattices or distance fields), if the input shape arrives as a surface representation (e.g., triangle mesh) conversion to a volumetric representation can result in a loss of time, accuracy, and flexibility.

In contrast, we define opening and closing operations with ball-shaped structuring elements directly as differential *flows* of the input shape's *surface*. We avoid composition of erosion and dilation operations, which would each change the entire surface of a shape. Instead, we observe that — in general — opening and closing operations will leave some parts of the shape in place. We identify these regions based on curvature during an evolving surface flow.

This insight connects the theories of mean-curvature flow of surfaces and mathematical morphology of volumetric solids. Our surface-based definitions of opening and closing enable our proposed discretization for piecewise-linear curves in 2D and triangle-meshes in 3D. We introduce a semi-implicit time integration method based on a modified Dirichlet energy with an adaptive remeshing step. In contrast to existing smooth flows, to achieve closing and opening behaviors we introduce a curvature-based *obstacle*, which prevents the flow from continuing beyond the curvature of the given structuring element. Unlike approaches that compose erosion and dilation, our method is guaranteed to only move vertices or change connectivity near regions of the surface that should actually change during the desired opening or closing operation. Unlike voxel- or grid-based operations, our method is coordinate-system independent and consequently does not suffer from grid aliasing artifacts (see Fig. 3). Considering asymptotic complexity compared to a traditional voxel-grid approach, our algorithm reduces the problem in dimension. Further, moving only the vertices that *should* move makes our method's complexity output-sensitive both in theory and in our implementation. Our mesh implementation relies on standard subroutines from the geometry processing toolbox.

The partial differential equation defining our flow is defined by local *surface* information, while the standard morphological operations are defined by local *spatial* information. This disparity leads to interesting differences that can emerge in areas where surface-geodesic distance differs from Euclidean distance. We analyze this situation in detail and explore some applications in which our definition is preferrable to the traditional one. Our flow definition also

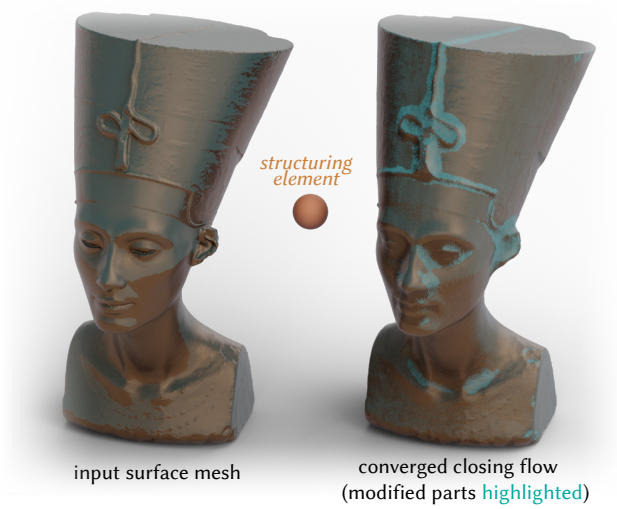


Fig. 4. Our method, when ran on a fine mesh of a copper statue, can be used to simulate the accumulation of blue rust on concave regions (moved by our method and highlighted in blue). 3D model released by Jan Nikolai Nelles and Nora Al-Badri under CC BY-NC-SA 4.0.

enables unique generalizations and even new operations previously unconsidered in mathematical morphology.

To demonstrate the effectiveness of our method we show its success on a variety of examples. We compare to standard volumetric morphology methods and existing curvature flows. Our closing flow guarantees that the output surface contains the input shape, suggesting its use for designing conservative hulls or cages. Due to our variational formulation, we can add additional problem specific objective functions and constraints, as well as spatially varying structuring element sizes. Finally, we show possible applications to various domains including geometric design for fabrication, molding, smoothing and surface weathering (see Fig. 4).

2 BACKGROUND & RELATED WORK

Our work bridges two research topics with vast literatures: mathematical morphology and surface flows. We focus this section on establishing sufficient background for each and providing context with previous works related to ours in methodology or applications.

2.1 Mathematical Morphology

Mathematical morphology applies the concepts of "growth" and "shrinkage" to analyze and filter a geometric shape, for example a solid region $A \subset \mathbb{R}^2$ of a 2D image [Serra 1969]. The fundamental building blocks are dilation (growth) and erosion (shrinkage) *by* a solid structuring element $\sigma \subset \mathbb{R}^2$. They are defined in terms Minkowski addition ($\oplus$) and subtraction ($\ominus$):

$$\text{dilation}(A, \sigma) = (A \oplus \sigma) = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in \sigma\}, \quad (1)$$

$$\text{erosion}(A, \sigma) = (A \ominus \sigma) = (A^c \oplus \sigma)^c, \quad (2)$$

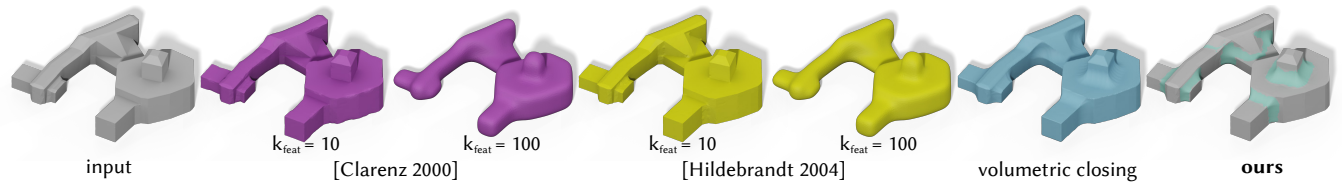


Fig. 5. Despite our method's formulation resembling the PDEs used in anisotropic surface diffusion, these works are not designed to and do not succeed at replicating the closing operation regardless of parameter choice. 3D model by Walnut Grove Secondary under CC BY 4.0.

where A^c indicates the set complement of A. The *opening* and *closing* operations are then defined and typically (if not always) implemented in terms of dilation and erosion:

$$\text{opening}(A, \sigma) = \text{dilation}(\text{erosion}(A, \sigma), \sigma) = ((A \ominus \sigma) \oplus \sigma) \quad (3)$$

$$\text{closing}(A, \sigma) = \text{erosion}(\text{dilation}(A, \sigma), \sigma) = ((A \oplus \sigma) \ominus \sigma) \quad (4)$$

The applications of these operations include not just low-level tasks such as denoising, but also high-level tasks such as segmentation.

Mathematical morphology theory is rich with generality. Its classic definition for binary occupancy images has been extended to grayscale images (e.g., intensity or density images) [Sternberg 1986]. In this paper, we do not consider grayscale morphology. The definitions above and larger theory are not limited to discrete images, two-dimensional shapes, or even Euclidean spaces. In this paper, we restrict our consideration to 2D and 3D solid shapes defined by piecewise-linear boundary curves and surfaces, respectively. While morphology theory encompasses arbitrarily shaped structuring elements σ (e.g., squares, diamonds, pyramids), we only consider balls (σ_r is the solid sphere with radius r).

In solid geometry processing, mathematical morphology operations have seen great success. Nooruddin and Turk [2003] voxelize an input triangle mesh, conduct a closing operation, convert back to a mesh, and apply simplification for mesh repair. Indeed, the closing operation is a common cleanup and analysis tool, more recently employed for preprocessing 3D printing shapes [Telea and Jalba 2011]. In fabrication and manufacturing, openings and closings have been used to design safe tool paths (e.g., [Hornus and Lefebvre 2017]).

Erosion and dilation are straightfoward to implement if both the input shape A and the structuring element σ are discretized as binary occupancy on a voxel grid. Utilizing sparse data structures [Calderon and Boubekeur 2017; Museth 2013] or layered depth images [Chen et al. 2018; Wang and Manocha 2013], voxel morphology operations can be made lightning fast. Jones and Satherley [2001] improved the accuracy of composite morphology operations like openings and closings by computing sub-voxel level sets of offset distances. Voxel-based approaches suffer from the standard limitation of introducing aliasing artifacts if the input shape is not already voxelized (see Fig. 3). Calderon and Boubekeur [2014] avoid voxelization by conducting dilation and erosion (and by composition openings and closings) on input point clouds. If the input is not already a point cloud, this method would introduce sampling bias and approximation error. Our goal is to avoid lossy data conversions.

Erosion and dilation can be implemented directly on boundary representations of solid shapes (e.g., triangle meshes) as Minkowski sums or offset surfaces [Barki et al. 2009; Campen and Kobbelt 2010; Zhou et al. 2016]. Implementing opening and closing as compositions of dilations and erosions this way leads to re-discretization of the entire shape including parts that do not change their underlying geometry (see Fig. 1). Our goal is output-sensitivity: avoid changing the mesh in regions that do not move.

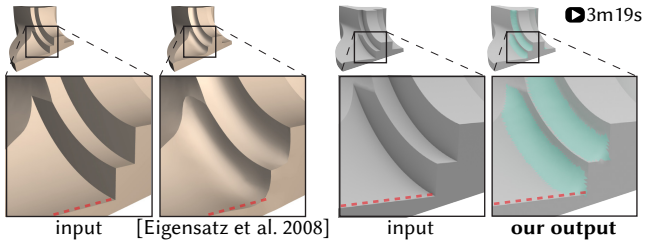
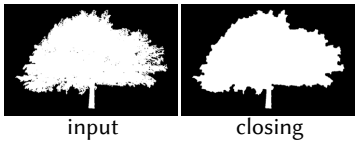


Fig. 6. Previous attempts at curvature clamping, like [Eigensatz et al. 2008], are successful but do not present other theoretical guarantees of closing, like output fully containing input (see red line, which exposes their output growing inward).

We are not the first work to consider the connection between morphology and partial differential equations (PDEs). Sapiro et al. [1993] formulate a PDE for describing the motion of a curve undergoing dilation (or erosion) and inevitably treats the curve *implicitly* as a levelset. Utilizing the fact that the solution to the non-linear Eikonal PDE is the signed distance function, Museth et al. [2002] conduct erosions and dilations of shapes stored as implicit level-set surfaces. Like the previously mentioned works, openings and closings are then defined via composition. Guichard et al. [2005] conducts the closing operation on level-sets in 2D via a time-switching PDE (i.e., dilation for $0 < t \leq 1/2$ and erosion for $1/2 < t \leq 1$), effectively integrating PDEs in sequence. Maragos and Vachier [2008] extend this idea to grayscale morphology. In contrast to these works, we operate on *explicit* surface meshes and define partial differential equations to directly compute closings and openings, without requiring a background grid or composing erosions and dilations.



One benefit of volumetric methods and composition-based methods in general is topological change: indeed, many previously explored applications revolve around topological denoising or simplification (see inset). While our proposed flow can produce *large* changes (see Fig. 8), it will not change the topology of the input. We argue this is neither good nor bad, but an important difference between our definition of opening and closing and the traditional, volumetric one.

Fig. 7. A closing of a polygon is used as a toolpath to cut a maple leaf from plywood using a 3-axis CNC mill.

We analyze this difference in detail and explore the consequences of avoiding the extrinsic or topological effects of the traditional definitions further in Section 4.

A key observation is that the output surfaces of the closing and opening operations have *bounded* curvature. Eigensatz et al. [2008] also produce surfaces with bounded curvature, although not in the context of mathematical morphology. Fig. 6 shows that these do not correspond to closings.

2.2 Surface Flows

Surface flows are a fundamental subroutine in geometric modeling [Brakke 1992]. Perhaps the most well-understood flow is mean-curvature flow. If we let Ω be a two-dimensional manifold, then we may define $\mathcal{S}^t : \Omega \to \mathbb{R}^3$ to be a smooth family of immersions that follow a velocity:

$$\frac{\partial \mathcal{S}^t}{\partial t} = -H\hat{\mathbf{n}} \tag{5}$$

where H is the mean curvature and $\hat{\mathbf{n}}$ is the outward unit surface normal. Because the Laplacian of the surface immersion is equal to the mean-curvature normal ($\Delta_t \mathcal{S}^t = H\hat{\mathbf{n}}$), this is also referred to as Laplacian smoothing. Following the Euler-Lagrange formulation, the same equation appears as the gradient flow of total surface area or equivalently Dirichlet energy of the immersion function:

$$\frac{\partial \mathcal{S}^t}{\partial t} = -\frac{\partial}{\partial \mathcal{S}^t} \int_\Omega \|\nabla \mathcal{S}^t\|^2 dA. \tag{6}$$

This flow and its cousins have been used in triangle-mesh processing for surface fairing [Bobenko and Schröder 2005; Taubin 1995]. This energy formulation affords additional design considerations such as volume preservation [Desbrun et al. 1999]. We will similarly enjoy this flexibility to add additional constraints and objectives.

Desbrun et al. also introduced the idea of using implicit time integration to improve stability. Implemented using the finite-element method on a triangle mesh, numerical issues can still occur (even before singularities in the smooth flow) due to poorly shaped evolving triangles. Prior works (e.g., [Crane et al. 2013b; Kazhdan et al. 2012]) alter the energy in Equation (6) to ensure conformal mappings and thus maintain good surface triangulation (without remeshing).

Anisotropic surface diffusion includes a diffusion tensor which multiplies $\nabla \mathcal{S}^t$ in Equation (6). The different forms for this tensor (e.g., [Clarenz et al. 2000; Hildebrandt and Polthier 2004]) depend on the *magnitudes* of the principal curvatures at each point of the shape (see Fig. 5) and on a curvature threshold parameter k_{feat}. The flagship application is feature-preserving denoising: regions with $|k| > k_{\text{feat}}$ are read as features and not smoothed; all others are. Unlike our flow, this formulation does not replicate closing (see Fig. 5), the morphological operation which smooths *high* absolute curvature regions and depends fundamentally on their *sign*.

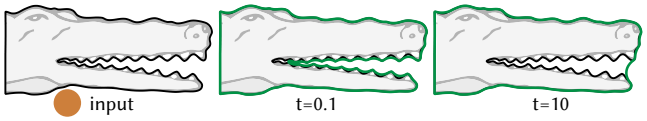
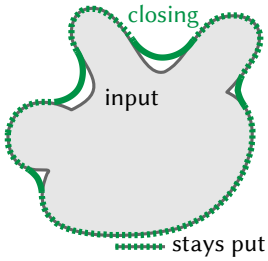


Fig. 8. Small concavities get filled in first, then stop moving, but eventually may get pulled out.

Further manipulating the mean-curvature flow, it is possible to derive more specialized flows that produce cubic stylizations [Liu and Jacobson 2019] or developable surfaces [Stein et al. 2018a]. Past works have explored the connection of curvature flows and medial-axis extraction [Tagliasacchi et al. 2016]; in particular, Au et al. [2008] fix vertices during a discrete flow in a manner similar to our algorithm at a high level, but in their case to approximate a curve skeleton inside the shape. Recent works that alter the *norm* in Equation (6) demonstrate sparsity-inducing or edge-preserving smoothing [He and Schaefer 2013; Stein et al. 2018b; Zhang et al. 2018]. Our derivation also begins with the Dirichlet energy, but we alter it such that the flow approaches the localized closing of the shape rather than smoothing or shrinking the entire shape. We utilize an adaptive remesher to ensure stability and accuracy in the regions that require movement.

3 METHOD

In this section we will write only about the closing operation. Equivalent statements and derivations corresponding to the opening can be constructed by exchanging the roles of convex and concave regions in what follows.



The observation core to the development of our method is that — in general – when conducting the closing operation on a shape, many parts of the surface will stay put (see inset).

For simplicity, we begin by considering a non-convex region in 2D $\mathcal{A} \subset \mathbb{R}^2$ with a sufficiently smooth (e.g., G^2) simple boundary curve $\mathcal{S} = \partial\mathcal{A}$. If the initial boundary $\mathcal{S}$ has minimum signed curvature $k_{\min} < 0$, then we can consider a *differential closing* by a disk σ_r of radius $r = -1/(k_{\min} + \varepsilon)$ with $\varepsilon > 0$ where the surface after closing remains sufficiently smooth (e.g, G^1). We observe a few key properties:

(1) The input shape is contained in its closing: $\mathcal{A} \subset \text{closing}(\mathcal{A}, \sigma_r)$.
(2) Changes in shape will be isolated *near* points on the input boundary with curvature $\leq k_{\min} + \varepsilon$.
(3) The minimum curvature on the closing boundary is now $-1/r$.

These key properties suggest to us a boundary flow for a given structuring ball with radius r that will satisfy the following:

(1) It will only ever move the shape's boundary *outward* (i.e., in the normal direction)

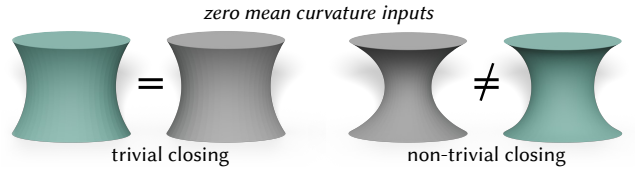


Fig. 9. Whether a point moves during closing cannot be determined by mean curvature.

(2) It will only affect regions of this boundary non-trivially if their curvature is smaller than $-1/r$
(3) The flow will only stop when the boundary's curvature is everywhere bounded from below by $-1/r$.

All criteria now refer to the input shape's boundary without explicit reference to the enclosed solid region. Unlike all previous closing definitions, we do not rely on composing erosion with dilation.

Our contribution is to model this *closing flow* as a modified Dirichlet energy gradient flow, discretizing it in time with a linearly semi-implicit Euler integration and discretizing it in space using triangle meshes. We will demonstrate that this flow retains good properties of the traditional closing operation even when r is much larger than the input surface's minimum curvature radius $-1/k_{\min}$. We continue our discussion first with curves in 2D, for which curvature is simpler, and then consider the more interesting case of surfaces embedded in 3D space.

3.1 Curves

For curves in 2D, our flow for closing by a disk with radius r has the form

$$\frac{\partial \mathcal{S}^t}{\partial t} = \begin{cases} -k\hat{\mathbf{n}} & \text{if } k < -\frac{1}{r} \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{7}$$

where k is the signed curvature and $\hat{\mathbf{n}} \in \mathbb{R}^2$ is the consistently oriented outward unit normal.

This flow has a discontinuous right-hand side, putting it in the family of *differential inclusions* (see e.g., [Kunze 2000]). This equation shares qualities of other problems in computer graphics, such as collision handling and Coulomb friction in physically based simulation [Stewart 2011]. Indeed, we can conceptually think of this flow as following the traditional curvature flow (2D analog of Equation (5)) in regions with very negative curvature until *colliding* with the curvature-bound *obstacle*. It may seem natural to propose a speed proportional to *the difference* between k and our bound (i.e., $-(k + 1/r)\hat{\mathbf{n}}$ instead of $-k\hat{\mathbf{n}}$). This would lead to a Zeno's paradox where the closing would only be reached at $t \to \infty$. We avoid this by making the speed proportional to curvature itself.

Fig. 10 shows the complex boundary of Vietnam undergoing a closing flow. The initial curve has interesting positive and negative curvature, but as the flow progresses the curve moves outward in regions with curvature less than the structuring element until all movement stops. The final output curve matches the reference closing computed using standard dilation and erosion on a dense grid. We point out that the active set of regions with zero velocity changes over time. Intuitively one region of the curve may grow outward, then freeze, and then later peel away as another moving region encroaches (see Fig. 8 and accompanying video).

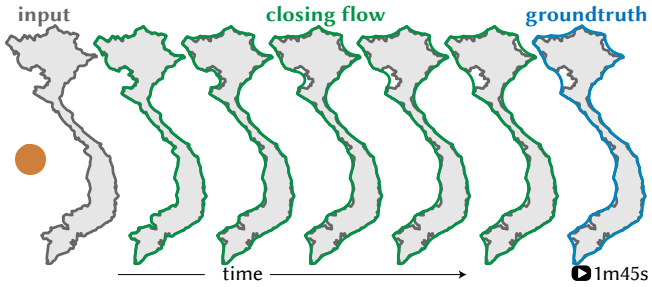
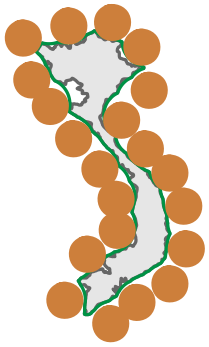


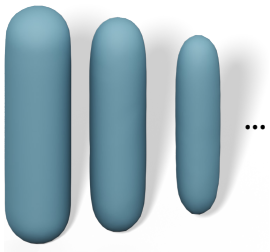
Fig. 10. Our curvature-bounded curvature flow works to replicate the effects of morphological closing in the plane. The converged output of our method (right, in green) is undistinguishable from a very finely voxelized computation of it, which we treat as the groundtruth (blue).

We will defer discussion of temporal and spatial discretization until Section 3.3. In two dimensions, the boundary of the closing contains regions from the input boundary that never moved and arcs of radius matching the structuring element. This corresponds to the intuitive definition of the closing as the complement of the union of all translations of the structuring element which don't overlap with the input shape (see inset). The arcs emerge by construction during the flow as curvature monotonically increases until reaching the bound. On surfaces of solids in three-dimensions, curvature exists in any tangent direction and care will be needed to generalize this intuition and consequently our proposed flow.



3.2 Surfaces

The typical analog of the length-shortening curvature flow for curves in 2D is the area-shrinking mean curvature flow for surfaces in 3D (Equation (5)). Mean-curvature flow might appear to be a good candidate for closing and opening operations. This approach turns out to be easy to foil, since mean curvature does not strictly correspond to convexity/concavity. Consider a (convex) capsule whose closing is trivial. Mean-curvature is positive so the entire shape shrinks inward (see inset). Conversely, consider a catenoid (revolved catenary) which has zero mean curvature and thus will not move during mean curvature flow. Depending on the size of the structuring element, a catenoid may have a non-trivial closing (see Fig. 9). Similar counterexamples can be constructed for other standard flows like Gaussian-curvature flow, Willmore flow, etc. (see Fig. 11 and accompanying video). So, when should the surface move? Or just as important, when should the surface stay put?



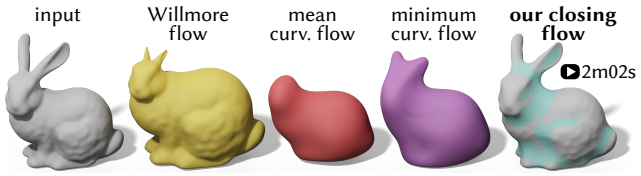


Fig. 11. Other flows such as Willmore (e.g., [Crane et al. 2013b]) or classic mean curvature do not behave like a closing. Minimum curvature flow, upon which our closing flow is based, moves everything and may shrink the shape in some regions. Our use of a curvature-based *obstacle* ensures outward flow toward the closing. See accompanying video for an animated version.

The intuition for curves in 2D was that the flow stops when reaching an arc of the given structuring element's radius. This arc hugs the disk that fits into the shape's complement space ($\mathbb{R}^2 \setminus \mathcal{A}$). In 3D, we should consider (spherical) ball structuring elements in the complement space and make sure the flow stops once we can just fit a ball against the surface. Neither mean curvature nor Gaussian curvature alone have enough information to tell us this. A ball can fit against the surface if the signed normal curvature radius in any direction is greater than the ball radius. We need to look at the *minimum* principal curvature.

Our proposed flow moves the surface in regions where a ball would not fit in the normal direction at a rate proportional to minimum curvature:

$$\frac{\partial \mathcal{S}^t}{\partial t} = \begin{cases} -k_2 \hat{\mathbf{n}} & \text{if } k_2 < -1/r, \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{8}$$

where k_2 is the minimum (signed) principal curvature (i.e., $k_2 \leq k_1$, see, e.g., [O'Neill 1966]). Analagous to the lower-dimensional case (see Section 3.1), this flow has a discontinuous right-hand side that ensures that the flow stops once an admissible curvature is reached. The speed of the flow is proportional to minimum curvature ensuring that only the regions where a ball would not fit move and continue moving without slowing down. This ensures that saddles (as in the case of the catenoid) will move at points where the structuring element will not *fit* against the surface. Parts of the surface that exceed the given bound stay put (see Fig. 13).

Fig. 11 demonstrates a comparison of mean curvature flow ($\partial \mathcal{S}^t/\partial t = -H\hat{\mathbf{n}}$), minimum-curvature flow ($\partial \mathcal{S}^t/\partial t = -k_2\hat{\mathbf{n}}$, without the case statement) and our proposed flow in Equation (8). Mean curvature flow shrinks inward, while minimum curvature flow also shrinks in some areas while expanding outward in concave areas. Our flow is strictly outward (by construction the regions that move have $-k_2 > 0$). The minimum-curvature flow and our closing flow should not be confused for the discrete flow of Stein et al. [2018a] which produces piecewise-developable surfaces (inspired by $\frac{\partial \mathcal{S}^t}{\partial t} = \frac{\partial}{\partial \mathcal{S}^t} \int \kappa_2^2$, where $|\kappa_2| \leq |\kappa_1|$ are *unsigned* principal curvatures). In Fig. 12, we use the same case statement condition "... if $k_2 < -1/r$, $\mathbf{0}$ otherwise", but compare setting the rate of normal motion proportional to mean curvature and minimum curvature. Despite the obstacle, mean curvature flow monotonically decreases surface area (recall, it is the gradient of surface area). Our proposed flow smooths the surface but toward the closing which in this case increases surface area.

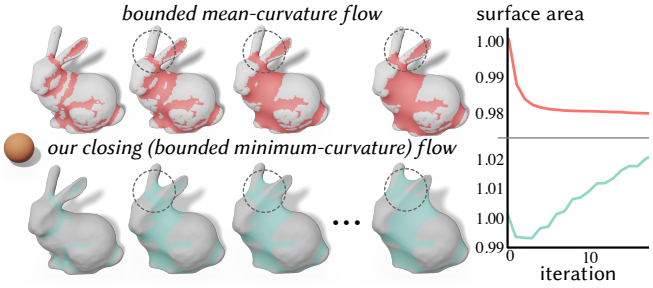


Fig. 12. Even with a minimum-curvature based bound, mean curvature flow (top row) will not manage to accurately replicate the effects of closing (see the bunny's ears, highlighted), since it will never allow for an increase in surface area (see top right). Our principal curvature flow, however, does allow for an increase in surface area if that means reducing the concaveness of the shape, accurately replicating closing.

3.3 Discretization

The differential equations in Equations (7) and (8) are non-linear and involve discontinuities. We discretize and integrate this equation by leveraging advances made over time by different communities within and beyond computer graphics and geometry processing.

Time. We discretize time by separating the minimum-curvature flow $\frac{\partial \mathcal{S}^t}{\partial t} = -k_2\hat{\mathbf{n}}$ and the discontinuous obstacle in the right-hand side of Equation (8) into linearly implicit and explicit terms, respectively. Just as mean curvature flow can be written as the gradient of Dirichlet energy (see Equation (6)), minimum-curvature flow can be derived as the minimum of a modified Dirichlet energy.

Minimum curvature is the second derivative of the immersion function (see [O'Neill 1966]) in the (unit) direction of minimum curvature ($\hat{\mathbf{d}}_2$):

$$\frac{\partial \mathcal{S}^t}{\partial t} = -k_2(t)\hat{\mathbf{n}} \tag{9}$$

$$= \nabla(\nabla \mathcal{S}^t \cdot \hat{\mathbf{d}}_2(t)). \tag{10}$$

Neglecting the higher order terms resulting from $\hat{\mathbf{d}}_2$'s dependence on time, we can make $\hat{\mathbf{d}}_2(t) \approx \hat{\mathbf{d}}_2$ and apply Green's Identity:

$$\frac{\partial \mathcal{S}^t}{\partial t} \approx \frac{\partial}{\partial \mathcal{S}^t} \int_\Omega \mathcal{S}^t \cdot \nabla(\nabla \mathcal{S}^t \cdot \hat{\mathbf{d}}_2) dA \tag{11}$$

$$= -\frac{\partial}{\partial \mathcal{S}^t} \int_\Omega \left(\nabla \mathcal{S}^t \cdot \hat{\mathbf{d}}_2\right)^2 dA, \tag{12}$$

where $\cdot$ represents matrix multiplication and Ω is the parametrization domain. Similar to the Dirichlet energy, the integral measures the gradient of the embedding function, but only considers the component in the minimum curvature direction. This variational form allows us to directly apply the linearly implicit integration employed by Desbrun et al. [1999] and others. Consider a finite difference with a timestep value of τ, our equation becomes

$$\frac{\mathcal{S}^{t+1} - \mathcal{S}^t}{\tau} = -\frac{\partial}{\partial \mathcal{S}^{t+1}} \int_\Omega (\nabla^t \mathcal{S}^{t+1} \cdot \hat{\mathbf{d}}_2^t)^2 dA \tag{13}$$

$$0 = -\frac{\partial}{\partial \mathcal{S}^{t+1}} \int_\Omega \tau(\nabla^t \mathcal{S}^{t+1} \cdot \hat{\mathbf{d}}_2^t)^2 + \frac{1}{2}(\mathcal{S}^{t+1} - \mathcal{S}^t)^2 dA,$$

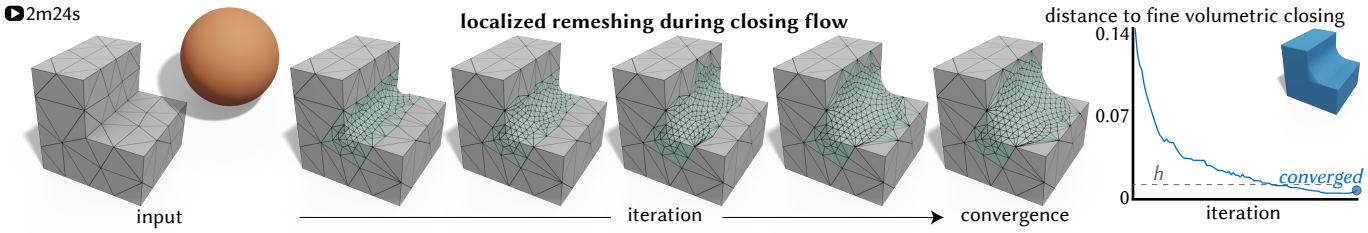


Fig. 13. At each iteration of our flow, we remesh only the parts of the shape that are moving with the flow, leaving all completely convex regions untouched. Our method converges to a fine, volumetric closing within half of our chosen edge-length h.

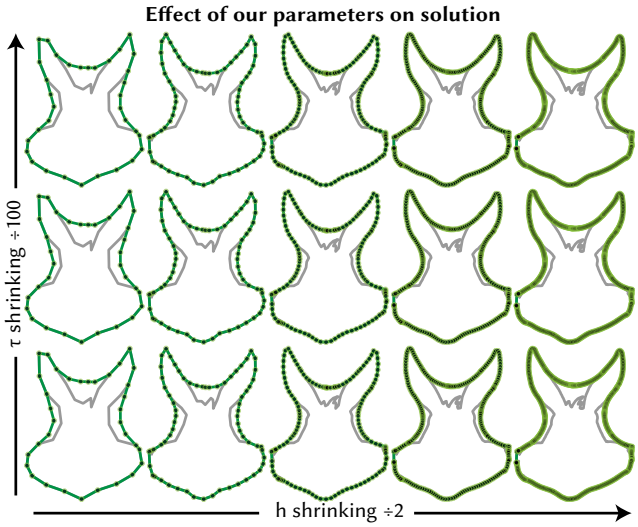


Fig. 14. Our method's main parameters are the edge length h and the timestep τ. Our method is stable to large changes in timestep, and a smaller edge length simply means approximating the analytic surface better, albeit at the cost of more iterations and longer-running ones.

where we call this a *linearly* implicit integration because ∇^t and $\mathbf{d}^t$ are defined by the explicitly known immersion at the beginning of the time step $\mathcal{S}^t$ (mirroring [Desbrun et al. 1999]). This equation corresponds to the Euler-Lagrange optimality conditions of minimizing the integral:

$$\min_{\mathcal{S}^{t+1}} \int_\Omega \tau(\nabla^t \mathcal{S}^{t+1} \cdot \hat{\mathbf{d}}_2^t)^2 + \frac{1}{2}(\mathcal{S}^{t+1} - \mathcal{S}^t)^2 dA. \tag{14}$$

The objective is quadratic in $\mathcal{S}^{t+1}$ and may be solved directly. As is, this flow follows minimum curvature as shown in Fig. 11.

To incorporate the obstacle, we *freeze* the flow for a time step if the current curvature exceeds the specified bound. In practice, while we are computing the minimum curvature directions $\hat{\mathbf{d}}_2^t$ we collect minimum curvature values k_2^t. Adding this constraint to our implicit integration above, our time integration can be written as a value-constrained quadratic optimization:

$$\min_{\mathcal{S}^{t+1}} \int_\Omega \tau(\nabla^t \mathcal{S}^{t+1} \cdot \hat{\mathbf{d}}_2^t)^2 + \frac{1}{2}(\mathcal{S}^{t+1} - \mathcal{S}^t)^2 dA \tag{15}$$

$$\text{subject to: } \mathcal{S}^{t+1} = \mathcal{S}^t\big|_{k_2^t \geq -1/r} \tag{16}$$

Space. We assume that the input to each time step of our method is a manifold triangle mesh with n vertices and m faces bounding a solid region of $\mathbb{R}^3$ (a triangulated polyhedron in the terminology of [Zhou et al. 2016]). We experimented with various methods for computing curvature (see, e.g., [Crane et al. 2013a]). Finally, we compute the minimum curvature k_i at the ith vertex via the mean curvature (H_i), Gaussian curvature (K_i), and the associated mass M_i, using the discretizations in [Sullivan 2008] (Sec. 4.4):

$$k_i = \frac{H_i - \sqrt{H_i^2 - M_i K_i}}{M_i} \tag{17}$$

$$H_i = \frac{1}{2}\sum_{i,j} \ell_{ij}\varphi_{ij}, \quad K_i = 2\pi - \sum_{i,j,k} \theta_{jik}, \quad M_i = \frac{1}{3}\sum_{i,j,k} A_{ijk} \tag{18}$$

where $\sum_{i,j}$ and $\sum_{i,j,k}$ sum over edges and triangles incident on vertex i respectively, ℓ_{ij} is the edge length, φ_{ij} is the dihedral angle, θ_{jik} is the internal angle at vertex i, and A_{ijk} is the triangle area. We compute the (unit) minimum curvature direction $\hat{\mathbf{d}}_f \in \mathbb{R}^3$ at the fth triangle via quadratic fitting using the six vertex positions of the corners of the triangle and its flap neighbors according to the method of Cazals and Pouget [2005].

With these in hand, we discretize the components of the minimization in Equation (15) using the typical piecewise-linear finite element discretization of the gradient operator $\mathbf{G} \in \mathbb{R}^{3m \times n}$ and mass matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ (see, e.g., [Meyer et al. 2003]). Given initial mesh vertex positions in the rows of a matrix $\mathbf{V}^t \in \mathbb{R}^{n \times 3}$, the positions at the next time step solve the discrete optimization:

$$\min_{\mathbf{V}^{t+1}} \frac{1}{2}\operatorname{tr}\left(\tau \mathbf{V}^{t+1\top}\mathbf{G}^\top\mathbf{D}^\top\mathbf{A}\mathbf{D}\mathbf{G}\mathbf{V}^{t+1} + (\mathbf{V}^{t+1} - \mathbf{V}^t)^\top\mathbf{M}(\mathbf{V}^{t+1} - \mathbf{V}^t)\right)$$

$$\text{subject to: } \mathbf{V}_i^{t+1} = \mathbf{V}_i^t \;\; \forall i \text{ such that } k_i \geq -1/r \tag{19}$$

where $\mathbf{D} \in \mathbb{R}^{m \times 3m}$ computes the per-face dot product with the computed curvature directions and $\mathbf{A} \in \mathbb{R}^{m \times m}$ is a diagonal matrix of triangle areas.

In the case of 2D curves, like in Figs. 10 and 8, we assume they are given as a polyline of vertices $\mathbf{P}$ and curvature is discretized as the norm of $\mathbf{LP}$, where $\mathbf{L}$ is the finite difference Laplacian.

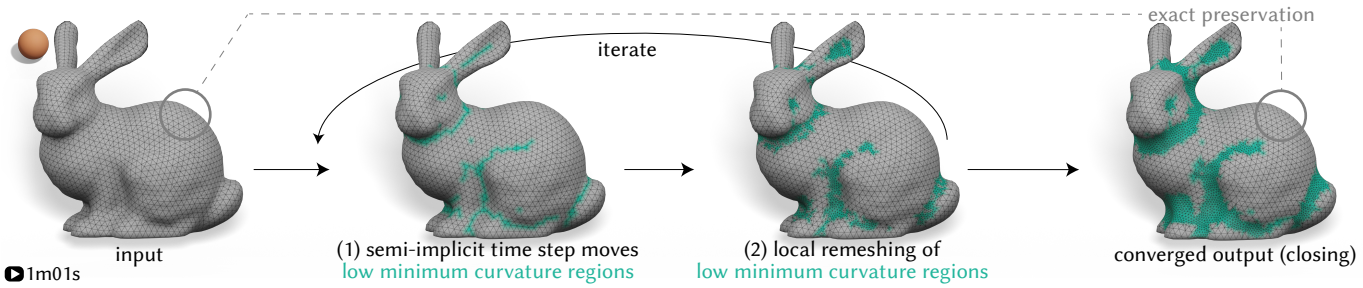


Fig. 15. Our method iteratively flows the surface according to a semi-implicit time integration and remeshes near moving regions.

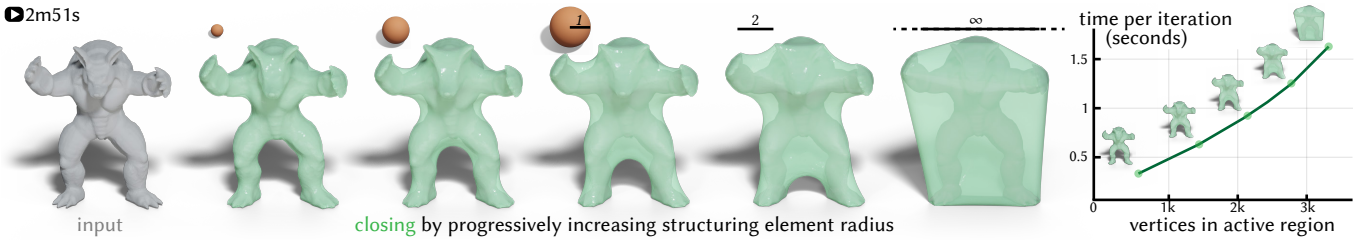


Fig. 16. The closing flows to a conservative hull for any radius r. As the structuring element radius is increase the resulting closing approaches the convex hull. The size of the moving region is larger as the structuring element grows. Since our method is output-sensitive, so grows the runtime per iteration. 3D model by the Stanford 3D Scanning Repository.

Remeshing. As a precondition, we expect the input triangle mesh at each time step optimization to have reasonable mesh quality both in terms of the finite-element matrices and the curvature estimations. On the other hand, our promise is to avoid changing the mesh away from parts of the shape that actually move during closing. Inspired by the success of recent surface-tracking and flow methods [Brochu and Bridson 2009; Da et al. 2014; Stein et al. 2018a], we propose an adaptive remeshing approach parameterized by a target edge-length h, provided as input. In cases where the input is extremely coarse, before beginning the flow, we recursively apply midpoint subdivision (in plane) to all triangles with any edge longer than $2h$. We optionally keep track of associated parent triangles in the input mesh, so we can re-identify sets of triangles that do not move. After each time-step, we run 10 iterations of isotropic remeshing [Botsch et al. 2010] modified to *protect* any edges incident on any vertex with curvature values greater than the given bound. Fig. 13 shows our adaptive remeshing as the flow progresses: convex regions far from the action remain untouched.

Putting our discretizations and remesher together in a loop we have our complete flow algorithm (see Fig. 15). Time integration stops when all vertices are fixed (outside curvature bound) or a numerical displacement tolerance is reached (for which we check every ten flow iterations). A further step can be taken, however, to significantly improve its wall-clock and asymptotic performance.

Output-sensitive implementation. Our key *theoretical* observation has been that large regions of a given shape remain identical after performing a closing operation. This observation can be further exploited to make our method output-sensitive, in a way that its performance depends on the size of the moving regions and not of the whole triangle mesh. To do this in practice, on our first iteration, we divide our mesh into a (generally disconnected) *active* submesh (the two-ring neighborhood of every vertex with $k_i \leq -1/r$) and an *inactive* one (all other vertices). In this way, every computation described in this section may be performed only on the active mesh. At the end of each iteration, the discrete curvatures of every non-boundary vertex of the active mesh are re-calculated, and vertices are added or removed from it so that it remains the two-ring neighborhood of all moving vertices.

As an example of the effect of this output-sensitivity, consider Fig. 4, where the input mesh contains 100k vertices. A naive implementation of our method that visits the entire mesh on each iteration takes 160 seconds. However, the size of the *active* regions is on average under 10k vertices, and our output-sensitive implementation converges in 21 seconds. To make this point clearer, we have chosen to render most of our results using green to highlight the parts of the shape that have been in the active region for at least one iteration, and grey for those that have always been inactive and therefore remain identical to the input.

4 EXPERIMENTS & RESULTS

We have implemented our main prototype in MATLAB (with GPTOOLBOX [Jacobson et al. 2016]), using C++ (with LIBIGL [Jacobson et al. 2018]) to implement the adaptive remesher and the (parallelized) curvature fitting steps. As shown empirically in Fig. 16, our code achieves the correct asymptotic behavior $O(\tilde{n}^{1+f}k)$ where $\tilde{n}$ is the number of active vertices, $f > 0$ accounts for the Laplacian-like sparse system solve (we refactor each iteration due to remeshing; low-rank update could be a possible performance optimization) and

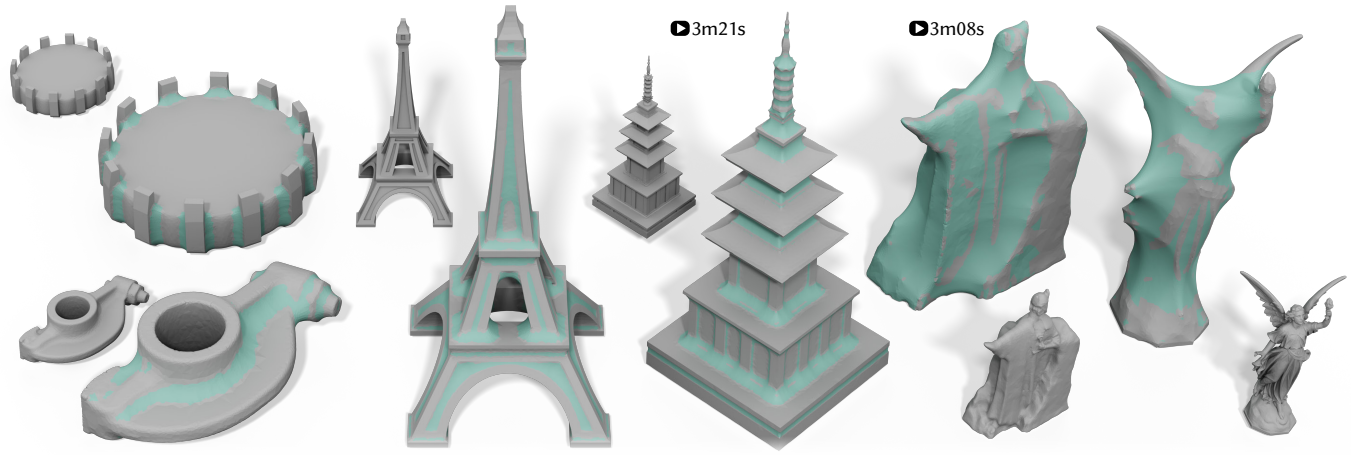


Fig. 17. Our closing flow can be used on inputs from a diverse set of origins, from machine parts (left) to architectural models (center) and statues (right). See accompanying video for animations. 3D models (top to bottom, left to right) by Van Alles Wat Ontwerp under CC BY-NC 4.0, AIM2SHAPE Mesh Repository, Ian Bunker under CC BY 4.0, Shim JinYoung under CC BY 4.0, Patrick Bentley under CC BY 4.0 and the Stanford 3D Scanning Repository.

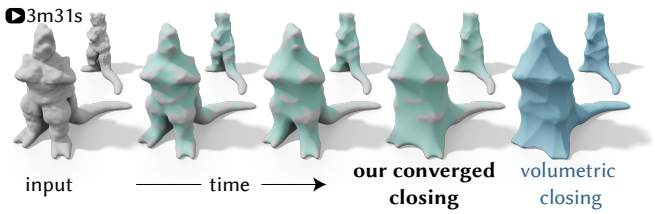


Fig. 18. Our flow accurately approximates the effects of closing in complex shapes with large structuring elements too, seamlessly merging the extremities of this model. See accompanying video for complete flow animation. 3D model by MakerBot under CC BY 4.0.

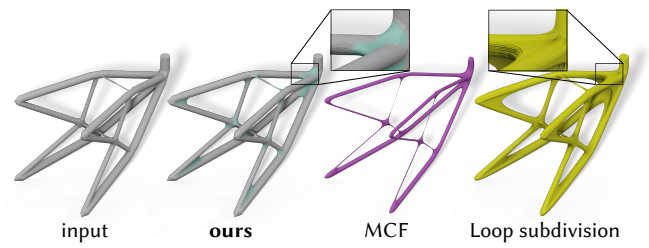


Fig. 19. Our closing flow can be used to smooth a complex rod structure, without presenting the singularities that characterize Mean Curvature Flow (center right) or the extreme mesh-dependency from Loop subidivision based smoothing as suggested by [Hart 2008].

k is the number of iterations until convergence. We report timings for a representative set of results from this paper in Table 1, as conducted on our machine with Intel Xeon CPU E5-2637 v3 @ 3.50 Hz (16 cores), Nvidia GTX 970 and 64 GB of RAM.

Besides the input mesh and structuring element radius (r), the main parameters to our method are the discrete time step (τ) and the target edge length (h). In our examples, where the inputs are normalized to tightly fit the unit cube, we use $\tau = 0.1$ and $h = \pi r/20$,

Table 1. Runtimes reported for a representative sample of results.

Model	Figure	Vertices	Iterations	Runtime
Ring	24	13k	30	2.4 s
Rocker	17	11k	30	10 s
Hook	19	14k	90	11 s
Gear	17	15k	40	13 s
Nefertiti	4	100k	40	21 s
Zipper	26	17k	70	25 s
Ice tray	23	15k	60	32 s
Eiffel	17	26k	120	59 s
Seokgatap	17	40k	140	89 s
Argonath	17	25k	220	212 s
Lucy	17	50k	180	302 s

though optimal values are dependent on the input and the desired output. We experimentally confirm that our method exhibits the time-step stability associated with implicit schemes (see Fig. 14), and that the number of iterations until convergence is affected by τ and h. The latter, which tends to dominate in our examples, is because the (combinatorially local) way in which we calculate curvature makes it so the size of our flow's moving region can only ever grow by a one-ring neighborhood per iteration, effectively setting a lower bound for k which depends inversely on h.

In Fig. 16, we show *converged* results for closing flows on the Armadillo for increasing radii. In this limit of increasing radii, the morphological closing tends toward the convex hull. We witness this extreme behavior in our flow as well.

In Fig. 7, we show a simple application using our flow to move a polygon to its rounded closing. This curve is in turn used as the guiding path for a 3-axis CNC-mill to cut a piece of plywood.

In Fig. 4, the closing flow for a small radius approximates the accumulation of dirt and grime during weather of a bronze statue. Our flow naturally *selects* the region so that coloring it differently

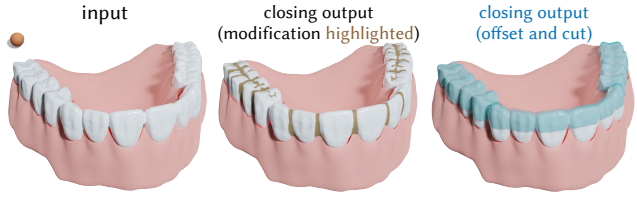


Fig. 20. Our closing flow is ran on a mesh of a lower human denture, highlighting with a "plaque" texture the regions that are moved outward by our flow and simulating the buildup. The same result is slightly offset in the normal direction and cut with a horizontal plane to create smooth yet tight fitting "braces." 3D mesh by Garrett Boughton under CC BY-NC 4.0.

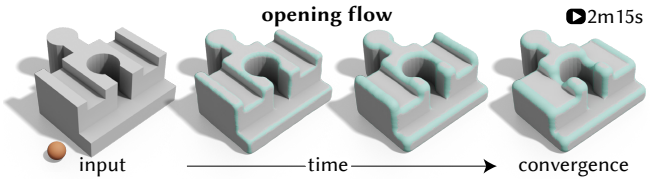


Fig. 21. The opening flow complements our closing flow. Convex corners get rounded out and the shape flows inward. See accompanying video for complete flow animation. 3D model by David Wilson under CC BY-SA 3.0.

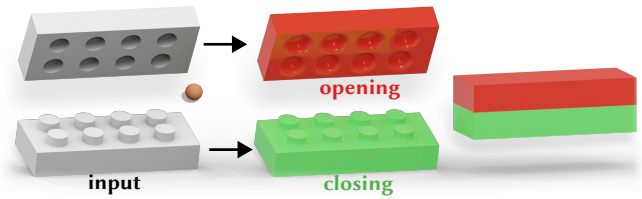


Fig. 22. Our opening and closing flows are the dual of one another. Snugly interlocking input blocks maintain their tight fit after *selectively* flowing just pegs and holes.

is trivial. The structuring element radius r is the primary parameter of our flow. Using a larger radius r should not be confused with running the flow for a longer period of time. Our flow is finite and runs until convergence (see Fig. 18) on a wide variety of shapes from various different origins (see Fig. 17). In Fig. 20, the same output of our closing flow is used to both model the build-up of plaque on a real molded human denture and to construct a set of invisible braces that fit tightly to the teeth. In Fig. 19, a rod structure is smoothed by our closing into a singularity-free output.

So far we have considered the closing flow, in Fig. 21 we flow an input shape toward its *opening*. This inward flow ensures that the result is contained inside the input.

Mirroring the traditional view of opening and closing operations, the results of our opening and closing flows are duals. In Fig. 22, we take advantage of our variational formulation to *selectively* apply an opening and closing to complementary parts of two interlocking blocks. A similar experiment is carried out in Fig. 23, where the closing of a tray results in the opening of the produced ice blocks.

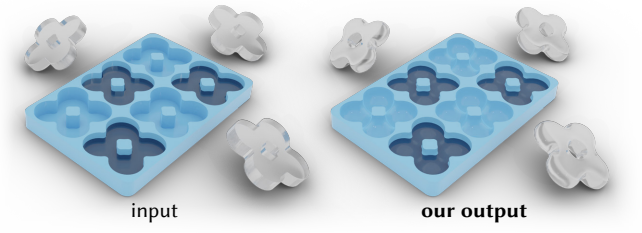


Fig. 23. In this molding-inspired example, we perform the closing of an ice tray and the opening of the produced ice using our defined flows. 3D model by Walter Hsiao under CC BY-SA 3.0.

Fig. 24. Our flow can be used to simulate the filth accumulating on a gold ring. The traditional volumetric closing merges the bottom components, producing an unrealistic effect.

In many cases, the converged closing flow matches its mathematical morphology counterpart (see direct comparisons in Figures 1, 13, 5, 18). However, our curvature-based flow uses at each timestep local *geodesic* information, while morphological operations by disks in general use local *Euclidean* information. We study the effects of this difference in Fig. 25. When two regions of a shape (in red) are *Euclidianly* close but geodesically far, *and* the structuring element radius is of the order of this Euclidean distance (third column), occassionally the volumetric closing will modify the topology of the shape and merge these regions together. As the radius shrinks, "spikes" will appear in the volumetric closing but not in ours (second column). As it shrinks further, the spikes will disappear and our closing matches the volumetric one (first column). The same happens once the radius increases beyond the magnitude of the Euclidean distance (fourth column). These topological changes and non-smooth artifacts are a positive feature of volumetric closing in some applications (tool reachability or CNC milling, for instance). On the other hand, we argue they can be considered undesirable when used for other applications like weathering simulation (see Fig. 24) or smoothing (see Fig. 26) .

Our flow formulation also allows for several generalizations that would be hard or impossible to replicate with traditional volumetric methods. For instance, our energy minimization can easily be transformed to accommodate fixed point or linear equality constraints. In Fig. 27, we select regions that would otherwise move during closing and freeze their positions during the flow.

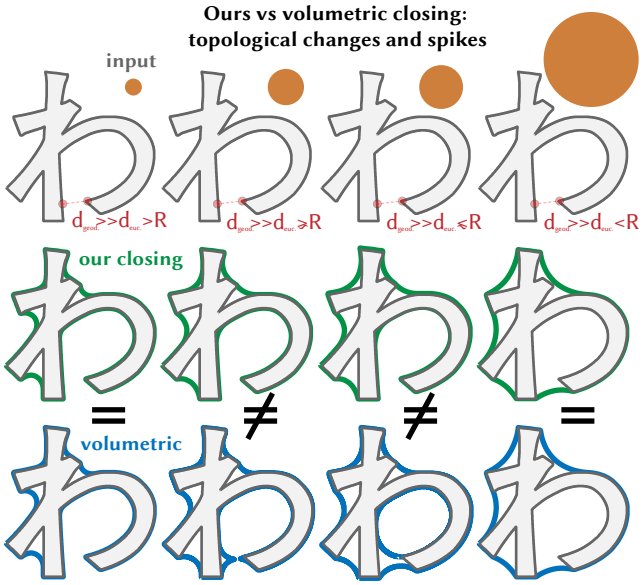


Fig. 25. When regions are geodesically distant but Euclideanly close, our closing flow and the volumetric definition can differ for a specific range of radii (center columns). The equality between the two returns when the radius becomes small enough (left) or big enough (right).

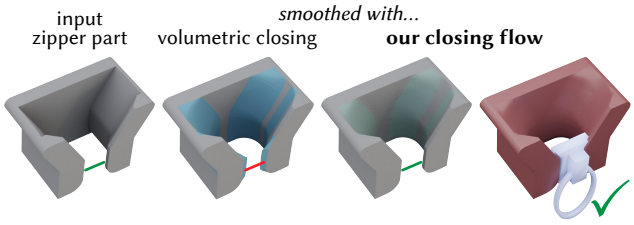


Fig. 26. Our method (center right) smooths a piece of a zipper (left) without producing the "spikes" associated with the volumetric closing (center right) that would make the output unusable for this purpose. 3D model by *CaptainKirk* under CC BY-SA 3.0.

Additionally, our curvature bound can be generalized from a scalar to a function of each point in the surface. In Fig. 28, we transition progressively between "completely fixed" and "moving" with a spatially varying structuring element radius.

Furthermore, stopping the flow *early* is an interesting alternative to using a smaller r and the behavior is not easily replicated with existing morphological operations. In Fig. 30, we explore this artistic control in the context of typeface stylization.

Finally, our energy minimization formulation even allows us to define new operations on the input surfaces. We can simultaneously combine our closing flow (smoothing concave regions) with our opening one (smoothing concave ones) without inducing a preferred ordering. We show an example of this interior and exterior filleting operation, which we dub the "clopening", in Fig. 29.

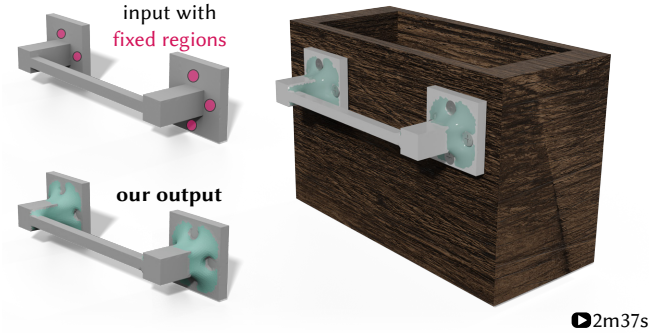


Fig. 27. Our variational formulation facilitates adding problem-specific constraints, such as constraining selected regions to stay put like these screw holes. See accompanying video for complete flow animation.

Fig. 28. Our curvature bound $1/r$ can be given as a scalar function of space, making our flow converge to the closing by a spatially-varying radius. See accompanying video for complete flow animation. 3D model by the Stanford 3D Scanning Repository.

5 LIMITATIONS & FUTURE WORK

Our method is a curvature-based geometric flow and, as such, it can mirror the same self-intersection and singularity behavior of previous methods (e.g., mean curvature flow and Wilmore flow), as shown in Figs. 31 and 32. The flow nature of our method means it can be easily combined with dynamic surface tracking methods like ElTopo ([Brochu and Bridson 2009]) if one wishes to guarantee a self-intersection free output (see Fig. 32), albeit sacrificing our output-sensitive performance.

The explicit obstacle handling in our flow places a constraint on the maximum time step (τ). We would like to understand this bound better. We are inspired by how in physically based simulation explicit collision handling gave way to implicit resolution (e.g., [Kaufman 2009]). We do not maintain a bijective correspondence between the input mesh and the parts that move and get remeshed. This should be possible with a technique such as MAPS [Lee et al. 1998].

As discussed earlier, our method enjoys its differences with the traditional volumetric opening and closing operations. Nonetheless, it would be interesting to explore whether topological changes could indeed be captured. We may consider a method based on first identifying distance bounds on the shape's medial axis (cf. [Yan et al. 2018]) and then using this to somehow attract the flow; topological remeshing would be necessary [Brochu and Bridson 2009].

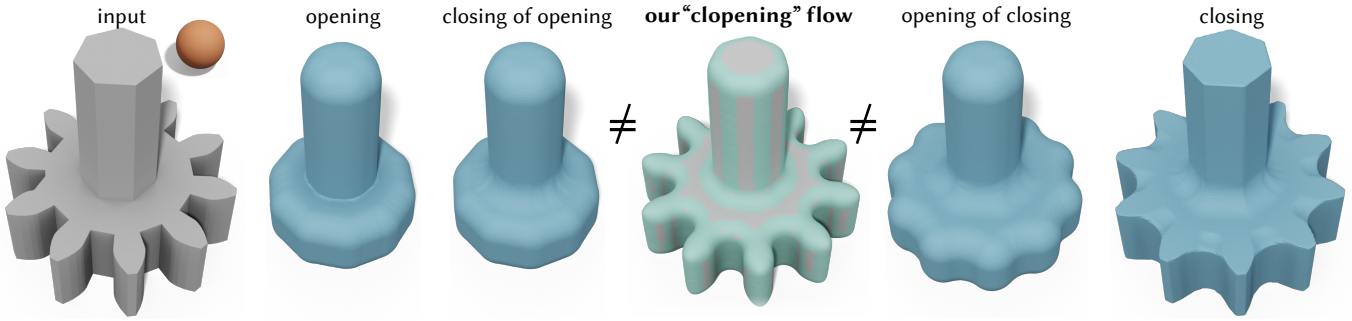


Fig. 29. Our flow enables new morphological operations such as the simultaneous opening and closing of a shape (we dub the "clopening"). Interestingly, the clopening is neither equivalent to closing-then-opening nor to opening-then-closing. 3D model by Jon Ducrou under CC BY-SA 3.0.

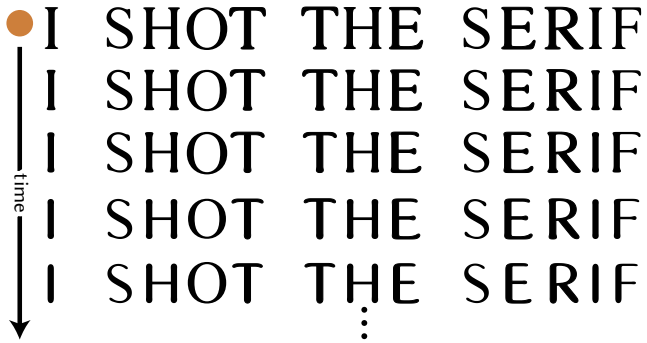


Fig. 30. The *intermediate* steps (not to be confused with the results of opening by increasing radii) of our opening flow can be used for stylization. We progressively *open* each letter, removing serifs. Discrete, non-flow based methods cannot produce intermediate results.

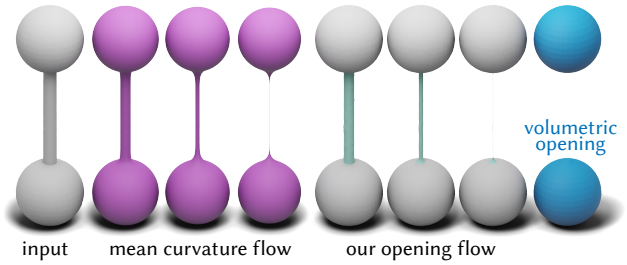


Fig. 31. Singularities and degeneracies common to curvature-based flows can also be encountered with our method.

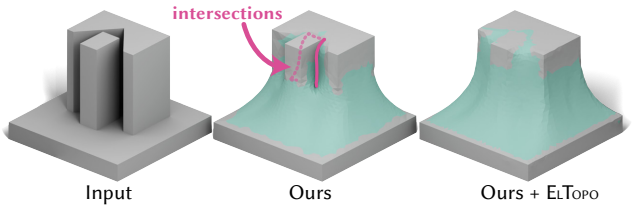


Fig. 32. The local nature of our flow can produce global self-intersections, which one can avoid by combining our method with global remeshers like ElTopo ([Brochu and Bridson 2009]).

We only consider ball-shaped structuring elements. It would be interesting to extend our method to other shapes, possibly starting with oriented ellipsoids. Other, especially non-smooth, structuring elements could be possible by manipulating the norm of the Dirichlet-like energy. It would be interesting to consider asymmetric structuring elements as in the work that original inspired ours [Calderon and Boubekeur 2017].

We hope that our work on surface-only flows for morphological operations ignites future research both on the theory bridging PDEs and mathematical morphology as well as the practice of mesh-based geometry processing.

ACKNOWLEDGMENTS

The first three authors were supported by the 2018 Fields Undergraduate Summer Research Program. Silvia Sellán was also funded by the 2017 - 2019 María Cristina Masaveu Peterson Scholarship for Academic Excellence. This project is funded in part by NSERC Discovery (RGPIN2017–05235, RGPAS–2017–507938), New Frontiers of Research Fund (NFRFE–201), the Ontario Early Research Award program, the Canada Research Chairs Program, the Fields Centre for Quantitative Analysis and Modelling and gifts by Adobe, Autodesk and MESH Inc.

We thank Daniel de la Fuente, Héctor Jardón Sánchez, David Levin, Mirela Ben-Chen, Oded Stein, Leonardo Sacht, Etienne Corman, Noam Aigerman and Derek Liu for their insightful conversation and advice; Ryan Schmidt for his help with the remeshing step of our method; Keenan Crane for sharing his Discrete Differential Geometry slides with us; Josh Holinaty for milling the example in Fig. 7 and modelling the ring in Fig. 26; Rahul Arora and John Kanji for their help putting together the supplemental video; Yasaman Rohanifar for narrating it and Abhishek Madan, Honglin Chen and Ruiqi Wang for proofreading.

REFERENCES

Oscar Kin-Chung Au, Chiew-Lan Tai, Hung-Kuo Chu, Daniel Cohen-Or, and Tong-Yee Lee. 2008. Skeleton extraction by mesh contraction. *ACM Trans. Graph.* 27, 3 (2008).
Hichem Barki, Florence Denis, and Florent Dupont. 2009. Contributing vertices-based Minkowski sum of a non-convex polyhedron without fold and a convex polyhedron.

In *IEEE Intl. Conference on Shape Modeling and Applications*. IEEE, 73–80.
Alexander I Bobenko and Peter Schröder. 2005. Discrete Willmore flow. In *ACM SIGGRAPH 2005 Courses*. 5–es.
Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. 2010. *Polygon mesh processing*. CRC press.
Kenneth A Brakke. 1992. The surface evolver. *Experimental mathematics* 1, 2 (1992).
Tyson Brochu and Robert Bridson. 2009. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing* 31, 4 (2009), 2472–2493.
Stéphane Calderon and Tamy Boubekeur. 2014. Point morphology. *ACM Trans. Graph.* 33, 4 (2014), 1–13.
Stéphane Calderon and Tamy Boubekeur. 2017. Bounding proxies for shape approximation. *ACM Trans. Graph.* 36, 4 (2017), 1–13.
Marcel Campen and Leif Kobbelt. 2010. Polygonal boundary evaluation of minkowski sums and swept volumes. In *Computer Graphics Forum*, Vol. 29. 1613–1622.
Frédéric Cazals and Marc Pouget. 2005. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design* 22, 2 (2005).
Zhen Chen, Daniele Panozzo, and Jeremie Dumas. 2018. Half-space power diagrams and discrete surface offsets. *arXiv preprint arXiv:1804.08968* (2018).
Ulrich Clarenz, Udo Diewald, and Martin Rumpf. 2000. *Anisotropic geometric diffusion in surface processing*. IEEE.
Keenan Crane, Fernando De Goes, Mathieu Desbrun, and Peter Schröder. 2013a. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH Courses*.
Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013b. Robust fairing via conformal curvature flow. *ACM Tran. on Graph.* 32, 4 (2013), 1–10.
Fang Da, Christopher Batty, and Eitan Grinspun. 2014. Multimaterial mesh-based surface tracking. *ACM Trans. Graph.* 33, 4 (2014), 112–1.
Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H Barr. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 317–324.
Michael Eigensatz, Robert W Sumner, and Mark Pauly. 2008. Curvature-domain shape processing. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 241–250.
Frederic Guichard, Petros Maragos, and Jean-Michel Morel. 2005. Partial differential equations for morphological operators. In *Space, Structure and Randomness*.
George W Hart. 2008. Sculptural forms from hyperbolic tessellations. In *2008 IEEE International Conference on Shape Modeling and Applications*. IEEE, 155–161.
Lei He and Scott Schaefer. 2013. Mesh denoising via L_0 minimization. *ACM Trans. on Graph.* 32, 4 (2013), 1–8.
Klaus Hildebrandt and Konrad Polthier. 2004. Anisotropic filtering of non-linear surface features. In *Computer Graphics Forum*, Vol. 23. Wiley Online Library, 391–400.
Samuel Hornus and Sylvain Lefebvre. 2017. Iterative carving for self-supporting 3D printed cavities. (2017).
Alec Jacobson et al. 2016. gptoolbox: Geometry Processing Toolbox. http://github.com/alecjacobson/gptoolbox.
Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.
Mark W Jones and Richard Satherley. 2001. Using distance fields for object representation and rendering. In *Proc. 19th Ann. Conf. of Eurographics (UK Chapter)*.
Daniel M Kaufman. 2009. *Coupled principles for computational frictional contact mechanics*. Citeseer.
Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. 2012. Can mean-curvature flow be modified to be non-singular?. In *Computer Graphics Forum*, Vol. 31. 1745–1754.
Markus Kunze. 2000. *Non-smooth dynamical systems*. Vol. 1744. Springer Science & Business Media.
Aaron WF Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. 1998. MAPS: Multiresolution adaptive parameterization of surfaces. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 95–104.
Hsueh-Ti Derek Liu and Alec Jacobson. 2019. Cubic Stylization. 38, 6, Article 197 (2019).
Petros Maragos and Corinne Vachier. 2008. A PDE formulation for viscous morphological operators with extensions to intensity-adaptive operators. In *2008 15th IEEE International Conference on Image Processing*. IEEE, 2200–2203.
Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. Springer, 35–57.
Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.* 32, 3 (2013), 1–22.
Ken Museth, David E Breen, Ross T Whitaker, and Alan H Barr. 2002. Level set surface editing operators. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 330–338.
Fakir S. Nooruddin and Greg Turk. 2003. Simplification and repair of polygonal models using volumetric techniques. *IEEE TVCG* 9, 2 (2003).
Barrett O'Neill. 1966. *Elementary differential geometry*. Academic Press.
Guillermo Sapiro, Ron Kimmel, Doron Shaked, Benjamin B Kimia, and Alfred M Bruckstein. 1993. Implementing continuous-scale morphology via curve evolution. *Pattern recognition* 26, 9 (1993), 1363–1372.
Jean Serra. 1969. *Introduction la Morphologie Mathématique*.
Oded Stein, Eitan Grinspun, and Keenan Crane. 2018a. Developability of triangle meshes. *ACM Trans. Graph.* 37, 4 (2018), 1–14.
Oded Stein, Eitan Grinspun, Max Wardetzky, and Alec Jacobson. 2018b. Natural boundary conditions for smoothing in geometry processing. *ACM Trans. Graph.* 37, 2 (2018), 1–13.
Stanley R Sternberg. 1986. Grayscale morphology. *Computer vision, graphics, and image processing* 35, 3 (1986), 333–355.
David E Stewart. 2011. *Dynamics with Inequalities: impacts and hard constraints*. SIAM.
John M Sullivan. 2008. Curvatures of smooth and discrete surfaces. In *Discrete differential geometry*. Springer, 175–188.
Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 2016. 3d skeletons: A state-of-the-art report. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 573–597.
Gabriel Taubin. 1995. A signal processing approach to fair surface design. In *Proc. of the 22nd annual conference on Computer graphics and interactive techniques*.
Alexandru Telea and Andrei Jalba. 2011. Voxel-based assessment of printability of 3D shapes. In *International symposium on mathematical morphology and its applications to signal and image processing*. Springer, 393–404.
Charlie CL Wang and Dinesh Manocha. 2013. GPU-based offset surface computation using point samples. *Computer-Aided Design* 45, 2 (2013), 321–330.
Yajie Yan, David Letscher, and Tao Ju. 2018. Voxel Cores: Efficient, robust, and provably good approximation of 3D medial axes. *ACM Trans. Graph.* 37, 4 (2018), 1–13.
Juyong Zhang, Bailin Deng, Yang Hong, Yue Peng, Wenjie Qin, and Ligang Liu. 2018. Static/dynamic filtering for mesh geometry. *IEEE TVCG* 25, 4 (2018), 1774–1787.
Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh arrangements for solid geometry. *ACM Trans. Graph.* 35, 4 (2016), 1–15.