Surface-Filling Curve Flows via Implicit Medial Axes

YUTA NOMA, The University of Tokyo, Japan and University of Toronto, Canada
SILVIA SELLÁN, University of Toronto, Canada
NICHOLAS SHARP, NVIDIA, USA
KARAN SINGH, University of Toronto, Canada
ALEC JACOBSON, University of Toronto and Adobe Research, Canada

Fig. 1. *Snakes on a Plane.* Given an intricate surface with complex topology, our algorithm generates a curve (63,000 nodes) that completely covers the surface while avoiding intersections in under one minute.

We introduce a fast, robust, and user-controllable algorithm to generate surface-filling curves. We compute these curves through the gradient flow of a simple sparse energy, making our method several orders of magnitude faster than previous works. Our algorithm makes minimal assumptions on the topology and resolution of the input surface, achieving improved robustness. Our framework provides tuneable parameters that guide the shape of the output curve, making it ideal for interactive design applications.

ACM Reference Format:
Yuta Noma, Silvia Sellán, Nicholas Sharp, Karan Singh, and Alec Jacobson. 2024. Surface-Filling Curve Flows via Implicit Medial Axes. *ACM Trans. Graph.* 43, 4 (July 2024), 12 pages. https://doi.org/10.1145/3658158

Authors' addresses: Yuta Noma, noma@akg.t.u-tokyo.ac.jp, The University of Tokyo, Tokyo, Japan and University of Toronto, Toronto, Canada; Silvia Sellán, sgsellan@cs.toronto.edu, University of Toronto, Toronto, Canada; Nicholas Sharp, nmwsharp@gmail.com, NVIDIA, USA; Karan Singh, karan@dgp.toronto.edu, University of Toronto, Toronto, Canada; Alec Jacobson, jacobson@cs.toronto.edu, University of Toronto and Adobe Research, Toronto, Canada.

0730-0301/2024/7-ART
https://doi.org/10.1145/3658158

1 INTRODUCTION

A *surface-filling curve* is a continuous map from a one-dimensional space onto a two-dimensional manifold, which guarantees that every point on the surface lies within a given maximum distance of the curve (see Figure 1). In computer graphics, these curves have primarily been leveraged for digital fabrication and artistic creation; yet, their applicability extends to a variety of fields, including robotic path planning, sensor placement, circuit design, and more. Nonetheless, despite their significant potential, existing methods suffer from slow computation time, robustness issues, and limited user control.

We introduce a fast, robust, and controllable algorithm to generate surface-filling curves. We model the problem through a two-term energy that balances curve length and distance to the curve's medial axis. Instead of computing the medial axis *explicitly*, we only *implicitly* compute the distance to it, making it ideal for speed and

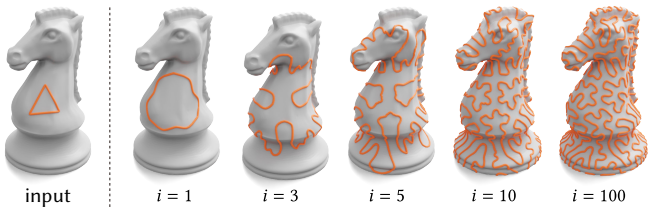


Fig. 2. We compute surface-filling curves through a geometric flow that evolves to balance total length with distance to the curve's medial axis.

Fig. 3. Surface-filling curves can potentially be used in applications like robotic path planning for tasks including painting or polishing surfaces.

robustness. The gradient of this energy yields a geometric flow that evolves any input curve into a surface-filling one (see Figure 2), which is efficiently evaluated by solving a *sparse* linear system with size proportional to the number of nodes in the curve.

Recently, the *Repulsive Curves* algorithm [Yu et al. 2021] showed applications to this problem, generating curves constrained to simple surfaces with a physically-inspired energy. While that method must deal with a *dense* linear system solve at each iteration, our method produces qualitatively similar results via *sparse* ones, leading to a speed-up of several orders of magnitude. Combined with this massive performance improvement, our algorithm's superior robustness enables curve generation on a broader set of surfaces with diverse styles, resolutions, and topologies. Together with its speed and robustness, our high degree of user control makes it ideal for interactive applications in which a designer may wish to change the spacing of the curve, align it to a given density map or vector field, or even edit the underlying surface.

We showcase the performance of our method by testing it on a diverse set of inputs inspired by a broad range of contexts. We extensively evaluate our algorithmic choices through ablation studies and include experiments showing the effect of the different parameters in our energy formulation. Our flow's significant improved performance enables a whole new set of applications, which we exemplify through results in which a user can interactively specify the curve's global shape, local density, and alignment.

2 RELATED WORK

Surface-filling curves are a special case of *space*-filling curves that fill a given d-dimensional domain with one continuous stroke without self-intersections (e.g., [Hilbert 1891; Peano 1890]). Over the past decades, space-filling curves have received renewed interest from computer graphics researchers due to their applications ranging from infill patterns for 3D printing [Chermain et al. 2023] to toolpath planning for CNC milling [Zhao et al. 2018] and procedural art [Pedersen and Singh 2006]. To address these applications, a key challenge lies in filling out intricate 2D shapes or 3D surfaces instead of simpler geometries (e.g., spheres [Gerlach and von der Mosel 2011]). We classify such methods into two subclasses: initialization-based methods and geometric flows. We also show a feature table of methods that can compute surface-filling curves in Table 1.

2.1 Initialization-based methods for space-filling curves

Initialization-based methods begin by searching for an approximation near the optimal state, often followed by a local relaxation of the curve to determine its final form. These methods can be categorized as either *discrete* or *continuous*.

Discrete approaches begin by expressing the domain as a finite set of points [Giannatsis et al. 2015], clusters [Yan and Mostofi 2016], grid cells [Cheng et al. 2020; Joshi et al. 2019] or more general polygonal elements [Akleman et al. 2013; Bedel et al. 2022; Lin et al. 2019; Xing et al. 2012] with some notion of adjacency. This representation is then treated as a graph on which a Hamiltonian cycle is computed or approximated, often followed by some local, geometric post-processing in the form of smoothing [Xing et al. 2012] or alignment to a given vector field [Bedel et al. 2022].

Among discrete approaches, the most relevant for our work is the surface-filling curve generation method by Xing et al. [2012] and the follow-up work by Akleman et al. [2013]. They compute a Hamiltonian cycle on the dual graph of any input surface mesh, followed by optional subdivision and smoothing. By construction, these algorithms are heavily dependent on the input triangulation and provide little to no user control in terms of curve spacing, alignment, and overall shape.

Continuous approaches to space-filling curve generation directly model the curve as a smooth, geometric object on the input domain. A subclass of continuous methods extracts the iso-contours of a continuous function, such as a distance transform [Zhao et al. 2016, 2018], periodic function [Fang et al. 2020; Tricard et al. 2021], cylindrical parametrization [Schüller et al. 2018], or simply the z-coordinates [Zhong et al. 2023]. However, they must carefully stitch disconnected components (see, e.g., [Chermain et al. 2023]) or manually segment the mesh into cylindrical pieces (see [Schüller et al. 2018]) to ensure a true single-stroke filling curve.

To escape the inherent problems in initialization-based methods, we instead opt for growing curves from arbitrary initializations via geometric flows.

2.2 Geometric flows for space-filling curves

Geometric flows evolve a geometry by following the variational gradient of some energy functional [Brakke 1992]. The most basic example is curve-shortening flows, in which the curve evolves to minimize its own length. Often combined with area-preservation

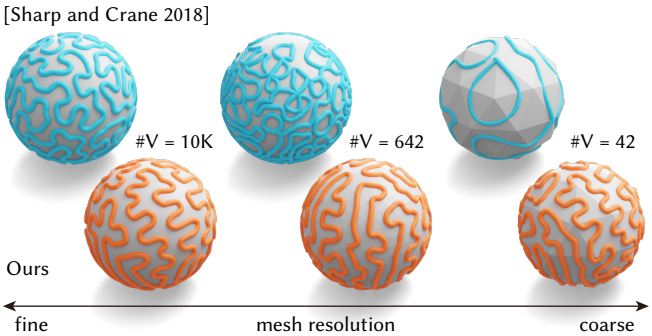


Fig. 4. Unlike the distortion-minimizing flow by Sharp and Crane [2018], our algorithm is robust to changes in underlying mesh resolution.

Table 1. Previous methods fall short in one way or another.

	Custom penalties	Low-res meshes	Spacing control	Unicursality	Mesh-surgery-free
[Xing et al. 2012]	✗	✗	✗	✓	✓
[Akleman et al. 2013]	✗	✗	✗	✓	✓
[Schüller et al. 2018]	✗	✓	✓	✓	✗
[Zhao et al. 2018]	✗	✓	✓	✗	✗
[Sharp and Crane 2018]	✓	✗	✗	✗	✗
[Yu et al. 2021]	✓	✗	✗	✓	✓
Ours	✓	✓	✓	✓	✓

terms [Crane et al. 2013] or modified to avoid singularities [Kazhdan et al. 2012], variants of this flow have become common in geometry processing tasks like smoothing [Taubin 1995] and even morphological operations [Sellán et al. 2020].

To the best of our knowledge, only two prior works have suggested using geometric flows to compute surface-filling curves. First, Sharp and Crane [2018] observed that their conformality-inducing flow serendipitously produced surface-filling curves if the weight on the length penalization was low enough. While an impressive offshoot result, this method suffers from robustness issues at low mesh resolutions (see Figure 4) and the inability to handle open curves.

More recently, Yu et al. [2021] proposed following the gradient flow of a physics-inspired repulsive energy between every disconnected pair of points on the curve. The authors followed a rigorous mathematical derivation and introduced a modification of gradient descent combined with a multi-grid scheme to evolve the curve according to the proposed energy. While this mathematical and computational machinery makes their algorithm tractable, by considering an energy defined between every pair of points, their energy is fundamentally dense. Although they provide a hierarchical multigrid solver to circumvent dense linear solves, the linear solves associated with each level of the hierarchy still remain a bottleneck. In this paper, we build on the work by Yu et al. [2021] and show that this repulsive energy can be substituted by a simple geometric proxy based on the distance from each point on the curve to its medial axis. As we show in Figure 5 and empirically throughout the paper, this substitution produces qualitatively similar results while critically converting the dense solve into a sparse one, obtaining a massive speed-up and enabling a broader set of applications.

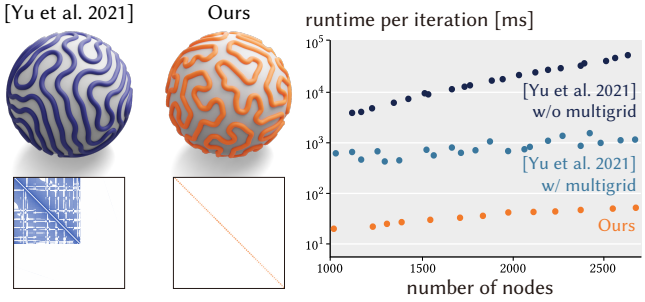


Fig. 5. We qualitatively match the results produced by Yu et al. [2021] (top left), but instead of a dense linear system solve, we only need a sparse one whose entries are banded near the diagonals (bottom left), providing several digits of speed-up (right). See Appendix C for details on runtime measurement.

3 METHOD

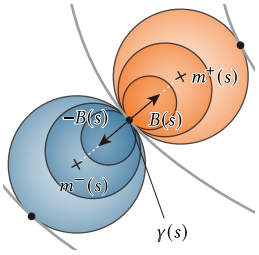
In what follows, let us assume that we are given a surface $\Omega \in \mathbb{R}^3$ and a curve $\gamma : [0, l] \mapsto \Omega \in \mathbb{R}^3$ on it. We define a consistent frame on γ as follows: at any point $\gamma(s)$ on γ, $T(s)$ is the (unit) tangent vector to the curve, $N(s)$ is the (orthogonal) normal vector to the surface at $\gamma(s)$ and $B(s)$ is the (unit) bitangent vector, which is defined as $B(s) = T(s) \times N(s)$. We will use the notation $\|a - b\|_\Omega$ to denote the geodesic distance between two points $a, b \in \Omega$, and $\|a - b\|$ to denote the Euclidean distance between two points $a, b \in \mathbb{R}^3$.

Our goal will be to find another curve $\gamma^\star$ that fills Ω with some radius r; that is, for any point $p \in \Omega$, there exists a point $q \in \gamma^\star$ such that $\|p - q\|_\Omega \leq r$. We will do this by constructing an energy functional $E(\gamma)$ and iteratively minimizing it with respect to γ starting from some initial curve γ_0.

3.1 Energy construction

To motivate our choice of energy, we will start from the very definition of a surface-filling curve stated above: for any point $p \in \Omega$, there must exist a point $q \in \gamma^\star$ such that $\|p - q\|_\Omega \leq r$. Let us now briefly consider the minimum distance function $d(p, \gamma) = \min_{q \in \gamma} \|p - q\|_\Omega$ as a function of p. We can then rephrase the surface-filling condition as: for any point $p \in \Omega$, its minimum distance to the curve is below r. $d(p, \gamma)$ being a continuous function of p, it is sufficient to impose this condition on the points $p \in \Omega$ at which $d(p, \gamma)$ is locally maximized, and they always lie on the *medial axis* of γ on Ω, which we denote $\mathcal{M}_{\gamma,\Omega}$. In other words, γ is a surface-filling curve if for every point p on the medial axis $\mathcal{M}_{\gamma,\Omega}$, its distance to the curve is below r.

We will now define a mapping from $[0, l]$ to $\mathcal{M}_{\gamma,\Omega}$, which will allow us to restate the surface-filling condition in terms of the curve γ itself. The key observation here is that we only need the distance to the medial axis rather than its explicit graph. For every $s \in [0, l]$, we will grow a (geodesic) sphere of radius r tangent at $\gamma(s)$ until it touches a different point on γ (see inset). The center of this maximal sphere is, by definition, an element of the medial axis $m(s)$. In fact, one can grow two such spheres, one



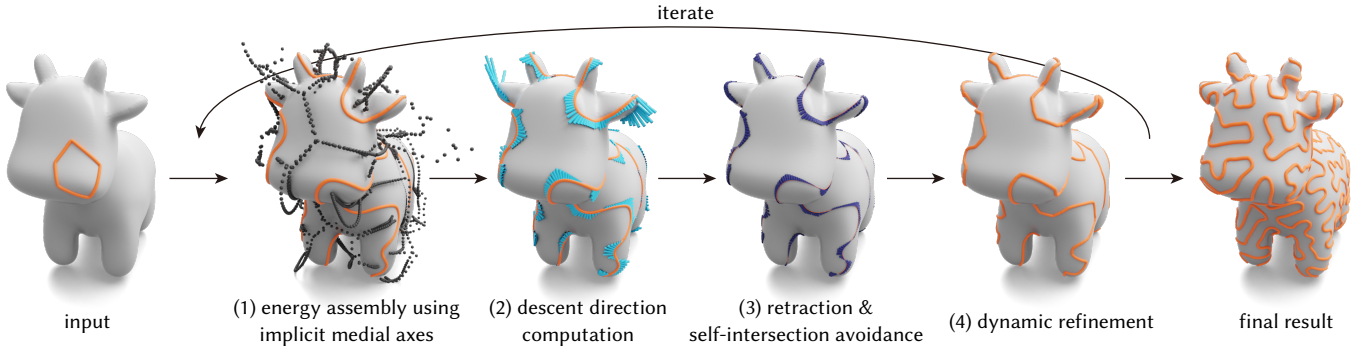


Fig. 6. We obtain a surface-filling curve by iteratively growing an input curve through a geometric flow. At each iteration, we efficiently assemble a quadratic energy, compute its retracted descent direction on the surface, and evolve the curve along it, dynamically refining it to maintain a homogeneous discretization.

in the direction of $B(s)$ and one in the opposite direction $-B(s)$, leading to two (not necessarily distinct) points $m^+(s), m^-(s) \in \mathcal{M}_{\gamma,\Omega}$, which we call the *implicit medial axis*. When one of the spheres grows infinitely (e.g., a completely convex curve), we clamp the distances to a parameter r_{max}.

These mappings $m^+, m^- : [0, l] \mapsto \mathcal{M}_{\gamma,\Omega}$ are not necessarily injective, but they are surjective: as a direct consequence of the definition of medial axis, every element in it is the image through one (or both) of m^+ or m^- of at least one value of s. Thus, we can now rephrase the surface-filling condition as a condition on the curve: γ is surface-filling if for every $s \in [0, l]$, $\left\|\gamma(s) - m^+(s)\right\|_\Omega \leq r$ and $\|\gamma(s) - m^-(s)\|_\Omega \leq r$. This reformulation is critical, as it allows us to define the curve's *medial axis energy* as

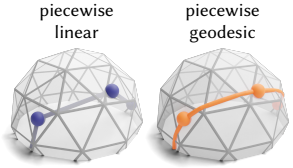
$$E_M = \int_\gamma \left(\left\|\gamma(s) - m^+(s)\right\|_\Omega^2 + \|\gamma(s) - m^-(s)\|_\Omega^2 \right) ds. \tag{1}$$

Since it accumulates the distance values in an L^2 sense, we can expect that progressively minimizing this energy will concentrate the medial axis distances around an increasingly smaller value. Of course, there is an infinite number of curves that satisfy the surface-filling constraint as defined above. A reasonable choice between all of them is the *shortest* one, which we can enforce by adding a term $L_\Omega(\gamma)$ measuring the total (geodesic) length to the energy

$$E = L_\Omega(\gamma) + \alpha E_M\,, \tag{2}$$

where α is a balancing coefficient, which we will later show to be related to the desired curve width r. Eq. (2) is our final analytical energy, which we will now discretize in the form of a sparse quadratic form that one can efficiently assemble.

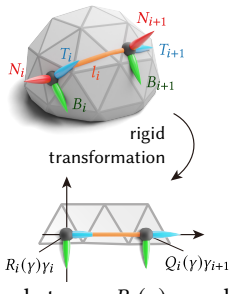
3.2 Discretization



Let us assume that Ω is given as a triangular mesh with vertices $\mathcal{V}$ and faces $\mathcal{F}$. Instead of existing representations (e.g., [Bischoff et al. 2005]),we represent γ as a *piecewise geodesic* curve by storing it as a set of n nodes $\gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ that are connected by geodesic segments on Ω (see inset). Beyond manifoldness, we make no assumption on the topology or connectivity of γ (see Figure 10) or Ω (see Figure 9 or Figure 12). In what follows, we refer to elements in the underlying surface as *vertices* and *faces* whereas elements in the curve as *nodes* and *segments*. Each node γ_i is equipped with the vertex-blended normal N_i, the normalized bisection tangent T_i projected onto its tangent plane, and their cross product B_i. As noted by Sharp and Crane [2020], these geodesic segments are *intrinsic*; i.e., they are independent of how the surface patch containing the segment is embedded in $\mathbb{R}^3$. We will begin by making use of this property to approximate the curve length via a quadratic form.

Fig. 7. Our method is fast enough to obtain surface-filling curves with hundreds of thousands of nodes (180K) in a reasonable time (229 s).

3.2.1 Curve length. Consider the geodesic segment between γ_i and γ_{i+1}, which we denote $\gamma_{i,i+1}$, with geodesic length l_i. Let $R_i(\gamma)$ be the (unique) rigid transformation that maps γ_i to the origin 0 and aligns the tangent vector T_i with the x-axis and the normal vector N_i with the z-axis, and let $Q_i(\gamma)$ be the rigid transformation that maps $\gamma_{i,i+1}$ to the point $(l_i, 0, 0)$, similarly aligning the tangent vector T_{i+1} with the x-axis and the normal vector N_{i+1} with the z-axis. Intuitively, these are the rigid transformations one would obtain from *unrolling* all the triangles in the surface patch containing $\gamma_{i,i+1}$ onto the plane (see inset).



Trivially, the geodesic distance between γ_i and γ_{i+1} is the Euclidean distance between $R_i(\gamma)\gamma_i$ and $Q_i(\gamma)\gamma_{i+1}$ on the XY plane:

$$l_i = \|\gamma_i - \gamma_{i+1}\|_\Omega = \|R_i(\gamma)\gamma_i - Q_i(\gamma)\gamma_{i+1}\|_2\,. \tag{3}$$

By assuming only small changes in γ, we can linearize the products in Eq. (3) and multiply and divide by l_i to write the length as a

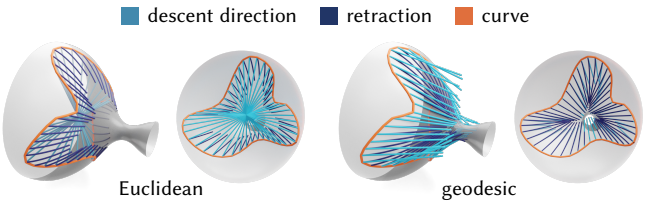


Fig. 8. The descent direction of Eq. (5) stays on the tangent plane of the surface (right), allowing larger steps than merely evaluating the Euclidean length between each node (left).

Fig. 9. Our method does not assume the topology or connectivity of the underlying surface, which may be open and have boundaries.

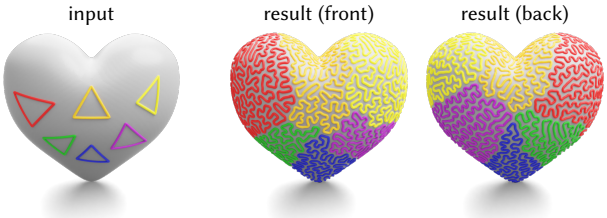


Fig. 10. Our method does not assume the topology or connectivity of the input curve and preserves it throughout the flow.

quadratic form in γ_i, γ_{i+1}:

$$l_i \approx \frac{1}{l_i} \left\| R_i \gamma_i - Q_i \gamma_{i+1} \right\|_2^2 . \tag{4}$$

Accumulating the length of every segment, we can now approximate the curve length through a quadratic form on $\boldsymbol{\gamma}$:

$$L_\Omega(\boldsymbol{\gamma}) \approx \sum_{i=1}^{n-1} \frac{1}{l_i} \left\| R_i \gamma_i - Q_i \gamma_{i+1} \right\|_2^2 = \frac{1}{2} \boldsymbol{\gamma}^T \mathbf{A} \boldsymbol{\gamma} + \ell \boldsymbol{\gamma} , \tag{5}$$

where $\boldsymbol{\gamma}$ is the vector of stacked coordinates of the curve nodes, $\mathbf{A}$ is a sparse symmetric positive-semidefinite matrix of size $3n \times 3n$, and ℓ is a vector of size $3n$ that accounts for the translation component in the rigid transformations. We observed that this approximation allows larger steps than merely adding up the Euclidean distance between γ_i and γ_{i+1} (see Figure 8).

Now that we have managed to approximate the curve length as a quadratic form, we can proceed to do the same for the second half of Eq. (2), the medial axis energy.

3.2.2 Medial axis energy. Recall that the medial axis energy is defined as

$$E_M = \int_\gamma \left(\left\| \gamma(s) - m^+(s) \right\|_\Omega^2 + \left\| \gamma(s) - m^-(s) \right\|_\Omega^2 \right) ds ,$$

where $m^+(s)$ and $m^-(s)$ are the implicit medial axis, i.e., the two points on the medial axis $\mathcal{M}_{\gamma,\Omega}$ that are closest to $\gamma(s)$.

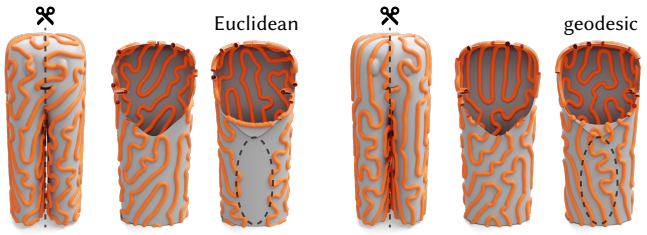
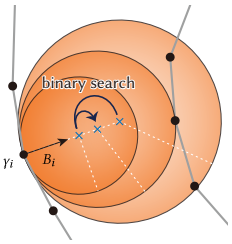


Fig. 11. The implicit medial axis in Euclidean space is affected by curves on nearby regions of the surface without local connectivity.

The most immediate question one faces when attempting to discretize E_M is how to compute $m^\pm$ for a given curve node γ_i. It is known that computing the explicit medial axis (i.e., the cut locus) on curved surfaces often becomes tedious (see, e.g., [Mancinelli et al. 2021] Section 7), so we opt for computing the implicit medial axis for each node. One could loop over every γ_j, computing the largest geodesic sphere that is tangent to both γ_i and γ_j to then choose the minimum radius among all of them. However, this approach would lead to a quadratic complexity in the number of nodes, which is prohibitive for large curves we wish to model (see Figure 7).

Instead, we once build a closest point query data structure on the curve nodes (and the boundary vertices of the underlying mesh if they exist) and then compute m_i^+ by carrying out a binary search on the radius of a sphere grown from γ_i in the direction of B_i until it lays within a tolerance of a given node γ_j and contains no other node (see inset). We then repeat the same process for m^-, growing a sphere in the direction of $-B_i$ instead.



The geodesic closest point query can be built upon any state-of-the-art geodesic computation algorithm [Crane et al. 2020], but substituting geodesic spheres with Euclidean spheres can make the algorithm even faster. This substitution is a common operation in geometry processing (e.g., [Yuksel 2015]). The Euclidean closest point query can be built upon a bounding volume hierarchy, leading to a time complexity of $O(\log n)$ per node and $O(n \log n)$ in total. We note that all of the quantitative measurements in this paper are based on the Euclidean medial axis. The tradeoff is that $m_i^\pm$ is not guaranteed to lie on the surface (see Figure 6 (1)), leading to interference from nearby surfaces without local connectivity (see Figures 11 or 12) or larger errors on high curvature regions (see Figure 13). We note that the Euclidean medial axis is more suitable in some situations regardless of the error, e.g., path planning for 3D printing where the filament has a constant width in Euclidean space. We leave this choice to the user, depending on their desired applications.

To discretize Eq. (1), we adopt the same idea as the curve length to *unroll* the geodesic path between γ_i and $m_i^\pm$ onto the tangent space of γ_i by:

$$\left\| \gamma_i - m_i^\pm \right\|_\Omega = \left\| \gamma_i - (\gamma_i \pm r_i^\pm B_i) \right\|_2 , \tag{6}$$

where r_i^+ and r_i^- are the sphere radii obtained by the binary search.

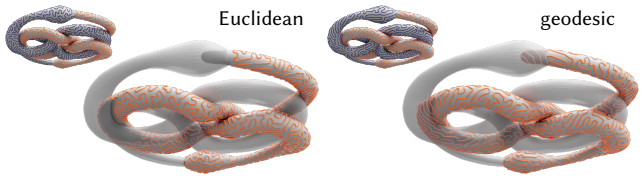


Fig. 12. Our method does not assume the topology or connectivity of the underlying surface, which may be self-intersecting. We also observed that the selection of Euclidean/geodesic space for the implicit medial axis affects the result near the intersecting regions.

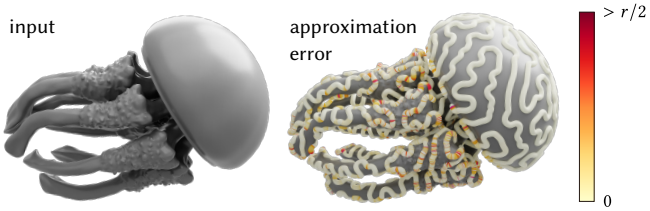


Fig. 13. Our medial axis energy can be defined via Euclidean or geodesic sphere radii, with the difference between these two being higher in high curvature regions.

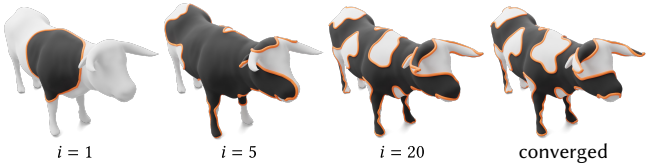


Fig. 14. Our method preserves the inside/outside of the curve throughout the flow.

By treating the unrolled implicit medial axis $m_i'^{\pm} = \gamma_i \pm r_i^{\pm} B_i$ as momentarily fixed for an optimization iteration and summing contributions from each curve vertex, we approximate the medial axis energy as:

$$E_M(\boldsymbol{\gamma}) \approx \sum_{i=1}^{n} \frac{l_{i-1} + l_i}{2} \left(\left\| \gamma_i - m_i'^{+} \right\|_2^2 + \left\| \gamma_i - m_i'^{-} \right\|_2 \right), \tag{7}$$

which is yet another quadratic form

$$E_M(\boldsymbol{\gamma}) \approx \frac{1}{2} \boldsymbol{\gamma}^T \mathbf{M} \boldsymbol{\gamma} + \mathbf{n} \boldsymbol{\gamma}, \tag{8}$$

where $\mathbf{M}$ is a symmetric matrix (see Appendix B for discussions on this approximation).

By combining Eq. (5) and Eq. (8), we can now write the approximation of the energy from Eq. (2) as a quadratic form

$$E(\boldsymbol{\gamma}) = \boldsymbol{\gamma}^T \mathbf{A} \boldsymbol{\gamma} + \ell \boldsymbol{\gamma} + \alpha \frac{1}{2} \boldsymbol{\gamma}^T \mathbf{M} \boldsymbol{\gamma} + \alpha \mathbf{n} \boldsymbol{\gamma} = \frac{1}{2} \boldsymbol{\gamma}^T \mathbf{H} \boldsymbol{\gamma} + \mathbf{v} \boldsymbol{\gamma}, \tag{9}$$

where, notably, $\mathbf{H}$ is a *sparse* symmetric matrix.

3.3 Curve evolution

Given a discrete curve $\boldsymbol{\gamma}^k$ at the k-th timestep of the flow, we can compute its approximated discretized energy $E(\boldsymbol{\gamma}^k)$ by evaluating Eq. (9). We can then compute the gradient of this energy with respect to $\boldsymbol{\gamma}^k$ as $\mathbf{g}^k = \nabla E(\boldsymbol{\gamma}^k) = \mathbf{H}^k \boldsymbol{\gamma}^k + \mathbf{v}^k$, and its Hessian as $\nabla^2 E(\boldsymbol{\gamma}^k) = \mathbf{H}^k$. We can then use both these quantities to get the preconditioned descent direction via a single sparse linear system solve

$$\mathbf{d}^k = -(\mathbf{H}^k)^{-1} \mathbf{g}^k. \tag{10}$$

Of course, this coordinate-wise descent direction is not guaranteed to keep the curve on the surface Ω, so we will need to instead compute the *retraction* (see, e.g.,[Boumal 2023]) of the descent direction onto Ω. Given the descent direction d_i^k for each node γ_i^k, we compute the retraction by finding the projection of d_i^k onto the tangent plane of Ω at γ_i^k, which we denote $\hat{d}_i{}^k$, and explicitly tracing the geodesic path from γ_i^k in the direction $\hat{d}_i{}^k$ of length $\|\hat{d}_i{}^k\|$. Finally, we reconnect the piecewise geodesic segments using a geodesic computation algorithm. In practice, we find that the magnitude of the descent is larger in the first iterations and then plateaus (see Figure 16).

3.3.1 Self-intersection avoidance. To guarantee an intersection-free output, we add a backtracking line search during our curve evolution. After following the descent direction in its entirety, we loop over every triangle and check that no two geodesic segments lying on the same triangle intersect (this is easy enough to do, as geodesics have the form of straight segments within each triangle). If they do intersect, we multiply the descent direction by a coefficient $c \in [0, 1)$, evolve the curve, and start the check again. We continue this procedure until the curve never causes self-intersection. This procedure was necessary for all of the models we tested.

3.3.2 Dynamic refinement. Our computation of the medial axis energy only considers distances between pairs of nodes; as such, it relies heavily on the curve being finely discretized. Further, for our curve evolution to find an optimal surface-filling disposition, it needs to be provided with enough degrees of freedom during the optimization. For these reasons, we follow the lead of other flow-based approaches [Sellán et al. 2023, 2020] and use a dynamic refinement scheme that adaptively subdivides the curve during the flow: given a user-defined value h, we collapse edges with lengths smaller than h and subdivide those larger than $2h$. We discuss the effect of h in Section 4.2.

Table 2. Runtimes reported for several meshes. We kept iterating until the average length of the descent direction per node went below $3 \times 10^{-4} r$.

Model	Vertices	Nodes (when converged)	Iterations	Runtime
Boot	19.9K	7.5K	41	8.8 s
Bunny	3.3K	11.2K	24	3.6 s
Fish	10.8K	4.7K	26	2.3 s
Hand	8.0K	10.2K	25	2.5 s
Spot	11.5K	3.0K	25	1.5 s
Springer	9.7K	16.1K	27	4.8 s
Stuffedtoy	6.7K	58.1K	32	17.3 s

4 EXPERIMENTS AND APPLICATIONS

Implementation details. We implemented our algorithm in C++ using GeometryCentral [Sharp et al. 2019] and libigl [Jacobson et al. 2018] for common geometry processing subroutines, TinyAD [Schmidt et al. 2022] for computing the Hessian in Eq. (10), knn-cpp [Meyer 2019] for closest point queries, and the MMP algorithm

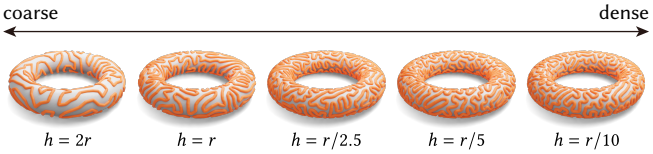


Fig. 15. Our flow dynamically splits and collapses edges to maintain a user-specified edge-length h. In practice, we find our flow to be robust to this parameter choice as long as it is below $r/2.5$.

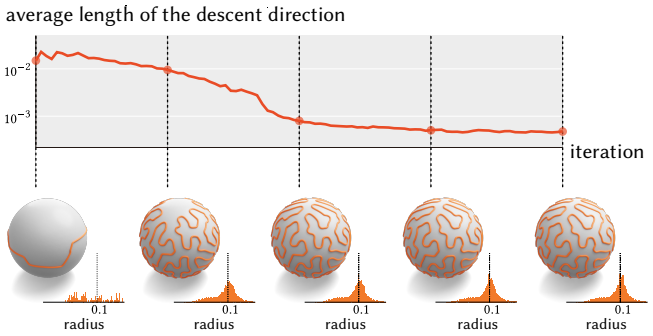


Fig. 16. As the flow progresses, the magnitude of each descent diminishes and the distance to the medial axis concentrates around a single value.

[Mitchell et al. 1987] to reconnect the geodesic segments. In Table 2, we report the runtimes of several representative models in this paper, which we have confirmed scales with $O(n^{1+f})$, where $f > 0$ accounts for the Laplacian-like sparse system solve. All our reported runtimes were carried out on a 2022 M2 MacBook Air with 24GB of RAM.

4.1 Comparisons

We compare the performance of our algorithm to both the first work to suggest computing surface-filling curves through geometric flows [Sharp and Crane 2018] (Figure 4) as well as to the most recent state-of-the-art algorithm for this task [Yu et al. 2021] (Figure 5).

As we show in Figure 4, our algorithm presents a robustness and mesh independence that is not present in the work by Sharp and Crane [2018], allowing its use in a broader set of surfaces. Further, as seen in Figure 5, by substituting the dense linear solve proposed by Yu et al. [2021] for a sparse one, our algorithm provides massive speed-ups in several orders of magnitude, enabling us to compute intricate surface-filling curves with hundreds of thousands of vertices (see, e.g., Figures 1 and 7).

This performance improvement comes at a theoretical cost in the form of the approximations carried out in Section 3.2.2; however, as we exhaustively show in the rest of this section, these approximations do not seem to have a significant impact on the quality of the curves produced by our algorithm.

4.2 Experiments and ablations

As we illustrate in Figure 6, our algorithm works by iteratively following the surface-retracted gradient of the energy in Eq. (2). In general, as shown qualitatively in Figure 2 and numerically in Figure 16, we find that the first iterations of the flow tend to produce

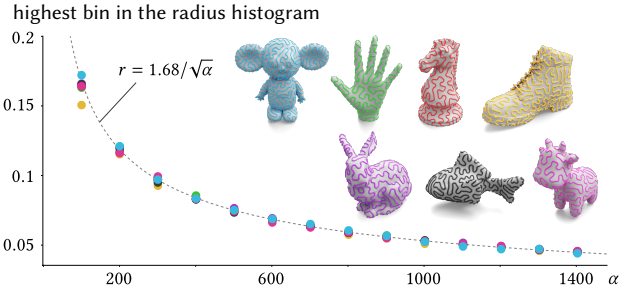


Fig. 17. We find an empirical relation between the energy weight α and the curve thickness (here computed as the highest bin in the medial axis distance histogram) that is consistent across input surfaces and curves.

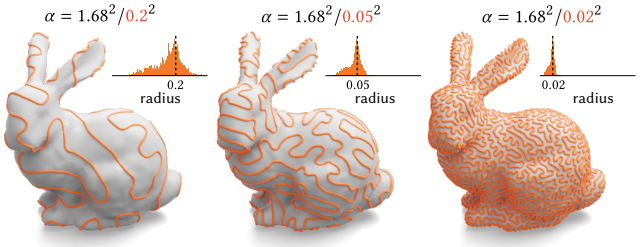


Fig. 18. Given the observed relation in Figure 17, a user can specify the desired surface-filling thickness through the parameter α.

the largest changes in the curve followed by a gradual convergence to a surface-filling curve whose medial axis distances are clustered around a value r.

A key parameter in this energy is the balancing coefficient α, which we find serves as a proxy for the thickness r of the final surface-filling curve. Experimentally, we find the relation $r \approx 1.68/\sqrt{\alpha}$, which appears to be consistent across different curve shapes, thicknesses, and underlying meshes (Figure 18). This matches the dimensional units of Eq. 2, where α is presumed to have a dimension of inverse squared length. While a theoretical derivation of this relation escapes us, as shown in Figure 17, one can still utilize it to specify the width of the curve by simply changing α.

Our algorithm also requires the specification of two other parameters; namely, the target segment length h and the maximum sphere size r_{max}. As shown in Figure 15, we find that our algorithm is invariant to the selection of h as long as $h < r/2.5$. Similarly, as shown in Figure 19, we find that r_{max} can be set to any value larger than $10r$ without too significantly affecting the shape of the final curve, with values between $2r$ and $10r$ producing curves with less pronounced wriggles.

4.3 User control

Our method's robustness makes it ideal for interactive applications where a user or designer may want to explore different curves, surfaces, or parameters. This application guides our algorithmic design even from a fundamental theoretical level; we make only minimal (manifoldness) assumptions on the topology of the input surface and curve (see Figures 9, 12, and 10). Also, as shown in Figure 20, its performance can accommodate even realtime deformation of the underlying surface while running the flow.

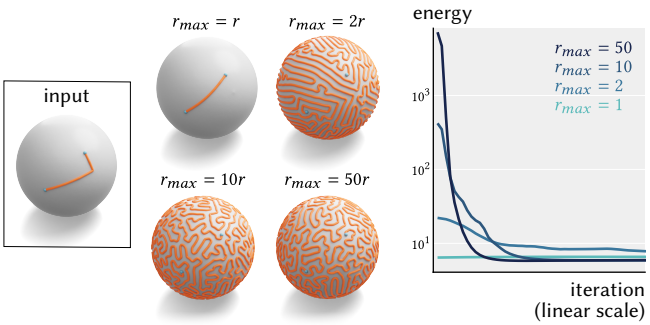


Fig. 19. r_{max} controls the numerical behavior of our energy by clamping the medial axis distance, critical at early flow stages. This parameter can be set to any value larger than $10r$ without much impact on the output, with values between $2r$ and $10r$ producing curves with less pronounced wriggles.

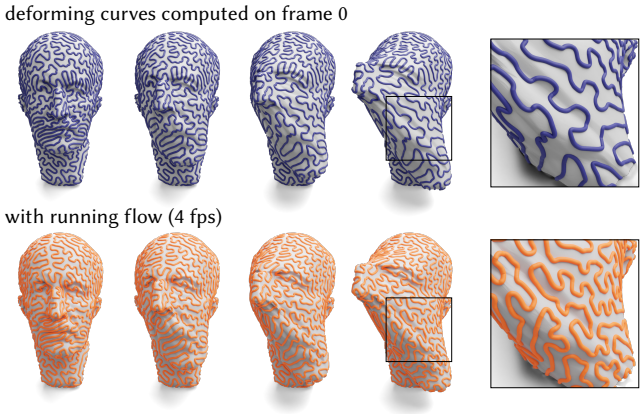


Fig. 20. Our algorithm is fast enough to run the flow for one step per frame while allowing real-time deformation of the underlying mesh. Even with a single step per frame, the spacing remains constant as opposed to merely deforming the curve.

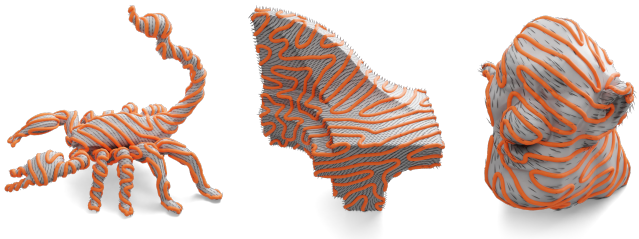


Fig. 21. By adding a field-alignment energy, we can encourage the curve to align itself to a specified tangent vector field.

For its use in artistic applications, a designer must be able to modify the generated curves in an easily controlled manner. Our framework allows for several modifications, such as aligning the curve to a given vector field (see Figure 21) or eliminating pronounced wriggles (see Figures 22) by adding simple sparse energies (see Appendix A for details).

Our algorithm also provides a powerful yet intuitive way to control the shape of the curve by specifying a spatially varying density

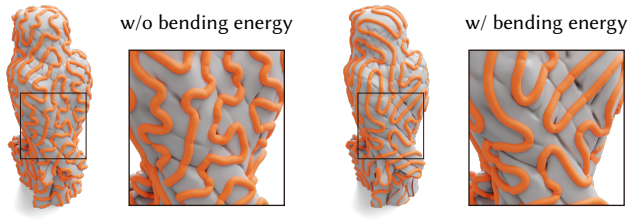


Fig. 22. By adding a biharmonic bending energy, we can encourage the curve to be straighter.

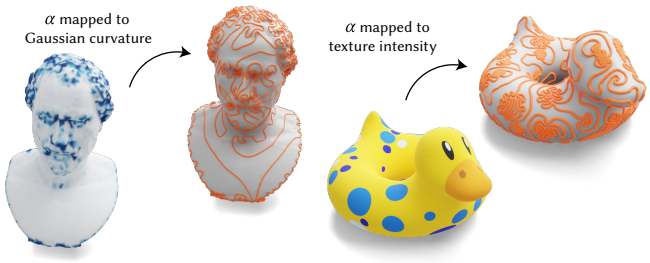


Fig. 23. An artist may choose to use a spatially-varying α to generate curves of varying density, by mapping it to a property like curvature for procedural generation (left) or to any texture for more manual control (right).

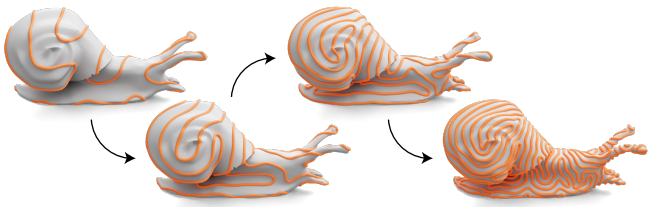


Fig. 24. By progressively decreasing r combined with the bending term, we can produce a surface-filling curve with a small number of sharp corners.

through the α parameter. As shown in Figure 23, an artist may procedurally generate aesthetic patterns by mapping α to geometric quantities like the Gaussian curvature of the underlying surface or, for more manual control, by simply painting the desired density on the surface through a texture. Also, as shown in Figure 24, we can obtain a curve with a minimum number of sharp corners by progressively decreasing r combined with the bending energy.

Finally, as shown in Figure 26, our method can obtain a unicursally connected curve, thanks to its topology-preserving feature (see Figure 14). This can be favorable for deploying a robotic toolpath (Figure 3), interactive sensor network (Figure 25), or heating/-cooling system (Figure 25 and Figure 27) on intricate surfaces.

Fig. 25. Surface-filling curves could be used for fashion design, such as an interactive body sensor network or sportswear that cools down body temperature.

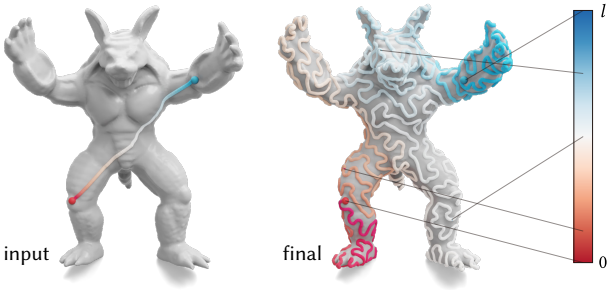


Fig. 26. Given two user-specified endpoints as an input curve, our method can compute a unicursally connected surface-filling path.

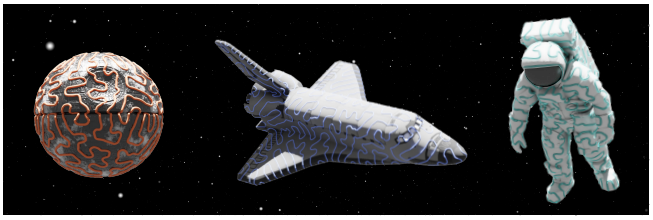
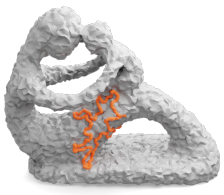


Fig. 27. One potential application of our method is a cooling or heating system deployed on complex surfaces, such as an artificial satellite, a spaceship, or a spacesuit.

5 DISCUSSION AND CONCLUSIONS

We have proposed a geometric flow that evolves a given curve on an input surface into a surface-filling curve with a user-specified thickness r. We have achieved this by defining a *medial axis* energy, which we have discretized through several approximations into an easily assemblable sparse quadratic energy whose descent direction we follow via a Newton scheme.

Our proposed algorithm achieves results qualitatively similar to the most recent work Yu et al. [2021] at a massive speed-up. However, it should be noted that surface-filling curves are only one example of the more general repulsive energy defined by Yu et al. [2021], for most of which said work remains state of the art. Specifically, a promising avenue for future work involves generalizing our approach to space-filling curves contained on volumetric domains, or even codimension-one space-filling *surfaces*.



Additionally, while our method showed excellent robustness on a diverse set of input surfaces, we observed failure in very few challenging situations where the underlying geometry has noisy normal directions (see inset). This is because our approach depends significantly on the normal direction, and we currently lack an alternative method to bypass this issue.

Also, a fundamental theoretical inquiry persists regarding the empirical relationship in Figure 17 between curve thickness r and the energy weight α.

Finally, we hope that by releasing our fast, robust, and user-controllable algorithm to the geometry processing community and beyond, we can enable a novel array of previously impracticable applications of surface-filling curves for both interactive artistic design and industrial applications at scale.

ACKNOWLEDGMENTS

We thank Chris Yu for early discussions, Abhishek Madan, Chenxi Liu, Kinjal Parikh, Victor Rong, and Zhecheng Wang for help in proofreading and rendering figures, and Esther Lin for video narration.

This work was supported in part by JST ACT-X (JPMJAX200L), JSPS KAKENHI (JP23KJ0582), NSERC Discovery (RGPIN–2022–04680), the Ontario Early Research Award program, the Canada Research Chairs Program, a Sloan Research Fellowship, an NSERC Vanier graduate scholarship, an Adobe Research fellowship, the DSI Catalyst Grant program, and gifts by Adobe Inc.

We acknowledge the authors of the 3D models used throughout this paper and thank them for making them available for academic use. Figures in this work contain the springer [TimEdwards 2023], plane [YahooJAPAN 2019], car [kotakotakota 2008], robot arm [jobojobojobo 2021], Spot [Crane 2019b], boot [inciprocal 2023], beetle [Sutherland 1972], tower [jansentee3d 2021], Inopos [CosmoWenman 2013], ouroboros [FarnkFella 2012], jellyfish [YahooJAPAN 2013a], stuffed toy [Baran and blobs 2020], cow [3dcs_ch 2023], hand [Max Planck Society e.V via SMPL 2021], bunny [The Stanford 3D Scanning Repository 2021], fish [Crane 2019a], Max Planck [MPI 2020], scorpion [YahooJAPAN 2021], fandisk [Hoppe 2020], Brucewick [Holinaty 2023], falcon statue [colinfizgig 2021], Demosthenes [ryanfb 2021], Bob [Crane [n. d.]], snail [YahooJAPAN 2013b], armadillo [The Stanford 3D Scanning Repository 2023], avatar [Adobe [n. d.]], artificial star [tato_713 2023], astronaut [MaxGrueter 2014], fertility [AIM@Shape [n. d.]], Space Shuttle [Microsoft 2014], and octopus [DiabaseEngineer 2014] meshes.

REFERENCES

3dcs_ch. 2023. Cow. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/cow, originally from thingiverse.com/thing:2431906. Asset licensed under CC BY 4.0..

Adobe. [n. d.]. Mixamo. mixamo.com

AIM@Shape. [n. d.]. Fertility.

Ergun Akleman, Qing Xing, Pradeep Garigipati, Gabriel Taubin, Jianer Chen, and Shiyu Hu. 2013. Hamiltonian cycle art: Surface covering wire sculptures and duotone surfaces. *Computers & graphics* 37, 5 (2013), 316–332.

E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. 1999. *LAPACK Users' Guide* (third ed.). Society for Industrial and Applied Mathematics, Philadelphia, PA.

Ilya Baran and Cosmic blobs. 2020. Stuffed toy. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/stuffedtoy, originally from mit.edu/~ibaran/autorig..

A. Bedel, Y. Coudert-Osmont, J. Martínez, R. I. Nishat, S. Whitesides, and S. Lefebvre. 2022. Closed space-filling curves with controlled orientation for 3D printing. *Computer Graphics Forum* 41, 2 (2022), 473–492. https://doi.org/10.1111/cgf.14488 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14488

Stephan Bischoff, Tobias Weyand, and Leif Kobbelt. 2005. Snakes on Triangle Meshes. In *Bildverarbeitung für die Medizin 2005*, Hans-Peter Meinzer, Heinz Handels, Alexander Horsch, and Thomas Tolxdorff (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 208–212.

Nicolas Boumal. 2023. *An introduction to optimization on smooth manifolds.* Cambridge University Press. https://doi.org/10.1017/9781009166164

Kenneth A Brakke. 1992. The surface evolver. *Experimental mathematics* 1, 2 (1992), 141–165.

Puikei Cheng, Wing Kam Liu, Kornel Ehmann, and Jian Cao. 2020. Enumeration of additive manufacturing toolpaths using Hamiltonian paths. *Manufacturing Letters* 26 (2020), 29–32.

Xavier Chermain, Cédric Zanni, Jonàs Martínez, Pierre-Alexandre Hugron, and Sylvain Lefebvre. 2023. Orientable Dense Cyclic Infill for Anisotropic Appearance

Fabrication. *ACM Trans. Graph.* 42, 4, Article 68 (jul 2023), 13 pages. https://doi.org/10.1145/3592412
colinfizgig. 2021. Falcon statue. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/falconstatue, originally from thingiverse.com/thing:46631. Asset licensed under CC BY-NC-SA 4.0..
CosmoWenman. 2013. Inopos. Downloaded from thingiverse.com/thing:49168. Asset licensed under CC BY 4.0..
Keenan Crane. [n. d.]. Bob. Downloaded from cs.cmu.edu/~kmcrane/Projects/ModelRepository..
Keenan Crane. 2019a. Fish. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/fish, originally from cs.cmu.edu/~kmcrane/Projects/ModelRepository..
Keenan Crane. 2019b. Spot. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/spot, originally from cs.cmu.edu/~kmcrane/Projects/ModelRepository..
Keenan Crane, Marco Livesu, Enrico Puppo, and Yipeng Qin. 2020. A Survey of Algorithms for Geodesic Paths and Distances. *CoRR* abs/2007.10430 (2020). arXiv:2007.10430 https://arxiv.org/abs/2007.10430
Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust fairing via conformal curvature flow. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.
DiabaseEngineer. 2014. Flexible Octopus. Downloaded from thingiverse.com/thing:159217. Asset licensed under CC BY 3.0..
Guoxin Fang, Tianyu Zhang, Sikai Zhong, Xiangjia Chen, Zichun Zhong, and Charlie CL Wang. 2020. Reinforced FDM: Multi-axis filament alignment with controlled anisotropic strength. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–15.
FarnkFella. 2012. Auryn Medallion from The NeverEnding Story. Downloaded from thingiverse.com/thing:49168. Asset licensed under CC BY 4.0..
Henryk Gerlach and Heiko von der Mosel. 2011. On Sphere-Filling Ropes. *Am. Math. Mon.* 118, 10 (2011), 863–876. https://doi.org/10.4169/AMER.MATH.MONTHLY.118.10.863
John Giannatsis, Aggelos Vassilakos, Vassilios Canellidis, and Vassilis Dedoussis. 2015. Fabrication of graded structures by extrusion 3D Printing. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE, 175–179.
David Hilbert. 1891. Ueber die stetige Abbildung einer Line auf ein Flächenstück. *Math. Ann.* (1891).
Josh Holinaty. 2023. Brucewick. Downloaded from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/brucewick. Asset licensed under CC BY-NC 4.0..
Hughes Hoppe. 2020. Fandisk. Downloaded modified version from common-3d-test-models https://github.com/alecjacobson/common-3d-test-models..
inciprocal. 2023. Boot. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/boot, originally from sketchfab.com/3d-models/harrison-work-boot-d9506d9a318e4c56ac3d8ab273bd2540. Asset licensed under CC BY 4.0..
Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.
jansentee3d. 2021. Tower. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/tower, originally from thingiverse.com/thing:3155868. Asset licensed under CC BY 3.0..
jobojobojobo. 2021. Industrial robot KUKA KR210 + Flange + Linear + Gripper+ Armature (Bones). Downloaded from blendswap.com/blend/29112. Asset licensed under CC-0.
Anant A Joshi, Maulik C Bhatt, and Arpita Sinha. 2019. Modification of Hilbert's space-filling curve to avoid obstacles: a robotic path-planning strategy. In *2019 Sixth Indian Control Conference (ICC)*. IEEE, 338–343.
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. Wiley Online Library, 1745–1754.
kotakotakota. 2008. Body of Nissan 350Z Low-Poly. Downloaded from turbosquid.com/3d-models/425765..
Sen Lin, Lingwei Xia, Guowei Ma, Shiwei Zhou, and Yi Min Xie. 2019. A maze-like path generation scheme for fused deposition modeling. *The International Journal of Advanced Manufacturing Technology* 104 (2019), 1509–1519.
C. Mancinelli, M. Livesu, and E. Puppo. 2021. Practical Computation of the Cut Locus on Discrete Surfaces. *Computer Graphics Forum* 40, 5 (2021), 261–273. https://doi.org/10.1111/cgf.14372 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14372
Max Planck Society e.V via SMPL. 2021. Hand. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/hand, adapted from M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, M. Black ""SMPL"" (SIGGRAPH Asia 2015). Asset licensed under CC BY 4.0..
MaxGrueter. 2014. Apollo Astronaut (The Original. Downloaded from thingiverse.com/thing:230305. Asset licensed under CC BY 4.0..
Fabian Meyer. 2019. knn-cpp. https://github.com/Rookfighter/knn-cpp.
Microsoft. 2014. Space Shuttle. Downloaded from sketchfab.com/3d-models/space-shuttle-ff4b00b7ebb24fdd98fb96b08f2c43c9. Asset licensed under CC BY 4.0..
Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. 1987. The Discrete Geodesic Problem. *SIAM J. Comput.* 16, 4 (1987), 647–668. https://doi.org/10.1137/0216045 arXiv:https://doi.org/10.1137/0216045
MPI. 2020. Max Planck. Downloaded modified version from common-3d-test-models https://github.com/alecjacobson/common-3d-test-models, originally from https://gfx.cs.princeton.edu/proj/sugcon/models/..
G Peano. 1890. Sur une courde qui remplit touteune aire plane. *Math. Ann.* 36 (1890).
Hans Pedersen and Karan Singh. 2006. Organic Labyrinths and Mazes. In *Proceedings of the 4th International Symposium on Non-Photorealistic Animation and Rendering* (Annecy, France) *(NPAR '06)*. Association for Computing Machinery, New York, NY, USA, 79–86. https://doi.org/10.1145/1124728.1124742
ryanfb. 2021. Demosthenes. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/demosthenes, originally from sketchfab.com/3d-models/louvre-demosthenes-photoscan-cf5d9dd9bf4a457c95dc76e17848727b. Asset licensed under CC BY 4.0..
P. Schmidt, J. Born, D. Bommes, M. Campen, and L. Kobbelt. 2022. TinyAD: Automatic Differentiation in Geometry Processing Made Simple. *Computer Graphics Forum* 41, 5 (2022), 113–124. https://doi.org/10.1111/cgf.14607 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14607
Christian Schüller, Roi Poranne, and Olga Sorkine-Hornung. 2018. Shape representation by zippables. *ACM Trans. Graph.* 37, 4, Article 78 (jul 2018), 13 pages. https://doi.org/10.1145/3197517.3201347
Silvia Sellán, Christopher Batty, and Oded Stein. 2023. Reach For the Spheres: Tangency-aware surface reconstruction of SDFs. , Article 73 (2023), 11 pages. https://doi.org/10.1145/3610548.3618196
Silvia Sellán, Jacob Kesten, Ang Yan Sheng, and Alec Jacobson. 2020. Opening and Closing Surfaces. *ACM Trans. Graph.* 39, 6, Article 198 (nov 2020), 13 pages. https://doi.org/10.1145/3414685.3417778
Nicholas Sharp and Keenan Crane. 2018. Variational Surface Cutting. *ACM Trans. Graph.* 37, 4, Article 156 (jul 2018), 13 pages. https://doi.org/10.1145/3197517.3201356
Nicholas Sharp and Keenan Crane. 2020. You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges. *ACM Trans. Graph.* 39, 6, Article 249 (nov 2020), 15 pages. https://doi.org/10.1145/3414685.3417839
Nicholas Sharp, Keenan Crane, et al. 2019. GeometryCentral: A modern C++ library of data structures and algorithms for geometry processing. https://geometry-central.net/. (2019).
Ivan Sutherland. 1972. Sutherland's Volkswagen.
tato_713. 2023. Death Star scaled one in two million. Downloaded from printables.com/model/57266. Asset licensed under CC BY 4.0..
Gabriel Taubin. 1995. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 351–358.
The Stanford 3D Scanning Repository. 2021. Bunny. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/bunny. Original mesh at graphics.stanford.edu/data/3Dscanrep..
The Stanford 3D Scanning Repository. 2023. Armadillo. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/armadillo. Original mesh by V. Krishnamurthy and M. Levoy ""Fitting Smooth Surfaces to Dense Polygon Meshes"" (SIGGRAPH 1996).
TimEdwards. 2023. Springer. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/springer, originally from thingiverse.com/thing:335658. Asset licensed under GPLv2.
Thibault Tricard, Jimmy Etienne, Cédric Zanni, and Sylvain Lefebvre. 2021. A brick in the wall: Staggered orientable infills for additive manufacturing. In *Proceedings of the 6th Annual ACM Symposium on Computational Fabrication*. 1–8.
Qing Xing, Ergun Akleman, Gabriel Taubin, and Jianer Chen. 2012. Surface covering curves. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*. 107–114.
YahooJAPAN. 2013a. Jellyfish. Downloaded from https://www.thingiverse.com/thing:182222. Asset licensed under CC BY 4.0..
YahooJAPAN. 2013b. Snail. Downloaded from thingiverse.com/thing:182275. Asset licensed under CC BY 4.0..
YahooJAPAN. 2019. Plane. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/plane, originally from thingiverse.com/thing:182252. Asset licensed under CC BY 3.0..
YahooJAPAN. 2021. Scorpion. Downloaded modified version from odedstein-meshes github.com/odedstein/meshes/tree/master/objects/scorpion, originally from thingiverse.com/thing:182363. Asset licensed under CC BY 3.0..
Yuan Yan and Yasamin Mostofi. 2016. Efficient clustering and path planning strategies for robotic data collection using space-filling curves. *IEEE Transactions on Control of Network Systems* 4, 4 (2016), 838–849.
Chris Yu, Henrik Schumacher, and Keenan Crane. 2021. Repulsive Curves. *ACM Trans. Graph.* 40, 2, Article 10 (may 2021), 21 pages. https://doi.org/10.1145/3439429
Cem Yuksel. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Computer Graphics Forum* 34, 2 (2015), 25–32. https://doi.org/10.1111/cgf.12538 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12538

Haisen Zhao, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2016. Connected Fermat Spirals for Layered Fabrication. *ACM Trans. Graph.* 35, 4, Article 100 (jul 2016), 10 pages. https://doi.org/10.1145/2897824.2925958

Haisen Zhao, Hao Zhang, Shiqing Xin, Yuanmin Deng, Changhe Tu, Wenping Wang, Daniel Cohen-Or, and Baoquan Chen. 2018. DSCarver: Decompose-and-Spiral-Carve for Subtractive Manufacturing. *ACM Trans. Graph.* 37, 4, Article 137 (jul 2018), 14 pages. https://doi.org/10.1145/3197517.3201338

Fanchao Zhong, Yonglai Xu, Haisen Zhao, and Lin Lu. 2023. As-Continuous-As-Possible Extrusion-Based Fabrication of Surface Models. *ACM Trans. Graph.* 42, 3, Article 26 (mar 2023), 16 pages. https://doi.org/10.1145/3575859

A ENERGY MODIFICATIONS

A.1 Field-alignment energy

A user may wish to align the curve to a vector field X on the input surface by adding a simple energy

$$\beta_F \sum_{i=1}^{n-1} \frac{1}{l_i} \left\| (R_i\gamma_i - Q_i\gamma_{i+1}) \times (R_iX_i + Q_iX_{i+1}) \right\|_2^2, \tag{11}$$

where $X_{i,i+1}$ is the vertex-blended vector field on $\gamma_{i,i+1}$ and β_F is the weight which we set to 0.5 to 1 in our experiments. Intuitively, it encourages the segment $\gamma_i\gamma_{i+1}$ to be parallel to X_i and X_{i+1} in the rigidly transformed plane.

A.2 Biharmonic bending energy

A user may also wish to encourage straighter segments by adding a biharmonic bending energy. Let θ_i be the angle of $\angle\gamma_{i-1}\gamma_i\gamma_{i+1}$ projected onto the tangent space of γ_i, and P_i be the rigid transformation matrix that maps γ_{i-1} to the point $(l_{i-1}\cos\theta_i, l_{i-1}\sin\theta_i, 0)$ while aligning the normal vector N_{i-1} with the z-axis. Intuitively, P_i maps γ_{i-1} onto the same plane as we evaluate the geodesic curve length of $\gamma_i\gamma_{i+1}$. We formulate our biharmonic bending energy as

$$\beta_B \sum_{i=1}^{n-1} \frac{l_{i-1}+l_i}{2} \left\| \frac{P_i\gamma_{i-1} - R_i\gamma_i}{l_{i-1}} - \frac{R_i\gamma_i - Q_i\gamma_{i+1}}{l_i} \right\|_2^2, \tag{12}$$

where β_B is the energy weight which we set to 10 to 1000 in our experiments. We note that both the field-alignment and biharmonic bending energies are sparse and quadratic, which does not have a large effect on the runtime.

A.3 Power of the energy

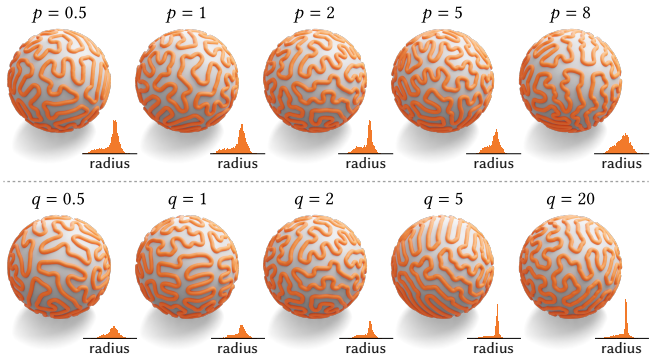


Fig. 28. We can obtain curves with varying global shapes by changing the power orders p and q of the two energy terms (see Eq. (13)).

A user can change the global shape of the curve by changing the square order of the energies into more general p and q powers,

$$\sum_i \frac{1}{l_i} \left\| R_i\gamma_i - Q_i\gamma_{i+1} \right\|_2^p + \sum_{i=1}^{n} \left(\left\| \gamma_i - m_i'^{+} \right\|_2^q + \left\| \gamma_i - m_i'^{-} \right\|_2^q \right), \tag{13}$$

in which case the Hessian and gradient required for the descent direction can be computed via automatic differentiation. As shown in Figure 28, we observed that smaller p and larger q encourage round corners and sharpen the radius histogram.

B DISCUSSION ON THE GRADIENT

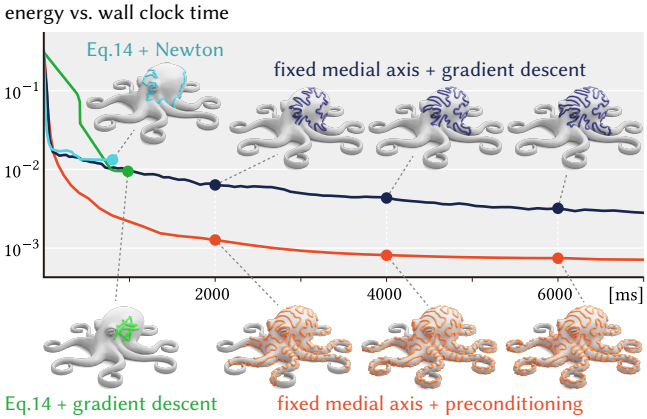


Fig. 29. Using our approximated Hessian as a preconditioner leads to robust and faster convergence, compared to treating the medial axis position as a function as in Eq. 14 or conducting the mere gradient descent as in Eq. 15.

In our gradient computation, we apply two modifications to the mere gradient descent of Eq. 2; we (1) fix the unrolled medial axis $m_i'^{\pm}$ for each step and (2) use the inverse of the (approximated) Hessian $\mathbf{H}^k$ as a preconditioner.

If we do not fix $m_i'^{\pm}$, Eq. 7 can be rewritten as

$$E_M(\boldsymbol{\gamma}) \approx \sum_{i=1}^{n} \frac{l_{i-1}+l_i}{2} \left(\left\| r_i^+(\gamma_i) \right\|_2^2 + \left\| r_i^-(\gamma_i) \right\|_2^2 \right), \tag{14}$$

where $r_i^{\pm}(\gamma_i) = \frac{\|\gamma_j - \gamma_i\|_2^2}{(\gamma_j - \gamma_i)\cdot(\pm B_i)}$ is the radius of the Euclidean medial sphere grown from γ_i that touches γ_j. However, we observed that the gradient directions become so noisy that even the smallest step would cause self-intersection, which completely stops the flow due to our intersection avoidance (Figure 29).

Under fixing $m_i'^{\pm}$, one can also conduct the mere gradient descent

$$\mathbf{d}^k = -\mathbf{g}^k, \tag{15}$$

equipped with an Armijo line search. While this gives qualitatively similar results, our preconditioned gradient evolves faster, especially in regions with thin features (see Figure 29).

Additionally, our approximated Hessian is banded near the diagonals, especially when the curve is singly connected and has an open end. In this case, a user can take advantage of a Cholesky solver tailored for symmetric banded matrices (e.g., the `dpbtrf` function in LAPACK [Anderson et al. 1999]). In practice, we observed an

approximately 30% speed up in the linear solve compared to using a normal sparse Cholesky solver.

C COMPARISON WITH [Yu et al. 2021]

The algorithm of Yu et al. [2021] achieved a significant speed-up thanks to their multigrid method. However, they neither explained nor implemented how to apply it to the surface constraint. To approximate the effect of the multigrid method in Figure 5, we first measured the runtime with and without the multigrid method on an example that is supported, and then multiplied the speed-up factor by the runtime with the surface constraint.