A Heat Method for Generalized Signed Distance

NICOLE FENG and KEENAN CRANE, Carnegie Mellon University

We introduce a method for approximating the signed distance function (SDF) of geometry corrupted by holes, noise, or self-intersections. The method implicitly defines a completed version of the shape, rather than explicitly repairing the given input. Our starting point is a modified version of the *heat method* for geodesic distance, which diffuses normal vectors rather than a scalar distribution. This formulation provides robustness akin to *generalized winding numbers (GWN)*, but provides distance function rather than just an inside/outside classification. Our formulation also offers several features not common to classic distance algorithms, such as the ability to simultaneously fit multiple level sets, a notion of distance for geometry that does not topologically bound any region, and the ability to mix and match signed and unsigned distance. The method can be applied in any dimension and to any spatial discretization, including triangle meshes, tet meshes, point clouds, polygonal meshes, voxelized surfaces, and regular grids. We evaluate the method on several challenging examples, implementing normal offsets and other morphological operations directly on imperfect curve and surface data. In many cases we also obtain an inside/outside classification dramatically more robust than the one obtained provided by GWN.

CCS Concepts: • **Computing methodologies → Shape analysis**; • **Mathematics of computing → Partial differential equations**.

Additional Key Words and Phrases: geometry processing, signed distance

ACM Reference Format:
Nicole Feng and Keenan Crane. 2024. A Heat Method for Generalized Signed Distance. *ACM Trans. Graph.* 43, 4, Article 92 (July 2024), 20 pages. https://doi.org/10.1145/3658220

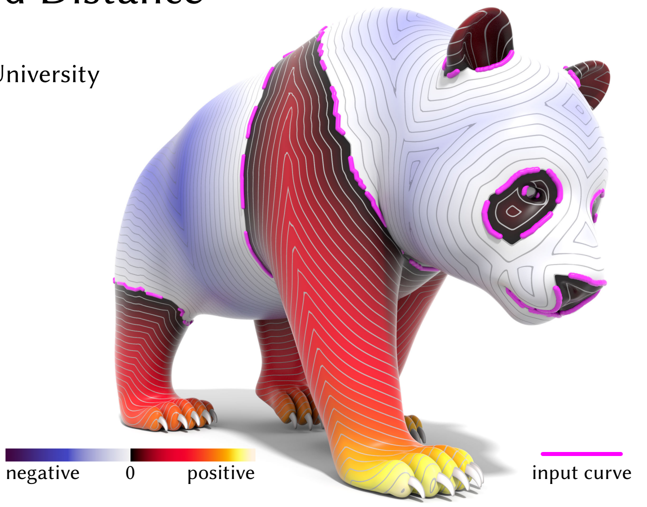


Fig. 1. The signed heat method computes a well-behaved signed distance function (SDF), even for imperfect or incomplete input geometry. Here we compute geodesic signed distance to the broken curve shown in magenta.

1 INTRODUCTION

A *signed distance function (SDF)* describes the distance to the closest point on the boundary of a given shape, using sign to indicate whether points are inside or outside. SDFs are a basic component of numerous algorithms from geometric modeling [Museth et al. 2002], physical simulation [Osher et al. 2004], rendering [Quilez 2008], path planning [Oleynikova et al. 2016], geometric learning [Yariv et al. 2023] and computer vision [Vicini et al. 2022]. In many problems, geometry exhibits severe defects due to errors in modeling or acquisition. Unfortunately, traditional SDF algorithms assume a "perfect" watertight description of the boundary, relying on a well-defined inside and outside to determine the sign [Sethian 1999]. At the same time, robust inside/outside tests do not provide a useful signed distance approximation [Jacobson et al. 2013]. Our method provides the best of both worlds: a *generalized signed distance (GSD)* function computed directly from broken geometry, which provides not only an inside/outside classification, but also distance information.

Authors' address: Nicole Feng; Keenan Crane, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213.

More precisely, we aim to approximate the signed distance to the boundary of a shape within a fixed domain M—for example, a curve on a surface, or a surface in $\mathbb{R}^3$. When M is itself a curved manifold, we consider the signed *geodesic distance*, *i.e.*, the minimal length among all paths along M. We assume we are given as input a defective description $\Omega \subset M$ of the shape boundary: a "broken" curve or surface which can have holes, self-intersections, and noise. We then seek a reasonable approximation of the SDF for the true (unknown) geometry.

Our solution to this problem, the *signed heat method (SHM)*, provides a "least squares" approach: we find a signed distance approximation that matches the input geometry as well as possible, even when it does not perfectly bound any region. There are three basic steps (Figure 2):

I. Diffuse the normals of Ω for a small time $t > 0$, by solving a vector-valued heat equation.
II. Normalize the diffused vectors, yielding a unit vector field that approximates the gradient of the (unknown) SDF.
III. Find the function whose gradient best matches the unit vector field, by solving a scalar Poisson equation.

A diffusion-based approach is valuable because it extends in a robust way to imperfect geometry. For instance, if a curve has gaps, or a surface has holes, diffusion averages together normals at nearby points—providing smooth interpolation of the observed data. Normalization of gradients then ensure that we recover a distance approximation, rather than just a smooth function.

Why not simply "fix" the broken geometry before computing distance? For example, one could fit an implicit function, extract an explicit mesh, then compute signed distance using any standard algorithm. Here, however, errors in reconstruction can propagate into distance computation (Figure 3, *left*). Moreover, each step (say, computing signed distance) already costs as much as our entire

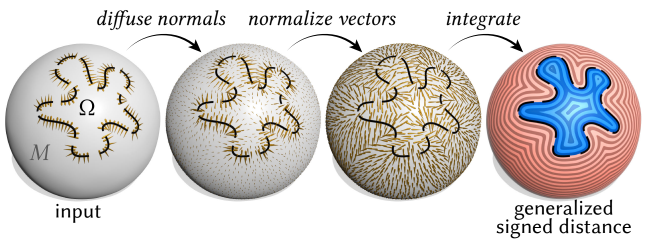


Fig. 2. The three basic steps of the signed heat method.

method. We avoid intermediate representations altogether, providing more accurate distance for about an order of magnitude lower cost (Section 9.5.3).

By formulating our method from the perspective of short-time heat diffusion, it also inherits many benefits of the original heat method: it can compute (geodesic) distance on curved domains, it applies directly to most discretizations (triangle/tet meshes, general polygon meshes, point clouds, *etc.*), is robust to noise in both the input geometry and the underlying domain (since it is based on solving "nice" elliptic problems), and is easily accelerated and parallelized using existing systems for sparse linear algebra (since the main cost is solving two standard linear systems). More broadly, a variational, PDE-based approach to SDF computation enables extensions not possible even with other geodesic distance algorithms—such as finding the "best fit" distance function for a collection of partially-observed level sets (Section 7.1), or computing a notion of generalized signed distance for input shapes that do not topologically bound regions (Section 7.2).

2 RELATED WORK

Our algorithm complements a large set of existing methods:

- Methods for robust inside/outside classification, which do not directly provide signed distance (Section 2.1).
- Methods for signed distance computation, which generally do not work well for broken geometry (Section 2.2).

It also of course builds on earlier heat methods (Section 2.3). Unfortunately, simply signing the unsigned distance yields a function completely different from the true SDF—see Figure 6. In general, there are only a few methods that explore *robust* SDF computation—and none suitable for curved geometry.

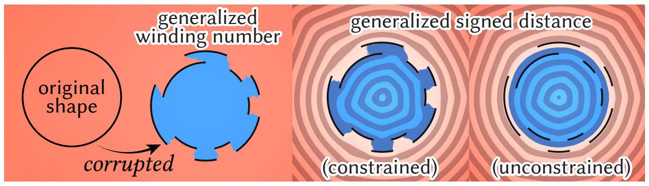


Fig. 3. Both GWN and GSD implicitly define a completed surface—but GSD also provides distance information. Moreover, GWN must interpolate the input geometry, leading to noisy output *(left)*. GSD can either interpolate or approximate the input, yielding a more faithful completion *(far right)*.

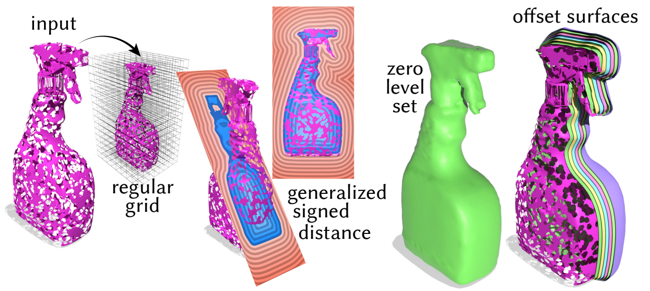
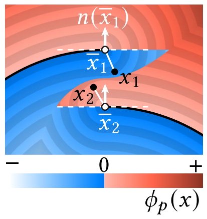


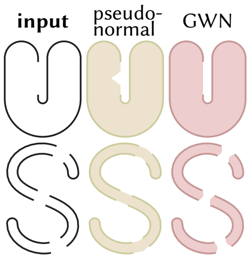
Fig. 4. Our method can be applied on virtually any spatial data structure, in any dimension. Here for instance we compute generalized signed distance to a badly broken surface *(left)*, by solving on a regular grid in $\mathbb{R}^3$. Contouring this function yields well-behaved and evenly-spaced offset surfaces *(right)*.

2.1 Inside/Outside Classification

Given a corrupted region boundary ∂A, there are two basic strategies for estimating whether a given point $x \in M$ is inside or outside:



- **Pseudonormal test.** Let $n(\overline{x})$ be the normal at the closest point $\overline{x} \in \partial A$. The sign of the function $\phi_p(x) := \langle n(\overline{x}), x - \overline{x} \rangle$ indicates whether x is inside or outside A [Bærentzen and Aanæs 2005]. This test can be applied as-is to broken geometry, effectively using the tangent plane of the closest boundary point as a proxy for missing geometry. However, since tangent planes are not globally consistent, neither is the *pseudonormal distance* ϕ_p, which in general is not even C^0 continuous (inset).
- **Winding number.** The *winding number* captures the number of times ∂A wraps around the point x, and is hence nonzero only when x is inside A. It is equivalent (up to a constant) to signed solid angle—known in graphics as the *generalized winding number (GWN)* [Jacobson et al. 2013], which is well-defined even for broken geometry. GWN supplies robust inside/outside queries in several algorithms [Zhou et al. 2016; Hu et al. 2018], but defines a harmonic function rather than an SDF—and hence cannot be used for a broad variety of geometric tasks (*e.g.*, Figure 26, *top*).



Each method implicitly applies a prior (linear vs. harmonic extension), suitable in different scenarios (inset). GSD effectively interpolates between these options: as $t \to 0$, points inherit the normal at the closest point; as $t \to \infty$, diffused vectors become componentwise harmonic (Figure 5). Yet unlike the pseudonormal, GSD is based on global integration, making it less sensitive to small perturbations. Unlike GWN, which must (by definition) interpolate the input, GSD is robust to noise in positions (Figure 3), and yields better surface completions due to smooth extrapolation of normal information (Figure 27).

2.1.1 Curved Domains. Inside/outside tests are also not easily generalized to curved domains. Here, closest point queries entail computing the geodesic distance at every point—at which point one may as well compute signed distance rather than the pseudonormal. Likewise, the recent method of Feng et al. [2023] generalizes GWN

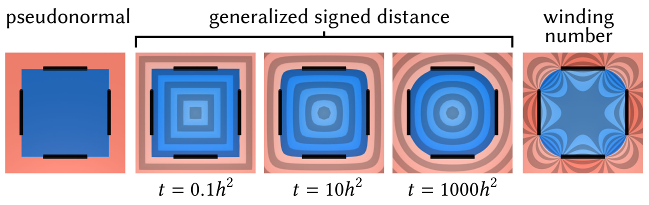


Fig. 5. For different diffusion times, our method effectively interpolates between the "linear" prior used by the pseudonormal test (as $t \to 0$) and the "harmonic" prior used by generalized winding numbers (as $t \to \infty$).

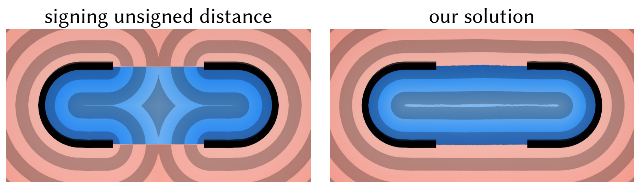


Fig. 6. For broken geometry, simply signing unsigned distance does not work, yielding completely different level sets *(left)* from the distance to the unbroken curve *(right)*.

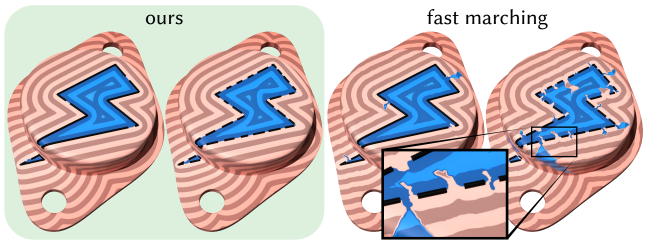


Fig. 7. Our elliptic formulation regularizes the available information, gracefully handling broken curves *(left)*. In contrast, hyperbolic methods like fast marching consider an expanding wavefront that propagates sign and distance errors *(right)*.

to curved surfaces, but again provides no notion of distance (and has a similar cost to GSD).

2.2 Signed Distance Computation

For watertight geometry in $\mathbb{R}^n$, *unsigned* distance can be computed via fast, exact closest point queries [Sawhney et al. 2020] and signed via basic inside-outside tests like ray shooting [Haines 1994]. Alternatively, one can sample geometry onto a grid and use methods like fast sweeping [Osher et al. 2004].

For broken geometry, it is tempting to again just sign the unsigned distance—but unless gaps are very small, the resulting function will be quite different from the true SDF (Figure 6). Likewise, wavefront-based methods like *fast marching* [Kimmel and Sethian 1998] and learning-based variants [Lichtenstein et al. 2019; Huberman et al. 2023] can be executed on broken geometry, but propagation of orientation errors yields significant artifacts (Figure 7).

Some "signed distance" methods do not actually compute a true distance function, but rather just a smooth implicit function that vanishes near the input geometry. Like GWN, such functions are unreliable for tasks like morphological modeling or sphere tracing [Hart 1996]. For instance, *smooth signed distance (SSD)* [Calakli and Taubin 2011] is actually a biharmonic function with unit-norm gradient only at the zero set. Likewise, many so-called *neural SDFs* encourage the signed distance property only near the zero set, or weakly enforce the eikonal condition $|\nabla\phi| = 1$ [Gropp et al. 2020], which is insufficient to characterize SDFs [Marschner et al. 2023].

2.2.1 Robust Signed Distance.

Some past work considers regularized signed distance for broken geometry. *E.g.*, Xu and Barbič [2014] effectively apply morphological operations on unsigned distance to "heal" holes of a user-specified size—at the cost of losing small details. Brunton and Rmaileh [2021] use smoothing to mitigate artifacts arising from pseudonormal-like sign estimation (Section 2.1); Bærentzen [2005] likewise smooth a pseudonormal distance obtained from voxelization. Mullen et al. [2010] sign the unsigned distance by performing ray intersection tests on a small band around the zero set. As noted above, neural networks have also been used to fit distance-like functions to noisy/incomplete input [Park et al. 2019; Atzmon and Lipman 2019; Gropp et al. 2020], but do not produce true SDFs even for watertight geometry. Unlike GSD, none of the methods in this section apply to curved domains, nor generalize to alternative spatial discretizations, nor handle the richer constraints furnished by our variational formulation.

2.2.2 Geodesic Distance.

With few exceptions, work over the past 40 years on geodesic distance computation has aimed largely at improving speed and accuracy—Crane et al. [2020] provides a detailed survey. We focus not on this performance race, but rather on a more *robust* notion of signed distance that can be applied in a broader range of real-world scenarios. However, robustness does not come at a major cost: our method remains competitive with widely-used methods such as the heat method and fast marching (Section 9.5).

Most past work considers distance to isolated points, which is always unsigned, though a few methods consider unsigned distance to curves embedded in surfaces [Bommes and Kobbelt 2007; Trettner et al. 2021]; as in $\mathbb{R}^n$, signing unsigned distance will fail to yield satisfactory results for broken curves. Overall, we are not aware of any work that computes robust signed distance to broken curves on surfaces, or more generally, submanifolds of a curved domain.

2.3 Heat Methods

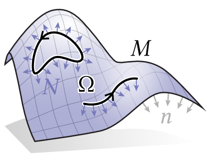
The signed heat method (SHM) builds on the *unsigned heat method (UHM)* of Crane et al. [2013b], and the *vector heat method (VHM)* of Sharp et al. [2019c]. UHM diffuses a scalar distribution concentrated on source points for a small time $t > 0$; *Varadhan's formula* then implies that, as $t \to 0$, the gradient of the resulting heat distribution becomes parallel with the gradient of the distance function ϕ [Varadhan 1967]. Since the true distance function satisfies the *eikonal property* $|\nabla\phi| = 1$, normalizing and integrating the initial gradient field (via a Poisson equation) recovers an accurate distance approximation. VHM instead starts by diffusing tangent vectors at points; a similar normalization yields the *parallel transport* of these vectors along minimal (shortest) geodesics.

Relative to UHM we change only the first step, diffusing normal vectors concentrated along curves or surfaces, rather than a scalar measure concentrated at points. Relative to VHM, we change its last three steps, integrating (normalized) diffused vectors to obtain distance. Our basic insight, not considered in prior work, is that diffused normals will be parallel to the gradient of *signed* distance, since parallel transport ensures they are tangent to *oriented* minimal geodesics (Section 3.1). Hence, normalizing and integrating these vectors recovers an accurate SDF approximation. For broken geometry the story remains largely the same, except that diffusion effectively interpolates normals from nearby points to obtain well-behaved gradients.

Overall, our method inherits many benefits of past heat methods: for instance, it is not tied to a particular spatial discretization (Section 8), and is robust not only to broken source geometry, but also poor discretization of the domain itself (Section 9.4). Other work improves or applies heat methods in various ways—for instance, Sharp et al. [2019b] and Gillespie et al. [2021] show how the accuracy and robustness of the heat method can be significantly improved via use of *intrinsic triangulations*—a strategy we also employ here. Several works improve efficiency and scalability via parallel or iterative solvers [Tao et al. 2019; Rawat and Biswas 2022a,b], Belyaev et al. [2013] improve accuracy by iterating the integration step, Sun and Liu [2022] extend the method to inhomogeneous and anisotropic metrics, and Litman and Bronstein [2016] approximate all-pairs distance using spectral methods. These extensions offer rich opportunities for future improvement and generalization of the signed heat method.

3 SMOOTH THEORY

Fundamentally, our algorithm is defined in terms of operations in the continuous setting, as described here. It can then be discretized in many different ways, as explored in Sections 5 and 8.



Throughout we consider an n-dimensional Riemannian manifold M with metric g, and want to compute the signed distance function ϕ for a codimension-1 submanifold $\Omega \subset M$ (*e.g.*, curves within a surface, or surfaces within a volume). In general we assume that this data might represent a corrupted version of ideal input, meaning M and/or Ω may have holes, self-intersections, noise, and may not be consistently oriented. (We also consider the more general case where Ω can include isolated points—see Appendix A.1.) We use N to denote the unit normals of Ω, and n for the unit normals of the domain boundary ∂M. We use μ_Ω to denote a measure concentrated on Ω, similar in spirit to an indicator function[1]; $N\mu_\Omega$ is likewise a vector field (or vector measure) equal to zero away from Ω, and determined by N for points in Ω. Finally, we use Δ for the negative-definite Laplace-Beltrami operator on M, and Δ^∇ for the negative-definite *connection Laplacian* [Gallier et al. 2020]; intuitively, these operators measure the deviation of a scalar function and tangent vector field (*resp.*) from their average in a local neighborhood.

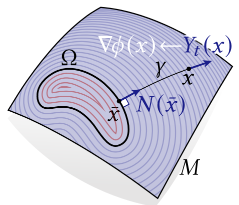
[1]More formally, for any Borel measurable set $U \subset M$, $\mu_\Omega(U) := \int_{\Omega\cap U} dV$, where dV is the usual volume measure on Ω.

3.1 Signed Heat Method

The first step of our algorithm is to diffuse the normals N from Ω to the rest of the domain M for a short time $t > 0$, which extends information about surface orientation to the rest of the domain. In particular we solve a vector-valued diffusion equation

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}t}X_t &= \Delta^\nabla X_t, \quad t > 0, \\ X_0 &= N\mu_\Omega. \end{aligned} \tag{1}$$

On $\mathbb{R}^n$ this equation simply diffuses each scalar component of the vector field; more generally it accounts for how vectors diffuse across a domain with curvature (see Sharp et al. [2019c, Section 4.3]). As with scalar diffusion, the magnitude of X_t decays exponentially with distance from Ω, but its direction remains well-defined almost everywhere since the support of the vector heat kernel is all of M.



Using diffusion to extrapolate orientation is not a heuristic, but is rather motivated by a key observation from differential geometry: as $t \longrightarrow 0$ the diffused vector $X_t(x)$ at each point $x \in M$ becomes parallel to the normal $N(\bar{x})$ at the closest point $\bar{x} \in \Omega$. More precisely, it aligns with the vector obtained via parallel transport of $N(\bar{x})$ along a minimal geodesic γ [Berline et al. 1992, Theorem 2.30]. Since parallel transport along geodesics preserves tangency, this vector will be tangent to γ itself—and since traveling along γ is the quickest way back to Ω, it must be parallel to the unsigned distance gradient $\nabla\phi$. Moreover, since we transport *oriented* normals, we get the correct sign. We can hence normalize X_t to obtain an approximation $Y_t := X_t/\|X_t\|$ of the signed distance gradient.

The vector field Y_t will not describe exact gradients for any SDF, due to both the diffusion approximation—and more significantly—errors in the input. However, we can still look for the function ϕ whose gradient is as close as possible, in a least-squares sense, to Y_t. In particular, we seek a minimizer for the problem

$$\min_{\phi:\, M\to\mathbb{R}} \int_M \|\nabla\phi - Y_t\|^2. \tag{2}$$

Using integration by parts, one can show that a minimizer satisfies the Poisson equation

$$\begin{aligned} \Delta\phi &= \nabla\cdot Y_t \ \text{ on } M \\ \tfrac{\partial\phi}{\partial n} &= n\cdot Y_t \ \text{ on } \partial M, \end{aligned} \tag{3}$$

whose solution is determined up to a constant shift. To exactly interpolate the input (and get a unique solution) we could also require that $\phi = 0$ along Ω—though if the input is corrupted, interpolation may be ill-advised (Figure 3); see Section 7.1 for further discussion.

In summary, we arrive at the following algorithm:

(1) Solve a vector diffusion equation $\frac{d}{dt}X_t = \Delta^\nabla X_t$ (Equation 1).
(2) Evaluate the vector field $Y_t = X_t/\|X_t\|$.
(3) Solve a Poisson equation $\Delta\phi = \nabla\cdot Y_t$ (Equation 3).

4 TIME DISCRETIZATION

As in past heat methods we discretize Equation 1 in time via one step of backward Euler [Crane et al. 2013b; Sharp et al. 2019c], and solve a linear equation

$$(\mathrm{id} - t\Delta^\nabla)X_t = X_0 \tag{4}$$

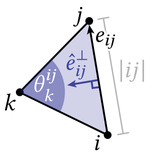
for a single, fixed time $t > 0$ (where id is the identity). To get a feel for the behavior of this equation, we can consider the Euclidean case ($M = \mathbb{R}^n$), where applying Δ^∇ to a vector field is equivalent to applying the scalar Laplacian Δ to each component. Then as $t \longrightarrow \infty$ we approach a standard Poisson equation; moreover, since $X_0 = N\mu_\Omega$ is zero almost everywhere, the solution will look nearly harmonic away from Ω.

5 SPATIAL DISCRETIZATION

To implement the signed heat method on any geometric data structure, one need only provide sparse matrices representing the Laplacian, connection Laplacian, and divergence operator. Section 8 explores several possibilities (regular grid, polygon mesh, point cloud, etc.); in this section we focus on triangle meshes.

Here we use edge-based operators for the vector diffusion step (Step I), which makes it straightforward to discretize curve sources; we then use standard vertex-based operators for the Poisson equation (Step III), so that our final SDF is stored at vertices. As in Djerbetian and Chen [2016] we adopt a complex encoding of the connection Laplacian), which uses half as much storage/bandwidth as the real-valued version used by Stein et al. [2020]: each 2x2 real block (four floats) is replaced by a single complex value (two floats).

5.1 Notation

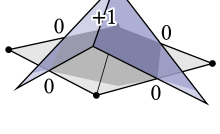


Our domain is a triangle mesh $M = (V, E, F)$ with arbitrary topology (*e.g.*, nonmanifold, nonorientable, and/or with boundary); we use C for the set of all triangle corners. We denote k-simplices by $(k+1)$-tuples of vertex indices, *i.e.*, vertices $i \in V$, edges $ij \in E$, and faces $ijk \in F$. Likewise, we denote the corner of triangle ijk at vertex i as $^{jk}_{i} \in C$. These indices are also used to express quantities stored on mesh elements—for instance, corner angles are denoted by θ^{jk}_i. We use $<$ and $>$ to indicate summation over all elements contained by or containing another element (*resp.*). For instance $\sum_{ijk>ij}$ sums over all triangles ijk containing edge ij. We use $|\cdot|$ to denote the volume of a simplex—*e.g.*, $|ij|$ is an edge length and $|ijk|$ is a triangle area. To avoid trigonometric functions, we evaluate all angle cotangents via the formula

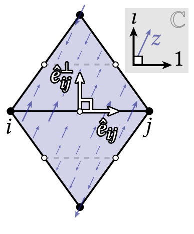
$$\cot\theta^{jk}_i = (|ij|^2 - |jk|^2 + |ki|^2)/(4|ijk|). \tag{5}$$

For each edge ij, we let e_{ij} be a vector parallel to the edge with arbitrary (but fixed) orientation, and magnitude equal to the edge length. We let $e^\perp_{ij}$ be the 90° rotation of e_{ij} in the counter-clockwise direction, and use $\hat{e}_{ij}, \hat{e}^\perp_{ij}$ for the corresponding unit vectors.

5.2 Edge Basis Functions



The *Crouzeix-Raviart (CR) basis functions* associate each edge $ij \in E$ with a face-wise linear function $\varphi_{ij} : M \longrightarrow \mathbb{R}$ interpolating the value 1 at the midpoint of ij, and 0 at all other edge midpoints (inset).



A corresponding basis for vector fields is expressed by identifying tangent vectors with complex numbers. In particular, at each edge ij we choose a coordinate system such that 1 and the imaginary unit $\imath$ correspond to $\hat{e}_{ij}$ and $\hat{e}^\perp_{ij}$, *resp.* The function $\psi_{ij} := \varphi_{ij} + 0\imath$ then defines a basis vector field parallel to the edge, and $z\psi_{ij}$ describes a locally supported vector field parallel to $z \in \mathbb{C}$ (right inset).

5.3 Edge-Based Laplacian and Mass Matrix

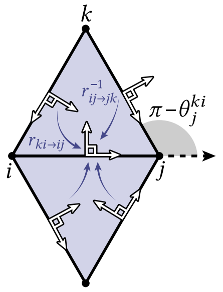
If we use the CR bases to discretize the scalar Laplacian (via a standard Ritz-Galerkin approach), we get a real-valued positive semidefinite matrix $\mathsf{L} \in \mathbb{R}^{|E|\times|E|}$ with nonzero entries

$$\begin{aligned} \mathsf{L}_{ij,jk} &= -2\cot\theta^{ki}_j, && \forall ij \in E,\ ijk > ij, \\ \mathsf{L}_{ij,ki} &= -2\cot\theta^{jk}_i, && \forall ij \in E,\ ijk > ij, \\ \mathsf{L}_{ij,ij} &= -\textstyle\sum_{ijk>ij} \mathsf{L}_{ij,jk} + \mathsf{L}_{ij,ki}, && \forall ij \in E, \end{aligned} \tag{6}$$

where either the length/area or angle-based expressions can be used. The mass matrix $\mathsf{M} \in \mathbb{R}^{|E|\times|E|}$ is diagonal, with nonzeros

$$\mathsf{M}_{ij,ij} = \tfrac{1}{3}\textstyle\sum_{ijk>ij} |ijk|, \quad \forall ij \in E. \tag{7}$$

5.4 Edge-Based Connection Laplacian



Intuitively, the connection Laplacian measures the deviation of a vector field from its average value in a small local neighborhood. To compute this average on a mesh, we must parallel transport vectors into a common coordinate system at each edge (see inset). Since edges are intrinsically flat, parallel transport is simply a translation—but we must also account for a change of coordinates. In particular, for each triangle ijk, we encode the parallel transport of a vector from edge ij to the next edge jk in counter-clockwise order as a rotation

$$r_{ij\to jk} := s_{ij\to jk}e^{-\imath(\pi-\theta^{ki}_j)} = -s_{ij\to jk}e^{\imath\theta^{ki}_j}, \tag{8}$$

where the sign $s_{ij\to jk} \in \{+1, -1\}$ is equal to $e^{\imath 0} = +1$ if ij and jk have the same relative orientation with respect to ijk, and $e^{\imath\pi} = -1$ otherwise. The positive-semidefinite connection Laplacian matrix $\mathsf{L}^\nabla \in \mathbb{C}^{|E|\times|E|}$ is then obtained by multiplying off-diagonal entries of L by edge rotations $r_{ij,jk}$, yielding nonzero entries

$$\begin{aligned} \mathsf{L}^\nabla{}_{ij,jk} &= -2r^{-1}_{ij\to jk}\cot\theta^{ki}_j, && \forall ij \in E,\ ijk > ij, \\ \mathsf{L}^\nabla{}_{ij,ki} &= -2r_{ki\to ij}\cot\theta^{jk}_i && \forall ij \in E,\ ijk > ij, \\ \mathsf{L}^\nabla{}_{ij,ij} &= \mathsf{L}_{ij,ij}, && \forall ij \in E, \end{aligned} \tag{9}$$

with cotans evaluated as in Equation 5. Since $r_{ij\to jk} = r^{-1}_{jk\to ij} = \overline{r}_{ij\to jk}$, the matrix L^∇ is Hermitian. It is positive semidefinite because it is the Hessian of a convex energy [Stein et al. 2020, Equation 5].

5.5 Vertex-Based Laplacian and Divergence

Our vertex-based operators are identical to those used in Step III of the unsigned heat method [Crane et al. 2013b, Section 3.2.1]. In

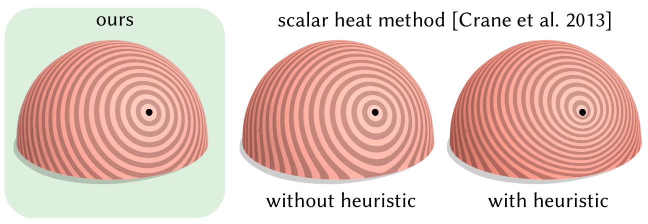
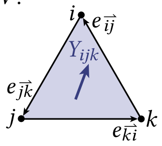


Fig. 8. The unsigned heat method exhibits bias near the domain boundary for large diffusion times (figure reproduced from Crane et al. [2013b], Figure 11 using $t = 100h^2$). Using their proposed boundary condition heuristic only slightly improves results. In contrast, our method has correct boundary conditions.

particular, we use the *cotan Laplacian* $\mathsf{C} \in \mathbb{R}^{|V|\times|V|}$ [Crane et al. 2013a, Chapter 6], which has nonzero entries

$$\begin{aligned} \mathsf{C}_{i,j} &= -\tfrac{1}{2}\textstyle\sum_{ijk>ij} \cot\theta_k^{ij}, & \forall ij \in E \\ \mathsf{C}_{i,i} &= -\textstyle\sum_{ij>i} \mathsf{C}_{i,j}, & \forall i \in V. \end{aligned} \tag{10}$$



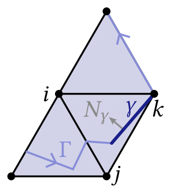
At interior vertices $i \in V$, the divergence of a per-face vector field Y_{ijk} is

$$(\nabla \cdot Y)_i := \tfrac{1}{2} \sum_{ijk>i} (\cot\theta_k^{ij} e_{\vec{ij}} + \cot\theta_j^{\vec{ki}} e_{\vec{ki}}) \cdot Y_{ijk},$$

where $e_{\vec{ij}}$ denotes the vector from vertex i to vertex j (see Polthier and Preuß [2003, Section 4]). For boundary vertices, the discrete divergence includes (due to integration by parts) an additional $n \cdot Y$ term—which we omit because it exactly cancels with our desired boundary conditions in Equation 3.

5.6 Source Discretization

We discretize the initial conditions from Equation 1 as values $(\mathsf{X}_0)_{ij} \in \mathbb{C}$ at edges ij, where the source set Ω can be a mix of oriented and unoriented curves, as well as isolated points (Figure 9). Here we treat oriented curves Γ; Appendix A treats other source types.



In principle our formulation works for arbitrary curves, though to establish explicit formulas we will assume Γ is comprised of straight segments, each of which is contained entirely inside one face or edge. Note that when the domain M is orientable, the normal N to Γ is uniquely determined by a 90° counter-clockwise rotation; on nonorientable domains, one must explicitly specify normals.

For each triangle $ijk \in F$, a segment γ contained in its interior contributes to the initial values at all three of its three edges. For instance, the contribution to $(X_0)_{ij}$ is given by the complex value

$$|\gamma| \left(N_\gamma \cdot \hat{e}_{ij} + \left(N_\gamma \cdot \hat{e}_{ij}^{\perp}\right) \imath\right) \varphi_{ij}(m_\gamma) \tag{11}$$

where $|\gamma|$, N_γ, and m_γ are the length, normal, and midpoint of the segment, *resp.* (and similarly for $(X_0)_{jk}, (X_0)_{ki}$). Empirically, however, results are more accurate if we omit the factor $\varphi_{ij}(m_\gamma)$. If γ runs along an edge ij, then it makes the same contribution, but only to $(X_0)_{ij}$. These contributions are summed over all segments to obtain final values for X_0. For intrinsic retriangulation (Figure 22), Equation 11 can be expressed using purely intrinsic data, rather

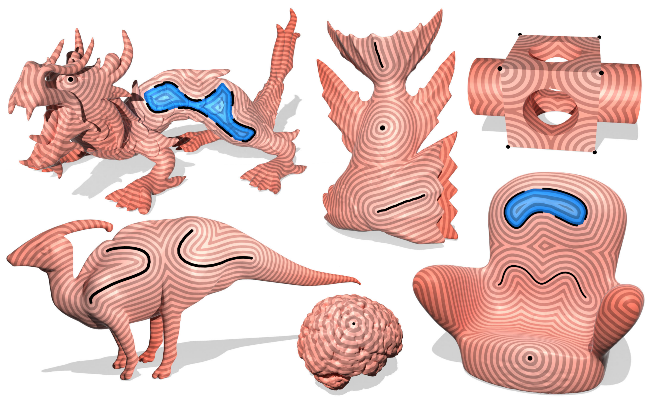


Fig. 9. We can mix and match signed and unsigned distance, selectively treating open curves as either broken region boundaries—or as literal open curves. We can also incorporate distance to isolated points.

than vectors in $\mathbb{R}^3$, by writing all vectors as differences of points in barycentric coordinates (see Sharp et al. [2021, Section 3.1]).

The formula in Equation 11 is obtained by projecting the continuous measure $N\mu_\gamma$ onto the vector CR basis. A nice feature of CR bases is that they are orthogonal. Hence, for a given segment γ we need only integrate the function ψ_{ij} with respect to the measure, or equivalently, take a Hermitian inner product along the segment:

$$\int_M \overline{\psi}_{ij} N_\gamma \, \mathrm{d}\mu_\gamma = \int_\gamma \overline{\psi}_{ij} N_\gamma \, \mathrm{d}s = N_\gamma \int_\gamma \psi_{ij} \, \mathrm{d}s = N_\gamma |\gamma| \varphi_{ij}(m_\gamma).$$

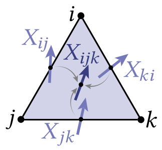
These equalities hold since N_γ is constant and $\psi_{ij} = \varphi_{ij} + 0\imath$ is linear along γ. Equation 11 expresses the final quantity in the edge basis.

6 ALGORITHM

Our discrete algorithm amounts to solving two sparse linear systems using the discrete operators defined in Section 5. First we solve the discrete vector heat equation,

$$(\mathsf{M} + t\mathsf{L}^{\nabla})\mathsf{X} = \mathsf{X}_0, \tag{12}$$

obtaining a diffused vector field X. Following Sharp et al. [2019c, Section 7.3], we let $t = h^2$, where h is the mean distance between nodes—in our case, edge midpoints, yielding half the mean edge length.



Next, we average the diffused vectors X to each face $ijk \in F$ via $X_{ijk} := (X_{ij} + X_{jk} + X_{ki})/3$ (taking care to express all vectors in the same basis, as in Section 5.4), and compute unit vectors $Y_{ijk} := \mathsf{X}_{ijk}/\|\mathsf{X}_{ijk}\|$ which represent the gradient of our (generalized) SDF. Finally, to obtain the SDF $\phi \in \mathbb{R}^{|V|}$ at vertices, we solve a sparse linear system

$$\mathsf{C}\phi = \mathsf{b} \tag{13}$$

where C is the cotan Laplacian (Section 5.5), $\mathsf{b} \in \mathbb{R}^{|V|}$ is the vector of discrete divergences given in Section 5.5.

Boundary Behavior. Unlike the unsigned heat method, our signed heat method exhibits the correct behavior at the boundary (Figure 8), without any special boundary treatment (as in Edelstein et al. [2023, Section 4.2]). The reason is that UHM obtains the vector field X as

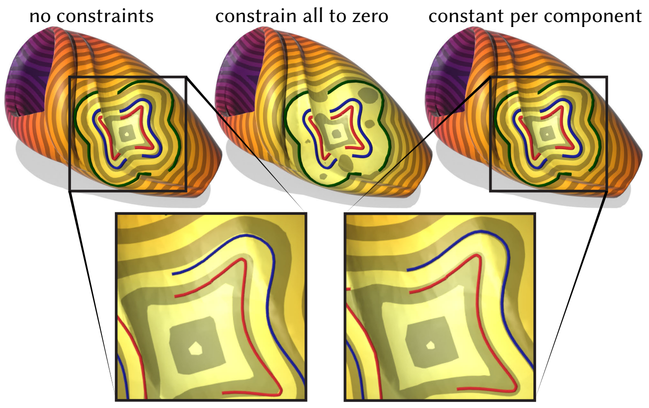


Fig. 10. We can also fit a signed distance function to several partial level sets. *Left*: Without constraints, isolines deviate slightly from source geometry. *Center*: Blindly constraining to zero along *all* curves grossly violates the distance property. *Right*: Constraining values to be constant along each component nicely matches the input geometry.

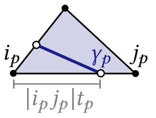
the gradient of a scalar heat distribution u with either zero-Dirichlet or zero-Neumann boundary conditions—in either case, the gradient of u cannot point in the right direction (either purely normal or purely tangential, *resp.*); Crane et al. [2013b, Section 3.4] suggests to simply take a fixed linear combination. In contrast, even at the boundary our vector diffusion step directly provides the normal at the closest point, which agrees with the gradient of the true SDF. The basic reason is that the discrete connection Laplacian (Equation 9) encodes zero-Neumann boundary conditions on the vector field itself [Gelfand et al. 2000, I.6], rather than a scalar potential u.

7 OPTIONAL EXTENSIONS

Here we discuss some optional extensions of our basic method. These extensions take advantage of the global variational nature of our method to provide capabilities not possible with iterative region-growing methods based on *MMP* [Mitchell et al. 1987] or *fast marching* [Kimmel and Sethian 1998].

7.1 Preserving Level Sets

The final Poisson solve (Equation 13) recovers the signed distance function ϕ only up to a constant shift. To make the zero level set of ϕ approximate the source geometry Ω, a common strategy is to shift ϕ by its average over Ω [Kazhdan et al. 2006; Calakli and Taubin 2011; Crane et al. 2013b]. An alternative in our setting is to add an explicit linear constraint to Equation 2 that ensures ϕ takes the same value at all points of Ω.



On a surface mesh, for instance, suppose Ω has at most one segment per triangle, with endpoints $\gamma_0, \ldots, \gamma_m$ on edges. Then we can impose linear constraints of the form

$$(1-t_p)\phi_{i_p} + t_p\phi_{j_p} = (1-t_0)\phi_{i_0} + t_0\phi_{j_0}, \quad p = 1, \ldots, m,$$

where $t_p \in [0,1]$ encodes the location of x_p along edge i_p, j_p (see inset). We encode these constraints by a matrix $\mathsf{A} \in \mathbb{R}^{m\times|V|}$. Minimizing $\|\nabla\phi - Y_t\|_2^2$ subject to $\mathsf{A}\phi = 0$ then corresponds to solving a

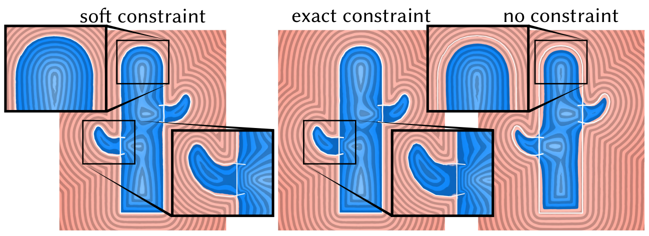


Fig. 11. Soft level set constraints can easily be incorporated into our PDE-based framework. Here soft constraints encourage the zero set to lie near the input curves, without distorting the SDF near self-intersections as much as a hard constraint.

saddle-point problem

$$\begin{bmatrix} \mathsf{C} & \mathsf{A}^{\mathsf{T}} \\ \mathsf{A} & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \mu \end{bmatrix} = \begin{bmatrix} \nabla \cdot \mathsf{Y}_t \\ 0 \end{bmatrix}, \tag{14}$$

where $\mu \in \mathbb{R}^m$ are Lagrange multipliers. Since $\phi|_\Omega$ is now constant by construction, we can shift it to be *exactly* zero on all of Ω—so long as the constraints are compatible. We apply this procedure in all figures unless otherwise noted. More generally, suppose the input Ω represents multiple, distinct level sets $\phi^{-1}(c_1), \phi^{-1}(c_2), \ldots$ for values c_i which are not known *a priori*. Here we can apply an identical set of constraints per connected component, ensuring that the value along each component is constant (Figure 10, *right*).

Finally, similar to Kazhdan and Hoppe [2013, Section 4], we can replace the hard constraint with a soft penalty term $\int_\Omega \|\phi(x) - c(x)\|^2 \, dx$, where $\lambda > 0$ is the penalty strength (Figure 11). The minimizer is obtained by solving the system $(\mathsf{C} - \lambda\mathsf{A}^\top\mathsf{A})\phi = \mathsf{b}$.

7.2 Nonbounding Loops and Discontinuous Distance

The gradient of unsigned distance can be discontinuous on a lower-dimensional set (analogous to the *cut locus* of a point), but is still tangent continuous relative to this set, meaning vectors on either side project to the same direction (Figure 13, *top*). In contrast, when Ω does not bound a region of M, the vector field Y_t from Step II of our method can fail to be tangent continuous—and hence fail to be integrable via a globally continuous function (Figure 13, *bottom*).

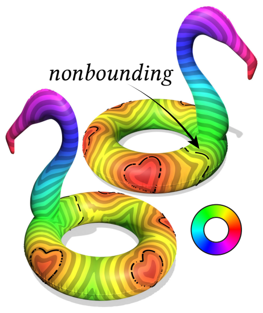


Fig. 12. GSD extends to curves that do not even bound a region.

One possibility is to simply filter out nonbounding components [Feng et al. 2023]; we instead consider a generalized notion of signed distance that remains meaningful. In particular, we now seek a piecewise continuous solution by replacing the L^2 problem in Equation 2 with an L^1 problem that ignores neighborhoods where Y_t is highly nonintegrable. On triangle meshes, we quantify nonintegrability via an edge-based curl inspired by Polthier and Preuß [2003, Section 4]:

$$(\nabla \times \mathsf{Y})_{ij} := (\hat{e}_{ij} + \hat{e}_{ji}) \cdot \mathsf{Y}_{ijk}. \tag{15}$$

This quantity directly captures tangent discontinuity along edges: taking edge orientation into account, it is simply the difference

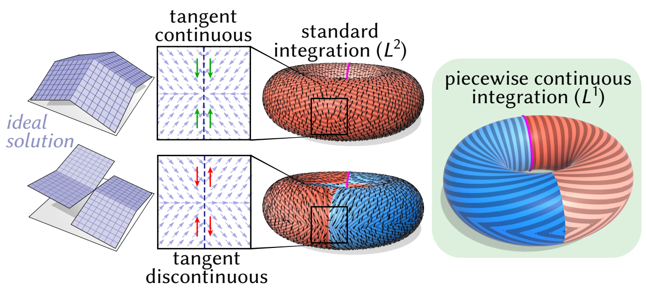

tangent continuous
standard integration (L^2)
piecewise continuous integration (L^1)
ideal solution
tangent discontinuous

Fig. 13. Unlike the gradient of an unsigned distance function *(top)*, the unit vector field we compute for signed distance might not be integrated by any continuous function *(bottom)*. In such scenarios, we can instead compute a piecewise continuous SDF *(right)*.

of projected vectors. Discontinuous functions ϕ are represented as piecewise linear functions interpolating values at corners. To integrate Y in a piecewise continuous sense, we then minimize a weighted L^1 norm that penalizes discontinuous jumps across edges:

$$\begin{aligned} \min_{\phi \in \mathbb{R}^{|C|}} & \sum_{ij \in E} |ij| e^{-|\nabla \times \mathrm{Y}|_{ij}} |\phi_i^{jk} - \phi_i^{lj}| \\ \text{s.t. } & \phi_j^{ki} - \phi_i^{jk} = \mathrm{Y}_{ij} \cdot e_{\overrightarrow{ij}}, \quad \forall_i^{jk} \in C, \end{aligned} \tag{16}$$

Exponential weights incur a lower penalty when the field is not integrable; simultaneously, the L^1 norm tries to minimize the length of this discontinuity as much as possible. The constraints ensure Y is integrated exactly within each triangle. In practice, we apply the same transformations as Feng et al. [2023, Section 3.4] to obtain a linear program with only $|F|$ degrees of freedom. Examples are shown in Figure 13 *(right)* and Figures 14 and 12. In Figure 12 we also use a small amount of heat flow to smooth out minor discontinuities from L_1 optimization.

7.3 Distance Sharpening

Unsigned geodesic distance can also be expressed as the solution to a convex optimization problem, akin to the convex formulation of graph distance [Dantzig 1963, Ch. 17], [Erickson 2019, Ch. H]:

$$\begin{aligned} \max_{\phi} & \int_M \phi(x) \, dx \\ \text{s.t. } & |\nabla \phi| \leq 1 \\ & \phi = 0 \text{ on } \Omega. \end{aligned} \tag{17}$$

Belyaev and Fayolle [2020] solve Equation 17 via ADMM to compute the distance to point sources. This formulation tends to be more accurate than our method—but is an order of magnitude slower (Section 9.5), and more importantly, can compute only unsigned distance. If desired, however, one can "sharpen" our results using a generalization of Equation 17. We simply replace the objective in Equation 17 with

$$\max_{\phi} \int_M \operatorname{sign}(\phi_0(\mathrm{x}))\phi(\mathrm{x}) \, d\mathrm{x} \tag{18}$$

where ϕ_0 is the distance computed by the signed heat method, and use ϕ_0 as an initial guess for ϕ. An example is shown in Figure 15.

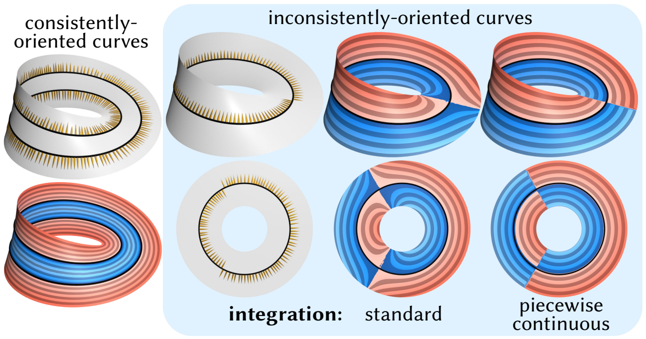


Fig. 14. Our algorithm works out-of-the-box for orientable input curves, even on nonorientable domains *(top)*. By adopting piecewise continuous integration (Section 7.2) we can also handle inconsistently-oriented curves, whether or not the domain itself is orientable *(bottom)*.

Note that robust sign information is available only thanks to our method—Equation 18 cannot be applied directly to broken curves.

In practice ADMM is quite slow, requiring repeated linear solves. Since we have a good initializer, it might be better to use a first-order method like the *primal-dual hybrid gradient method*, which is often faster than ADMM [Chambolle and Pock 2011, Figure 5] and easily implemented in parallel. We leave such exploration to future work.

8 OTHER SPATIAL DISCRETIZATIONS

8.1 Tetrahedral Meshes

The discretization from Section 5 generalizes directly to tet meshes—extending our method from curve processing to broader surface processing tasks. We again use a Crouzeix-Raviart finite element discretization, detailed in Appendix B. Note that we do not need a separate discretization for the connection Laplacian, since on a flat domain we can just apply the scalar Laplacian componentwise. For simplicity, we generate a mesh that conforms to input triangles (via *TetGen* [Hang 2015]), so that the source term is just $X^0(x) := \sum_{ijk \in \Omega} |ijk| N_{ijk} \psi_{ijk}(x)$. Examples are shown in Figures 19, 26.

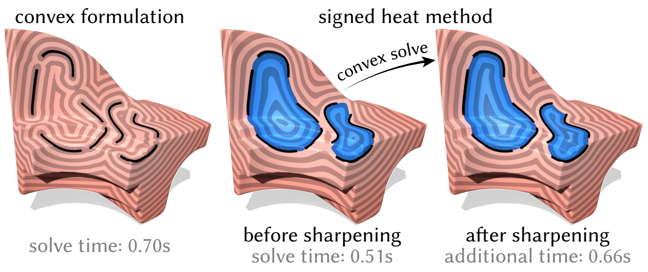


Fig. 15. *Left:* Methods based on convex optimization yield more accurate distances, but compute only unsigned distance. *Right:* Using our method as a warm start, we can "sharpen" distance while preserving the inside-outside classification. Here we start with a large diffusion time ($t = 100h^2$) to visually emphasize the effect.

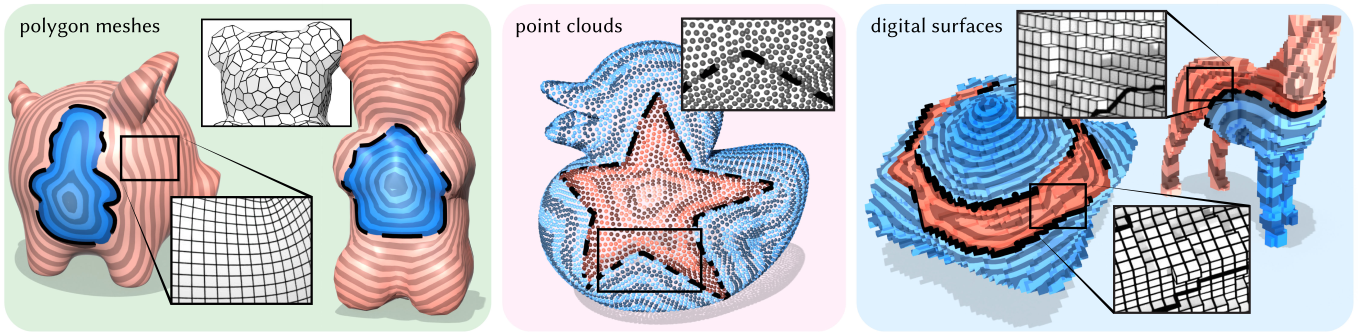


Fig. 16. Our method extends to polygon meshes, point clouds, and digital surfaces. Digital surface meshes are from Coeurjolly and Levallois [2015].

8.2 Regular Grids

To avoid the cost of tet mesh generation, we can also use a regular grid on $\mathbb{R}^3$. Here, rather than solve a linear system in Step I, we can directly approximate X_t via the fundamental solution of the operator id $- t\Delta$, given in $\mathbb{R}^3$ by the *Yukawa potential*

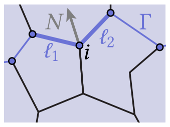
$$G_t(x,y) = -\tfrac{1}{4\pi\|x-y\|}e^{-\sqrt{1/t}\|x-y\|}.$$

(In practice, we find this approach works better than convolving with the heat kernel directly.) The diffused vector field at each grid node $x \in \mathbb{R}^3$ is then approximated by midpoint quadrature at triangle barycenters b_{ijk}:

$$X_t(x) = \sum_{ijk\in F}\int_{ijk} \mathsf{N}_{ijk}G_t(x,y)\,dy \approx \sum_{ijk\in F}|ijk|\mathsf{N}_{ijk}G_t(x,b_{ijk}),$$

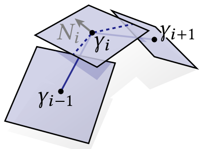
where N_{ijk} are triangle normals; this step could be accelerated from $O(|F|)$ to $O(\log|F|)$ via the method of Barnes and Hut [1986]. Step III uses a standard seven-point Laplacian, and evaluates $\nabla\cdot Y_t$ using forward differences. Figure 4 shows that we obtain high-quality distances and offset surfaces even for badly corrupted geometry (with contouring via *marching cubes* [Lorensen and Cline 1998]).

8.3 Polygon Meshes



On surface meshes made of general polygons (which may be nonconvex or even nonplanar), we adopt the Laplacian, divergence, and mass matrices defined by Bunge et al. [2020] in terms of a carefully-chosen triangular refinement. Since Bunge *et al.* do not define a connection Laplacian, we define tangent planes at vertices using the area-weighted sum of fine triangle normals (picking an arbitrary basis in each plane). As in Equation 9, a connection Laplacian is then obtained by multiplying off-diagonal entries with complex numbers that encode the smallest rotation between tangent bases at adjacent vertices. For simplicity we assume the curve Γ runs along edges; the initial conditions X^0 at each vertex i of Γ are then given by the average edge normal N times half the sum of incident edge lengths ℓ_1, ℓ_2 (inset). To compute divergence, we prolong the diffused vector field X to the fine vertices, average them onto faces, then apply the operator **D** from Bunge et al. [2020, Section 4.5]. Examples are shown in Figure 16, *left*.

8.4 Point Clouds



For point clouds, we use the Laplacian of Sharp and Crane [2020a, Section 5.7], which also defines per-point tangent planes, and the mass matrix from their *tufted cover* [Sharp and Crane 2020a, Section 3.3]. We again augment this Laplacian with off-diagonal rotations to get a connection Laplacian, as described by Sharp et al. [2019c, Section 6.2.2] and implemented in *geometry-central* [Sharp et al. 2019a]. Each component of Γ is a sequence of points γ_i from the cloud, where normal N_i can be computed by projecting $\gamma_{i-1}, \gamma_i, \gamma_{i+1}$ onto the tangent plane at point i and taking the in-plane normal (inset). As in Section 8.3, we compute X^0 by accumulating length-weighted normals at each point. To compute divergence, we average diffused vectors onto faces of the tufted cover and apply the formula from Section 5.5. An example is shown in Figure 16, *center*.

8.5 Voxelizations

Our method also extends to *digital surfaces*, which are the boundaries of voxelized volumes. We use the vertex-based Laplacian, divergence, and connection Laplacian defined by Coeurjolly and Lachaud [2022], using the implementation in *DGtal* [Coeurjolly et al. 2010]. Analogous to Section 8.3, we encode the source term via extrinsic projection of curve normals onto tangent bases defined at vertices, where vertex normals are normalized averages of per-face *corrected normals* [Coeurjolly et al. 2014]. We assume that Γ is piecewise linear on faces of the digital surface, and obtain initial conditions X^0 by projecting segment normals onto tangent bases at vertices, and multiplying by half the edge length of incident segments. Examples are shown in Figure 16, *right*.

9 EVALUATION

We next evaluate our method via a broad range of numerical experiments. As noted in Section 1, our primary focus is not on speed or accuracy, but rather on improving the robustness and generality of signed distance computation. Nonetheless, our *signed heat method (SHM)* is quite competitive in speed and accuracy, since it boils down to the same kinds of sparse linear systems as previous methods. In particular, the method takes only a few hundred milliseconds for meshes with hundreds of thousands of triangles, and exhibits the

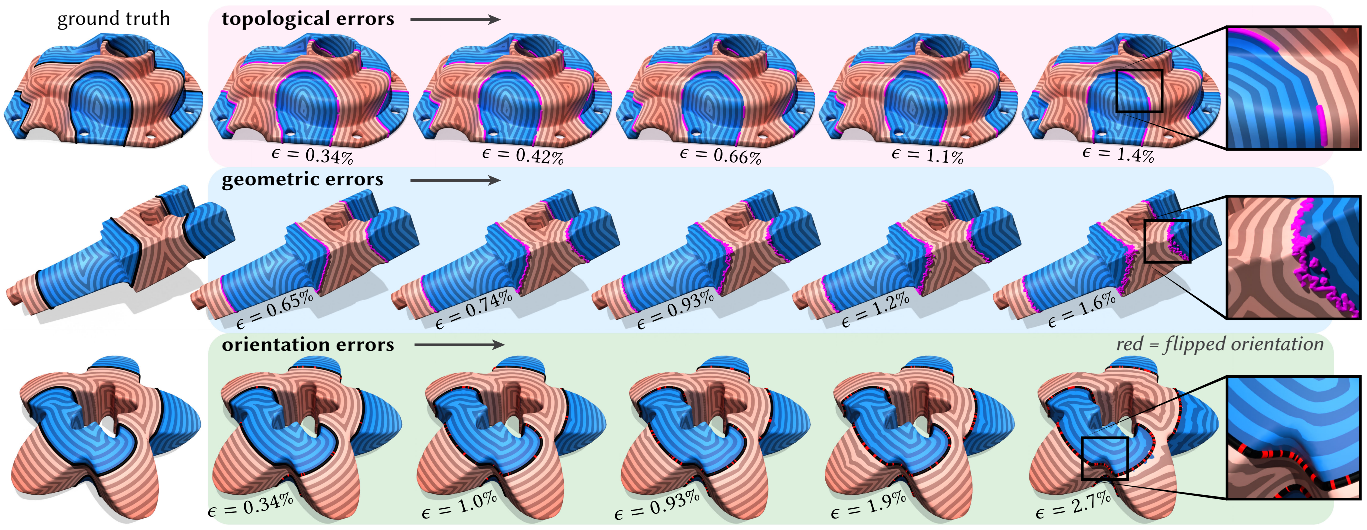


Fig. 17. Our method provides robust and reliable signed distance approximation, failing gracefully in the presence of significant topological, geometric, or orientation errors. Errors ϵ in geodesic distance are displayed relative to the exact polyhedral SDF of a finely sampled version of the original curve.

same linear convergence with respect to mesh refinement as earlier schemes. More importantly, for broken geometry it still provides an accurate approximation of the distance to the original, uncorrupted geometry—whereas past methods can fail to provide a reasonable distance approximation (Figure 7, *right*).

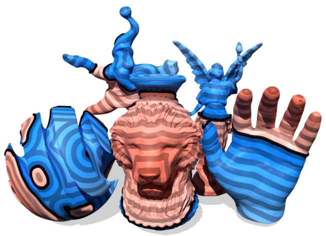
9.1 Implementation

All methods were implemented in C++, in double precision, using *geometry-central* [Sharp et al. 2019a] for mesh data structures and intrinsic retriangulation, and *Eigen* [Guennebaud et al. 2010] for sparse linear algebra. Timings were taken on an Apple M1 with 8GB RAM. The error ϵ in any distance approximation ϕ is quantified via L_2 error normalized by the range of the true distance ϕ^0, *i.e.*,

$$\epsilon(\phi) := \frac{1}{(\max(\phi^0)-\min(\phi^0))\,\mathcal{A}^{1/2}}\left(\sum_{i\in V} A_i\left(\phi_i-\phi_i^0\right)^2\right)^{1/2},$$

where $A_i = \frac{1}{3}\sum_{ijk>i}|ijk|$ is the area associated with vertex i, and $\mathcal{A} := \sum_{i\in V} A_i$ is the total surface area.

9.2 Dataset



For evaluation, we consider a dataset of closed, region-bounding loops (since past methods do not handle open or nonbounding curves). This dataset is derived from all 44 genus-zero meshes without boundary from Myles et al. [2014]. In order to perform a study of convergence under refinement (Figure 25) we first remesh each model to approximately 1.25k vertices using quadric error simplification [Garland and Heckbert 1997], then compute four levels of loop subdivision [Loop 1987]. At each level, we then extract level sets of the same low-frequency Laplacian eigenfunctions [Lévy and Zhang 2010] using marching triangles, and remesh to a constrained intrinsic Delaunay triangulation containing these curves [Sharp and Crane 2020b, Section 6.3].

9.3 Examples

9.3.1 Morphological Operations. One natural use case for generalized signed distance is to contour broken geometry, and generate accurate fixed-distance offsets (Figures 26, 19, 4). Generating such offsets from imperfect geometry is useful, for example, for 3D printing, or for downstream mesh processing tasks that require closed or manifold surfaces. One can also “inflate” or “shrink” shapes by taking positive or negative offsets—and combining these two operations in sequence can be used to simplify high-frequency features of broken shapes as if they were whole (Figure 20). Note that in Figure 26 we sample both functions onto the same tet mesh, using `libigl` to evaluate GWN [Jacobson et al. 2018].

9.3.2 Illustration on Surfaces. Geodesic distance is also used to design curves on surfaces—where prior work largely considers perfect closed curves [Nazzaro et al. 2022]. Our algorithm can be used to pre-process curves into the requisite closed format, *e.g.*, those hastily sketched on a surface (Figure 21). To control the behavior of this completion operation, one can use diffusion time to control completion behavior, providing an analog of line join options from 2D vector graphics (Figure 18).

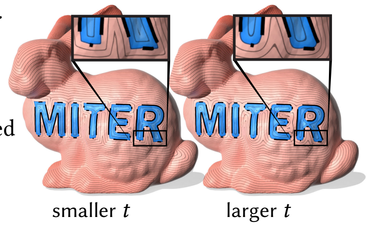


Fig. 18. Adjusting diffusion time fills in broken letters with either round or sharp corners, yielding effects similar to different line joins for 2D strokes.

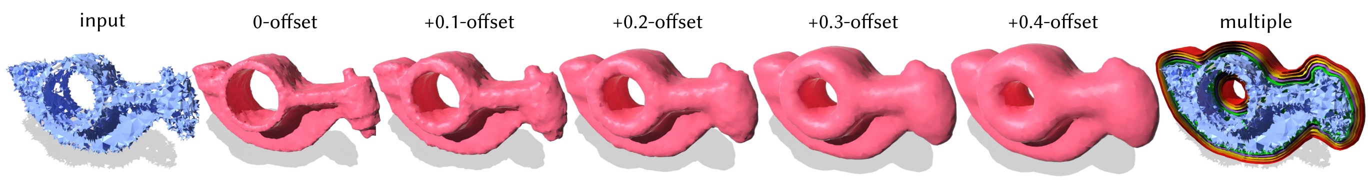


Fig. 19. By extracting level sets of generalized signed distance, we can convert broken, noisy, and nonmanifold input geometry *(far left)* into closed, regular, manifold surfaces and evenly-spaced offset surfaces.

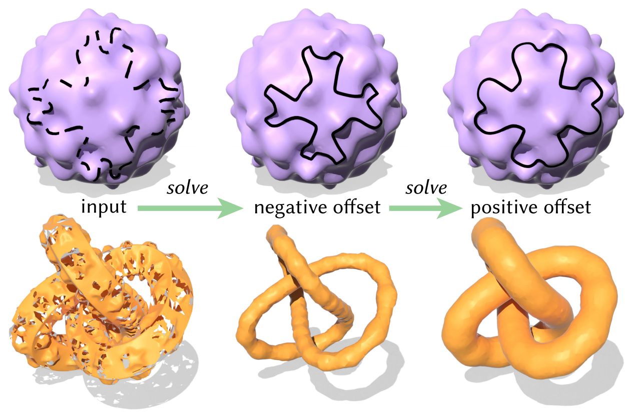
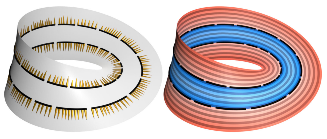


Fig. 20. One can simplify high-frequency features of broken curves and surfaces by taking consecutive positive/negative offsets of a generalized signed distance function, akin to dilation/erosion.

9.4 Robustness



Since our method is built on well-behaved elliptic PDEs, it performs well on not only broken input with corrupted topology, geometry, and orientation (Figure 17) but also challenging surface domains (Figure 23). Since our method is purely intrinsic, it applies to meshes that are only immersed (rather than embedded); it also applies out of the box to nonmanifold and nonorientable meshes (inset), since all our differential operators are local and defined per-face, and hence oblivious to any nonmanifold features. Our method is robust across varying degrees of nonmanifoldness and missing data (Figure 23, *top and bottom left*). As with all methods that rely on discretizing PDEs, the quality of the solution can degrade with poor tessellations of the geometry, though we can easily apply *intrinsic Delaunay triangulation* to get good-quality solutions (Figure 22). For surfaces in $\mathbb{R}^3$ our method remains robust, even for extremely corrupted input surfaces (Figures 4,19).

As seen in Figure 27 we also obtain more natural surface completion than GWN, which for general surfaces is hard to contour with any single level set value. Similar to spline interpolation, the zero set of GSD nicely matches both positions and normals along hole boundaries. Here meshes for both methods are extracted using the marching tetrahedra implementation in *libigl* [Jacobson and Panozzo 2017].

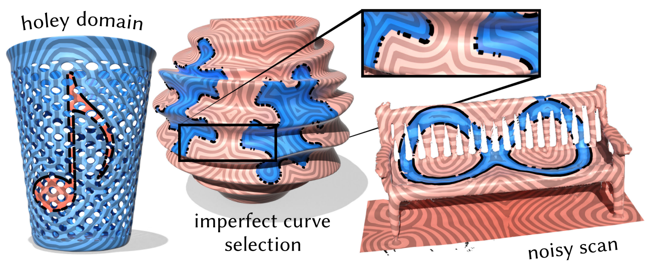


Fig. 21. Broken curves easily arise from attempting to draw curves on surfaces of high genus, with overhangs, and with holes and scanner noise. Our method yields signed distance functions robust to these challenges. (Scanned bench from from Choi et al. [2016].)

9.5 Accuracy and Performance

We compare against the unsigned heat method (UHM) of Crane et al. [2013b], which provides a useful reference point since nearly all work on geodesic distance algorithms from the past decade compares against this method. As a more recent reference point, we also compare with the convex formulation of Belyaev and Fayolle [2020] (labeled BF). Since neither method directly handles curve sources, we either integrate the initial scalar heat distribution against hat functions (for UHM) or simply use the set of curve vertices as the source set (for BF). (Methods that directly handle curve sources do not have an open source implementation [Bommes and Kobbelt 2007], or do not include curve sources in their public release [Trettner et al. 2021].) Finally, since BF must constrain the zero set, we impose the same constraints on UHM/SHM (Section 7.1), and do not pre-factor any matrices. Note, however, that for multiple source

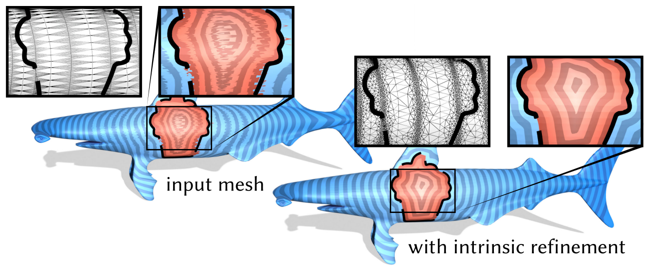


Fig. 22. The quality of our method depends on the underlying mesh quality; but since our formulation is purely intrinsic, we can trivially improve accuracy and robustness by invoking *intrinsic Delaunay refinement* [Gillespie et al. 2021], without changing anything else about our implementation.

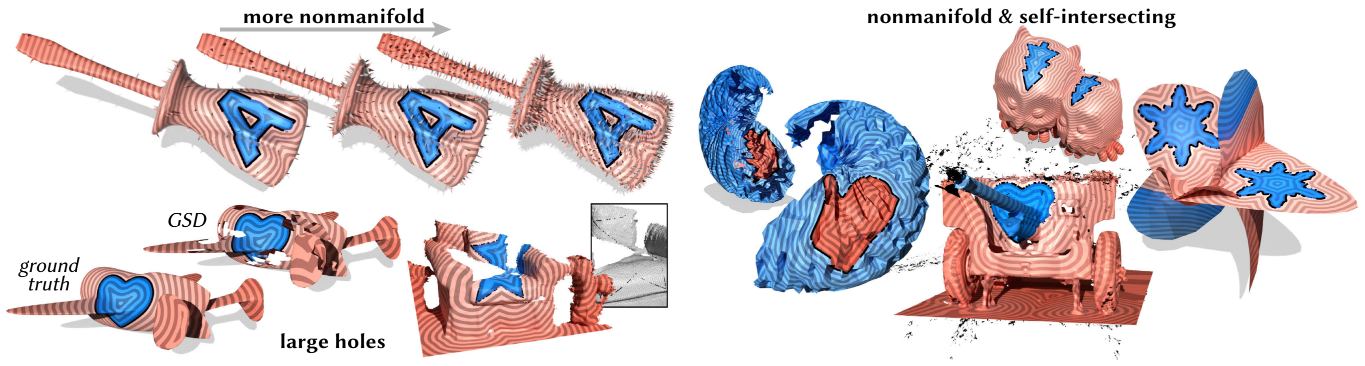


Fig. 23. Our method robust not only to errors in the source geometry, but also in the domain mesh itself. Here we obtain well-behaved SDFs even for meshes found "in the wild," such as amateur-created 3D scans [Choi et al. 2016]. Even in cases where a notion of inside and outside is meaningless (such as the rightmost mesh), our method fails gracefully—still producing a good signed distance approximation near the input curve.

terms, heat methods can achieve about two orders of magnitude speedup by omitting factorization [Crane et al. 2013b, Table 1].

9.5.1 Planar Domains. As noted by Crane et al. [2013b, Figure 21], even exact polyhedral distance (including MMP) provides only a 2nd-order accurate estimate of true (smooth) geodesic distance, due to errors in the approximation of the domain itself. To avoid conflating these two sources of error, we first consider closed, planar curves—where the exact SDF is easily computed via closest-point queries [Sawhney et al. 2020], and sign can unambiguously be determined via standard inside/outside tests [Haines 1994]. As seen in Figure 24, our method is slightly slower but slightly more accurate than UHM. Without sharpening, it is not as accurate as BF—but is an order of magnitude faster. Moreover, BF must trade off between bias near the boundary [Edelstein et al. 2023, Figure 3, *left*], or distortion in the presence of curve sources, depending on whether Hessian regularization is omitted or included (*resp.*).

9.5.2 Surface Domains. We next consider closed curves on surface meshes. Here we can no longer obtain the true distance on an unknown underlying smooth surface; we hence compute "ground truth" distance as the exact polyhedral distance to a finely-sampled version of the input curve (100 samples per curve edge) using MMP [Mitchell et al. 1987], which itself has $O(h^2)$ error. We plot convergence and solve times in Figure 25. We observe that our method has approximately linear convergence in mean edge length, with better consistency on curve sources compared to other methods. We find the same trend in solve times as in Section 9.5.

9.5.3 Broken Curves. Finally, we compare against the end-to-end pipeline of repairing broken geometry, computing unsigned distance to the fixed geometry, and signing the unsigned distance. In particular, we use *surface winding numbers (SWN)* to contour broken curves [Feng et al. 2023], and compute exact polyhedral distance using MMP [Mitchell et al. 1987] using the curve vertices as the source set. As input surface domains, we use the meshes with ~5k vertices from the same dataset as Section 9.5.2. As input curves, we take level sets of five different low-frequency Laplacian eigenfunctions, and add geometric and topological errors by taking the union of the curves with their offsets (found by taking boundaries of triangle strips), and deleting about 50% of the curve at random intervals.

The repair-distance-sign pipeline is particularly sensitive to errors in the input, since any errors made during contouring are permanent and will destroy the quality of the final SDF no matter how accurate the subsequent distance computation. In particular, contouring the winding number is notoriously difficult, and often

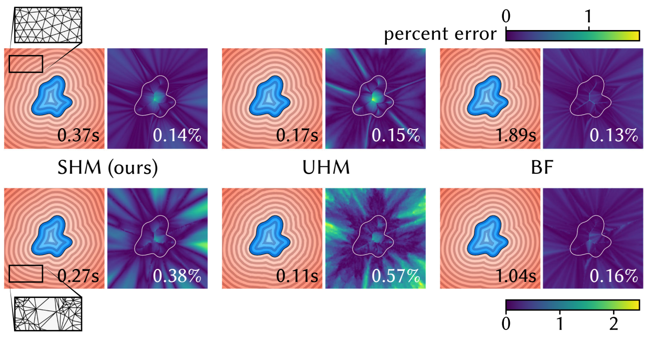


Fig. 24. Distribution of error in distance approximation for a perfect, unbroken curve on a high- and low-quality mesh *(top/bottom)*. Both meshes have about 100k faces. Inset numbers on SDF and error plots indicate compute time and mean error (*resp.*). Overall our method is quite comparable to the original heat method, and less accurate than BF on the low-quality mesh—but about 4–5x faster.

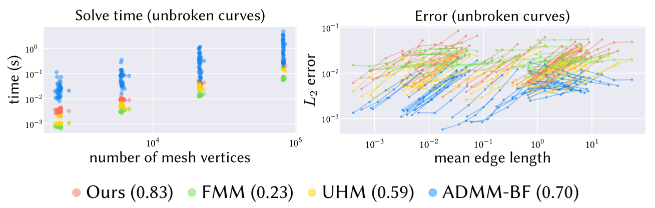


Fig. 25. We observe approximately linear convergence in distance accuracy on a benchmark of unbroken (closed) curves on 44 different meshes. The legend shows median orders of accuracy. Note that if we omit the zero set constraint, enabling us to re-use both factorizations, our method and UHM become 1–2 orders of magnitude faster.

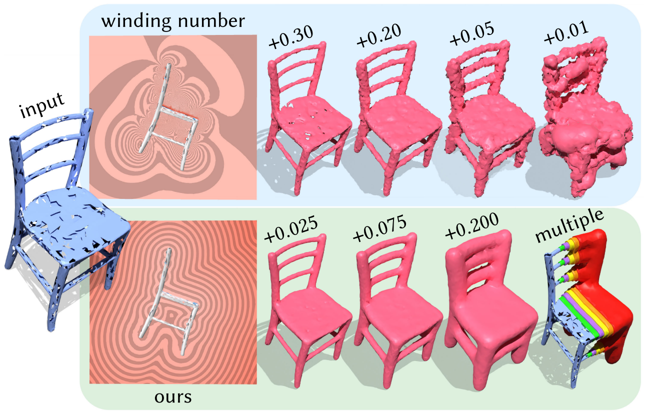


Fig. 26. *Top:* Generalized winding number (GWN) cannot be used for offset surfaces, since it provides only a smooth indicator function—and not a signed distance. *Bottom:* Our generalized signed distance (GSD) provides much nicer offsets on the same broken mesh.

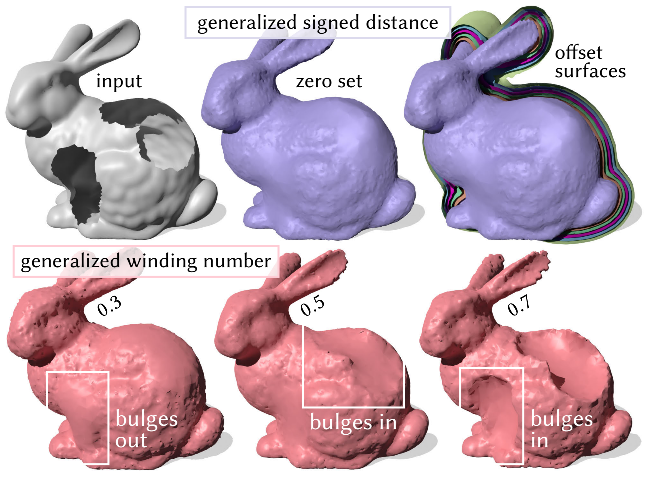


Fig. 27. GWN completes surfaces with saddle-shaped harmonic patches that exhibit poor normal continuity with the observed geometry (across many contour values). Our method directly incorporates normal information, providing more plausible reconstruction even for large holes.

leads to misclassified regions (Figure 28). Though Feng et al. [2023] suggest a rounding procedure, it remains unclear *which* half-integer level set to take as the boundary between inside and outside; we use the values of the rounded winding number function that appear most often along the input curves, though this results in 12% of examples with >50% of surface area misclassified. We also try contouring winding number according to the average of the winding number function along the input curves, but we find similar results. In contrast, our method achieves greater robustness by averaging normal vectors over the whole domain. The repair-distance-sign pipeline is also about 10x more expensive due to its intermediate steps; SWN alone has cost asymptotically equivalent to our method.

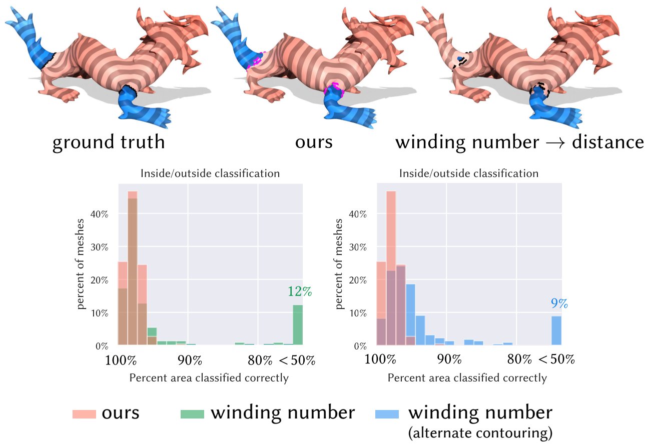
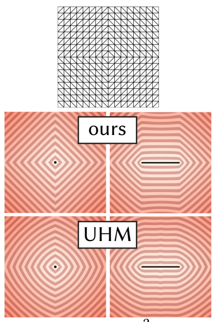


Fig. 28. We compare, on 220 examples, the accuracy of GSD versus a hybrid scheme that repairs broken geometry using winding numbers, computes unsigned distance to the repaired geometry, then signs the unsigned distance. Winding numbers are worse at classifying inside/outside, regardless of contouring method. As a result, the hybrid scheme yields about 3x lower distance accuracy on average (0.11 vs. 0.04 L_2 error, *resp.*), even though it benefits from exact distance (via MMP).

10 LIMITATIONS AND FUTURE WORK

As with past heat methods, diffusion time cannot be made arbitrarily small: similar to Crane et al. [2013b, Appendix A], we observe empirically that GSD behaves like graph distance as $t \longrightarrow 0$ (see inset). As with past heat methods, however, SHM consistently provides accurate distance approximation for $t = h^2$. Our strategy for topologically nonbounding and nonorientable curves (Section 7.2) incurs a more expensive L^1 problem—but is already a substantial generalization over anything handled by previous geodesic or SDF algorithms. A more careful study of this situation (including mitigation of small per-edge discontinuities) is an interesting direction for future work. Likewise, since broken input sometimes comes with quantifiable measurement error, it might be interesting to explore a notion of "confidence" in the computed distance values, along the lines of work by Sellán and Jacobson [2022].



$t = 0.01h^2$

Finally, like other robust inside-outside and reconstruction algorithms [Bærentzen and Aanæs 2005; Kazhdan et al. 2006; Jacobson et al. 2013; Feng et al. 2023], we assume input has mostly consistent orientation. Though we are fairly robust to orientation errors (Section 9.4), it may be interesting to explore diffusion of more general *line fields* to factor out normal orientation, *à la* Alliez et al. [2007].

ACKNOWLEDGMENTS

This work was funded by an NSF CAREER Award (IIS 1943123), NSF Award IIS 221229, a Packard Fellowship, and gifts from Adobe Research. The authors also thank Alec Jacobson for helpful conversations about winding numbers and the pseudonormal.

REFERENCES

Marc Alexa, Philipp Herholz, Max Kohlbrenner, and Olga Sorkine-Hornung. 2020. Properties of Laplace Operators for Tetrahedral Meshes. *Computer Graphics Forum* (2020). https://doi.org/10.1111/cgf.14068

P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun. 2007. Voronoi-Based Variational Reconstruction of Unoriented Point Sets. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Barcelona, Spain) *(SGP '07)*. Eurographics Association, Goslar, DEU, 39–48.

Matan Atzmon and Yaron Lipman. 2019. SAL: Sign Agnostic Learning of Shapes from Raw Data. *CoRR* abs/1911.10414 (2019). arXiv:1911.10414 http://arxiv.org/abs/1911.10414

J.A. Bærentzen and H. Aanæs. 2005. Signed distance computation using the angle weighted pseudonormal. *IEEE Transactions on Visualization and Computer Graphics* 11, 3 (2005), 243–253. https://doi.org/10.1109/TVCG.2005.49

J Andreas Bærentzen. 2005. Robust generation of signed distance fields from triangle meshes. In *Fourth International Workshop on Volume Graphics, 2005.* IEEE, 167–239.

Josh Barnes and Piet Hut. 1986. A hierarchical O (N log N) force-calculation algorithm. *nature* 324, 6096 (1986), 446–449.

Alexander Belyaev and Pierre-Alain Fayolle. 2020. An ADMM-based scheme for distance function approximation. *Numerical Algorithms* 84 (2020), 14. https://doi.org/10.1007/s11075-019-00789-5

Alexander Belyaev, Pierre-Alain Fayolle, and Alexander Pasko. 2013. Signed Lp-distance fields. *Computer-Aided Design* 45, 2 (2013), 523–528.

Nicole Berline, Ezra Getzler, and Michèle Vergne. 1992. *Heat Kernels and Dirac Operators* (1 ed.). Springer Berlin, Heidelberg.

David Bommes and Leif Kobbelt. 2007. Accurate Computation of Geodesic Distance Fields for Polygonal Curves on Triangle Meshes. *VMV*, 151–160.

Alan Brunton and Lubna Abu Rmaileh. 2021. Displaced Signed Distance Fields for Additive Manufacturing. *ACM Trans. Graph.* 40, 4, Article 179 (July 2021), 13 pages. https://doi.org/10.1145/3450626.3459827

Astrid Bunge, Philipp Herholz, Misha Kazhdan, and Mario Botsch. 2020. Polygon Laplacian Made Simple. *Computer Graphics Forum* 39, 2 (2020), 303–313. https://doi.org/10.1111/cgf.13931 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13931

Fatih Calakli and Gabriel Taubin. 2011. SSD: Smooth signed distance surface reconstruction. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1993–2002.

Antonin Chambolle and Thomas Pock. 2011. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *Journal of Mathematical Imaging and Vision* 40 (2011). https://doi.org/10.1007/s10851-010-0251-1

Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. 2016. A Large Dataset of Object Scans. *arXiv:1602.02481* (2016).

David Coeurjolly and Jacques-Olivier Lachaud. 2022. A Simple Discrete Calculus for Digital Surfaces. In *IAPR Second International Conference on Discrete Geometry and Mathematical Morphology*, Étienne Baudrier, Benoît Naegel, Adrien Krähenbühl, and Mohamed Tajine (Eds.). Springer, LNCS.

David Coeurjolly, Jacques-Olivier Lachaud, et al. 2010. DGtal: Digital Geometry tools and algorithms library. http://dgtal.org.

David Coeurjolly, Jacques-Olivier Lachaud, and Jérémy Levallois. 2014. Multigrid convergent principal curvature estimators in digital geometry. *Computer Vision and Image Understanding* 129 (2014), 27–41.

David Coeurjolly and Jérémy Levallois. 2015. VolGallery. https://github.com/dcoeurjo/VolGallery.

Keenan Crane, Fernando De Goes, Mathieu Desbrun, and Peter Schröder. 2013a. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses*. 1–126.

Keenan Crane, Marco Livesu, Enrico Puppo, and Yipeng Qin. 2020. A Survey of Algorithms for Geodesic Paths and Distances. *arXiv e-prints*, Article arXiv:2007.10430 (July 2020), arXiv:2007.10430 pages. arXiv:cs.GR/2007.10430

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013b. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)* 32, 5 (2013), 1–11.

George Dantzig. 1963. *Linear programming and extensions.* Princeton university press.

Alexandre Djerbetian and Mirela Ben Chen. 2016. *Tangent Vector Fields on Triangulated Surfaces-An Edge-Based Approach.* Ph.D. Dissertation. Computer Science Department, Technion.

Michal Edelstein, Nestor Guillen, Justin Solomon, and Mirela Ben-Chen. 2023. A Convex Optimization Framework for Regularized Geodesic Distances. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 2, 11 pages. https://doi.org/10.1145/3588432.3591523

Jeff Erickson. 2019. *Algorithms.* Jeff Erickson. http://jeffe.cs.illinois.edu/teaching/algorithms/

Alexandre Ern and Jean-Luc Guermond. 2004. *Theory and Practice of Finite Elements* (1 ed.). Applied Mathematical Sciences, Vol. 159. Springer, New York, NY. https://doi.org/10.1007/978-1-4757-4355-5

Nicole Feng, Mark Gillespie, and Keenan Crane. 2023. Winding Numbers on Discrete Surfaces. *ACM Trans. Graph.* 42, 4, Article 36 (jul 2023), 17 pages. https://doi.org/10.1145/3592401

Jean Gallier, Jocelyn Quaintance, Jean Gallier, and Jocelyn Quaintance. 2020. Operators on Riemannian Manifolds: Hodge Laplacian, Laplace-Beltrami Laplacian, the Bochner Laplacian, and Weitzenböck Formulae. *Differential Geometry and Lie Groups: A Second Course* (2020), 361–401.

Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 209–216. https://doi.org/10.1145/258734.258849

Izrail Moiseevitch Gelfand, Richard A Silverman, et al. 2000. *Calculus of variations.* Courier Corporation.

Mark Gillespie, Nicholas Sharp, and Keenan Crane. 2021. Integer Coordinates for Intrinsic Geometry Processing. *ACM Trans. Graph.* 40, 6 (2021).

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. *CoRR* abs/2002.10099 (2020). arXiv:2002.10099 https://arxiv.org/abs/2002.10099

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Eric Haines. 1994. Point in Polygon Strategies. *Graphics Gems* 4, 1 (1994), 24–46.

Si Hang. 2015. TetGen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw* 41, 2 (2015), 11.

John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545.

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral meshing in the wild. *ACM Trans. Graph.* 37, 4 (2018), 60–1.

Saar Huberman, Amit Bracha, and Ron Kimmel. 2023. Deep Accurate Solver for the Geodesic Problem. In *International Conference on Scale Space and Variational Methods in Computer Vision.* Springer, 288–300.

Alec Jacobson, Ladislav Kavan, and Olga Sorkine. 2013. Robust Inside-Outside Segmentation using Generalized Winding Numbers. *ACM Trans. Graph.* 32, 4 (2013).

Alec Jacobson and Daniele Panozzo. 2017. Libigl: Prototyping geometry processing research in c++. In *SIGGRAPH Asia 2017 courses.* 1–172.

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

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson Surface Reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy) *(SGP '06)*. Eurographics Association, Goslar, DEU, 61–70.

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.* 32, 3, Article 29 (jul 2013), 13 pages. https://doi.org/10.1145/2487228.2487237

Ron Kimmel and James A Sethian. 1998. Computing geodesic paths on manifolds. *Proceedings of the national academy of Sciences* 95, 15 (1998), 8431–8435.

Bruno Lévy and Hao Zhang. 2010. Spectral mesh processing. In *ACM SIGGRAPH 2010 Courses*. 1–312.

Moshe Lichtenstein, Gautam Pai, and Ron Kimmel. 2019. Deep eikonal solvers. In *Scale Space and Variational Methods in Computer Vision: 7th International Conference, SSVM 2019, Hofgeismar, Germany, June 30–July 4, 2019, Proceedings 7.* Springer, 38–50.

Roee Litman and Alex M Bronstein. 2016. Spectrometer: Amortized sublinear spectral approximation of distance on graphs. In *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, 499–508.

Charles Loop. 1987. *Smooth Subdivision Surfaces Based on Triangles.* Ph.D. Dissertation.

William E Lorensen and Harvey E Cline. 1998. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field.* 347–353.

Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. 2023. Constructive Solid Geometry on Neural Signed Distance Fields. In *SIGGRAPH Asia 2023 Conference Papers*. 1–12.

Joseph S. B. Mitchell, D. Mount, and C. Papadimitriou. 1987. The Discrete Geodesic Problem. *SIAM J. Comput.* 16, 4 (1987), 647–668.

Patrick Mullen, Fernando De Goes, Mathieu Desbrun, David Cohen-Steiner, and Pierre Alliez. 2010. Signing the Unsigned: Robust Surface Reconstruction from Raw Pointsets. *Computer Graphics Forum* 29, 5 (2010), 1733–1741. https://doi.org/10.1111/j.1467-8659.2010.01782.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2010.01782.x

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.

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph.* 33, 4, Article 135 (jul 2014), 14 pages. https://doi.org/10.1145/2601097.2601154

Giacomo Nazzaro, Enrico Puppo, and Fabio Pellacini. 2022. geoTangle: Interactive Design of Geodesic Tangle Patterns on Surfaces. *ACM Transactions on Graphics* 41, 2 (2022), 12:1–12:17. https://doi.org/10.1145/3487909

Helen Oleynikova, Alexander Millane, Zachary Taylor, Enric Galceran, Juan Nieto, and Roland Siegwart. 2016. Signed distance fields: A natural representation for both mapping and planning. In *RSS 2016 workshop: geometry and beyond-representations,*

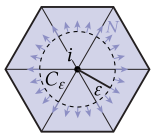
physics, and scene understanding for robotics. University of Michigan.
Stanley Osher, Ronald Fedkiw, and K Piechor. 2004. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.* 57, 3 (2004), B15–B15.
Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
Konrad Polthier and Eike Preuß. 2003. Identifying Vector Field Singularities Using a Discrete Hodge Decomposition. In *Visualization and Mathematics III*, Hans-Christian Hege and Konrad Polthier (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 113–134.
Inigo Quilez. 2008. Raymarching Signed Distance Fields. https://iquilezles.org/articles/raymarchingdf/.
Sudhanshu Rawat and Mantosh Biswas. 2022a. Enhanced Heat Method for Computation of Geodesic Distance on Triangular Meshes. In *2022 IEEE 9th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*. IEEE, 1–5.
Sudhanshu Rawat and Mantosh Biswas. 2022b. Modified Heat Method using GMRES for Geodesic Distance. In *2022 IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*. IEEE, 1–5.
Rohan Sawhney, Ruihao Ye, Johann Korndoerfer, and Keenan Crane. 2020. FCPW: Fastest Closest Points in the West.
Silvia Sellán and Alec Jacobson. 2022. Stochastic Poisson Surface Reconstruction. *ACM Trans. Graph.* 41, 6, Article 227 (nov 2022), 12 pages. https://doi.org/10.1145/3550454.3555441
James A Sethian. 1999. Fast marching methods. *SIAM review* 41, 2 (1999), 199–235.
Nicholas Sharp and Keenan Crane. 2020a. A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum (SGP)* 39, 5 (2020).
Nicholas Sharp and Keenan Crane. 2020b. You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges. *ACM Trans. Graph.* 39, 6 (2020).
Nicholas Sharp, Keenan Crane, et al. 2019a. GeometryCentral: A modern C++ library of data structures and algorithms for geometry processing. https://geometry-central.net/. (2019).
Nicholas Sharp, Mark Gillespie, and Keenan Crane. 2021. Geometry Processing with Intrinsic Triangulations. (2021).
Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019b. Navigating Intrinsic Triangulations. *ACM Trans. Graph.* 38, 4 (2019).
Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019c. The Vector Heat Method. *ACM Trans. Graph.* 38, 3 (2019).
Oded Stein, Max Wardetzky, Alec Jacobson, and Eitan Grinspun. 2020. A Simple Discretization of the Vector Dirichlet Energy. *Computer Graphics Forum* 39, 5 (2020). https://doi.org/10.1111/cgf.14070
Kaiyue Sun and Xiangyang Liu. 2022. Heat method of non-uniform diffusion for computing geodesic distance on images and surfaces. *Multimedia Tools and Applications* 81, 25 (2022), 36293–36308.
Jiong Tao, Juyong Zhang, Bailin Deng, Zheng Fang, Yue Peng, and Ying He. 2019. Parallel and scalable heat methods for geodesic distance computation. *IEEE transactions on pattern analysis and machine intelligence* 43, 2 (2019), 579–594.
Philip Trettner, David Bommes, and Leif Kobbelt. 2021. Geodesic distance computation via virtual source propagation. In *Computer graphics forum*, Vol. 40. Wiley Online Library, 247–260.
Sathamangalam R Srinivasa Varadhan. 1967. On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics* 20, 2 (1967), 431–455.
Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022), 125:1–125:18. https://doi.org/10.1145/3528223.3530139
Hongyi Xu and Jernej Barbič. 2014. Signed Distance Fields for Polygon Soup Meshes. *Graphics Interface 2014* (2014).
Lior Yariv, Omri Puny, Natalia Neverova, Oran Gafni, and Yaron Lipman. 2023. Mosaic-SDF for 3D Generative Models. *arXiv preprint arXiv:2312.09222* (2023).
Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh arrangements for solid geometry. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–15.

A ADDITIONAL SOURCE GEOMETRY

Here we derive discretizations for isolated point sources and unoriented curves.

A.1 Point Sources

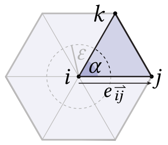
To compute unsigned distance to point sources, we encode a radially symmetric vector field centered at each point source. We adopt the approach of Sharp et al. [2019c, App. A] and consider a small geodesic circle C_ε of radius $\varepsilon > 0$ centered on a point source at vertex i, and take the limit as $\varepsilon \to 0$. We let $\mu_\varepsilon := (\Theta_i \varepsilon)^{-1}\mathcal{H}^1_{C_\varepsilon}$ be a measure of unit mass supported on C_ε, where $\mathcal{H}^1_{C_\varepsilon}$ is the Hausdorff measure on C_ε, and $\Theta_i := \sum_{ijk>i} \theta_i^{jk}$ be the discrete angle sum around vertex i. We let N denote the outward unit normals to C_ε. We integrate the vector-valued measure $N\mu_\varepsilon$ against CR basis functions for each edge $e \in E$ (Section 5.6).



We restrict our attention to a single triangle ijk containing i and e; the final contribution to the ij entry is the sum of contributions from all ijk incident on e. W.l.o.g. we let $e = ij$. We express all quantities in complex numbers with respect to the polar coordinate system with origin at vertex i and real axis along $\overrightarrow{ij}$, parameterizing C_ε by the angle θ between $x \in C_\varepsilon$ and $e_{\overrightarrow{ij}}$. Expressed in this basis, the unit normal $N(\theta)$ at $x = \varepsilon e^{\imath\theta}$ is simply $e^{\imath\theta}$. We let $\alpha := \theta_i^{jk}$, and evaluate

$$\lim_{\varepsilon\to 0}\int_{ijk}\langle n, \psi_{ij}\rangle \, d\mu_\varepsilon = \lim_{\varepsilon\to 0}\frac{1}{\Theta_i\varepsilon}\int_0^\alpha N(\theta)\varepsilon \, d\theta \frac{(1-e^{\imath\alpha})\imath}{\Theta_i}.$$

The contribution to edge ij multiplies this quantity by a sign $s_{\overrightarrow{ij}} \in \{+1, -1\}$ equal to $e^{\imath 0} = +1$ if $\overrightarrow{ij}$ agrees with the global orientation of ij, and $e^{\imath\pi} = -1$ otherwise.

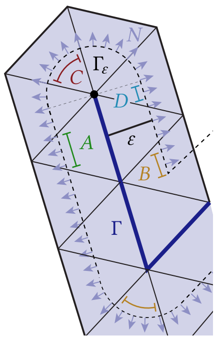


To compute the contributions to edge jk (*resp.* ki), the only difference is that $x \in C_\varepsilon$ and $n(\theta)$ must be expressed relative to the tangent basis at edge jk (*resp.* ki) instead of ij. This amounts to rotating the coordinate system by $r_{ij\to jk}$ (Equation 8). We arrive at the per-face contributions

$$\begin{aligned}(X_0)_{ij} &= s_{ij}\frac{(1-e^{\imath\alpha})\imath}{\Theta_i}\\ (X_0)_{jk} &= s_{ij}r_{ij\to jk}\frac{(1-e^{\imath\alpha})\imath}{\Theta_i}\\ (X_0)_{ki} &= s_{ij}r^{-1}_{ki\to ij}\frac{(1-e^{\imath\alpha})\imath}{\Theta_i}.\end{aligned} \tag{19}$$

If we were to use basis functions at vertices, there would not be enough degrees of freedom to encode radially symmetric vector fields at vertices; Sharp et al. [2019c, App. A] ameliorate this shortcoming by arbitrarily taking the limit as $\varepsilon \to 1$ instead of zero, but this limit is not scale-invariant like ours.

A.2 Unoriented Curves



Here we consider open curves to which we compute unsigned distance. Similar to Appendix A.1, we consider a geodesic ε-offset Γ_ε of the (piecewise linear) curve Γ, with a measure of unit density concentrated on Γ_ε, and take $\varepsilon \to 0$. For simplicity, we consider unoriented curves that lie along edges of the mesh. Then Γ_ε can generically be decomposed into four types of curves: (A) linear segments that intersect faces incident on edges of the curve Γ; (B) segments that intersect faces incident on interior vertices of Γ; (C) circular arcs that intersect faces incident on endpoints of Γ; and (D) linear segments that intersect faces incident on endpoints of Γ (inset).

As $\varepsilon \to 0$, the contributions of type-B and D segments go to zero. A type-A segment lying within face ijk, where edge ij lies on the curve Γ, converges to an oriented curve along edge ij as $\varepsilon \to 0$; its contributions to each edge within ijk are given by Equation 11. For type-C circular arcs, we obtain the same per-face formulas as Equation 19, except we normalize not by the total angle sum around the endpoint vertex but by the sum of corner angles of faces that intersect the circular portion of Γ_ε, and weight by the length of the adjacent edge.

B CROUZEIX-RAVIART IN 3D

Here we derive the Crouzeix-Raviart Laplace and mass matrices for tetrahedral meshes. In general, Crouzeix-Raviart basis functions are piecewise-linear and associated with the barycenters of codimension-one simplices in an n-dimensional simplex. The basis function θ_i associated with the $(n-1)$-dimensional face σ_i opposite vertex i has value 1 on σ_i, and value $1-n$ at i. Within each $(n-1)$-dimensional simplex incident on σ_i, the function θ_i is linear and is defined as $\theta_i(x) = 1 - n\lambda_i(x)$ where $\lambda_i(x)$ is the value of the hat function associated with vertex i at point x [Ern and Guermond 2004, §1.2]. We use φ_{ijk} to denote the basis function associated with triangle face ijk in a tet mesh.

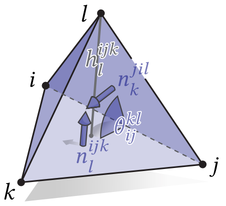
B.1 Scalar Laplacian

The $|F| \times |F|$ Crouzeix-Raviart Laplace matrix L of a tet mesh $M = (V, E, F, T)$ has entries given by

$$\mathsf{L}_{f,f'} = \int_M \langle \nabla\varphi_f(x), \nabla\varphi_{f'}(x)\rangle \,\mathrm{d}x$$

which will be nonzero only if faces f, f' are adjacent, so w.l.o.g. we let $f := ijk$, $f' := jil$ and compute

$$\begin{aligned}\mathsf{L}_{ijk,jil} &= \int_M \langle \nabla\varphi_{ijk}(x), \nabla\varphi_{jil}(x)\rangle \,\mathrm{d}x \\ &= \int_M \langle \nabla(1-3\lambda_l(x)), \nabla(1-3\lambda_k(x))\rangle \,\mathrm{d}x \\ &= \sum_{ijkl > ijk, jil} 9 \int_{ijkl} \langle \nabla\lambda_l(x), \nabla\lambda_k(x)\rangle \,\mathrm{d}x.\end{aligned}$$



The gradient $\nabla\lambda_l = n_l^{ijk}/h_l^{ijk}$, where n_l^{ijk} denotes the unit normal to face ijk opposite vertex l, and h_l^{ijk} the height of the tetrahedron with apex l and base ijk (and similarly for $\nabla\lambda_k$). Since the gradients $\nabla\lambda_l, \nabla\lambda_k$ are constant per tet, the inner integral is equal to

$$|ijkl| \frac{1}{h_l^{ijk} h_k^{jil}} \cos(\pi - \theta_{ij}^{kl}) = -|ijkl| \frac{|ijk|}{3|ijkl|} \frac{|jil|}{3|ijkl|} \cos\theta_{ij}^{kl} \qquad (20)$$

where θ_{ij}^{kl} denotes the dihedral angle at edge ij opposite edge kl. Since the volume of the tetrahedron $ijkl$ can be expressed $|ijkl| = \frac{2}{3|ij|}|ijk||jil| \sin\theta_{ij}^{kl}$, we obtain

$$\begin{aligned}\mathsf{L}_{ijk,jil} &= -\tfrac{3}{2}|ij| \cot\theta_{ij}^{kl}, && \forall ijk \in F,\ ijkl > ijk \\ \mathsf{L}_{ijk,ikl} &= -\tfrac{3}{2}|ki| \cot\theta_{ki}^{jl}, && \forall ijk \in F,\ ijkl > ijk \\ \mathsf{L}_{ijk,jlk} &= -\tfrac{3}{2}|jk| \cot\theta_{jk}^{il}, && \forall ijk \in F,\ ijkl > ijk \\ \mathsf{L}_{ijk,ijk} &= -\textstyle\sum_{ijkl>ijk} \mathsf{L}_{ijk,jil} + \mathsf{L}_{ijk,ikl} + \mathsf{L}_{ijk,jlk}, && \forall ijk \in F\end{aligned} \qquad (21)$$

For both $n = 2, 3$, the n-dimensional CR Laplacian has entries n^2 times the corresponding entries of the so-called primal vertex Laplacian Alexa et al. [2020], since CR basis functions are equal to the hat functions on medial n-simplices.

B.2 Divergence Operator

As in 2D, we average vectors X from triangles to tetrahedra via

$$\mathsf{X}_{ijkl} = \frac{1}{4}\left(X_{ijk} + X_{jil} + X_{ikl} + X_{jlk}\right),$$

which corresponds to evaluating the CR-interpolated field at the barycenter. Letting $\mathsf{Y}_{ijkl} := \mathsf{X}_{ijkl}/\|\mathsf{X}_{ijkl}\|$ in each tet, the discrete divergence is then

$$(\nabla\cdot\mathsf{Y})_i = \sum_{ijkl>i} |jlk| n^{jlk} \cdot \mathsf{Y}_{ijkl}. \qquad (22)$$

B.3 Mass Matrix

The $|F| \times |F|$ Crouzeix-Raviart mass matrix M has nonzero entries given by

$$\begin{aligned}\mathsf{M}_{ijk,jil} &= \int_M \langle \varphi_{ijk}(x), \varphi_{jil}(x)\rangle \,\mathrm{d}x \\ &= \sum_{ijkl>ijk,jil} 6|ijkl| \int_0^1 \int_0^{1-u} \int_0^{1-t-u} (1-3\lambda_l(x))(1-3\lambda_k(x)) \,\mathrm{d}s\,\mathrm{d}t\,\mathrm{d}u\end{aligned}$$

where x is parameterized using barycentric coordinates of tet $ijkl$ with vertex positions v_i, v_j, v_k, v_l, as $x = sv_i + tv_j + uv_k + (1-s-t-u)v_l$. We obtain entries of the symmetric matrix as

$$\begin{aligned}\mathsf{M}_{ijk,jil} = \mathsf{M}_{ijk,ikl} = \mathsf{M}_{ijk,jlk} &= -\tfrac{1}{20}|ijkl|, && \forall ijkl \in T \\ \mathsf{M}_{ijk,ijk} &= \tfrac{2}{5}\textstyle\sum_{ijkl>ijk} |ijkl|, && \forall ijk \in F.\end{aligned} \qquad (23)$$

C PSEUDOCODE

We give pseudocode for surface domains, expressed via a halfedge mesh data structure encoding a triangle mesh $M = (V, E, F)$. We use $\overrightarrow{ij}$ to denote the halfedge from i to j. We assume only that meshes have been specified via intrinsic quantities including edge lengths and corner angles, which we denote using the notation defined in §5.1. Subroutines not defined here are described in the list below. For simplicity, we assume here that M is oriented.

- Orientation($\overrightarrow{ij}$) — returns +1 if the orientation of halfedge $\overrightarrow{ij}$ matches the canonical orientation of its edge ij, and −1 otherwise.
- Face(p) — returns a face ijk that the barycentric point p lies within.
- SharedHalfedge(A, B) — returns the halfedge going from element A to B, which may be vertices or barycentric points, if any.
- SharedFace(A, B) — returns a face shared by mesh elements A and B, if any. The elements A and B may be vertices, edges, faces, or barycentric points.
- BarycentricVector(p_A, p_B) — returns a barycentric vector defined by the barycentric points p_A and p_B as its endpoints. If p_A and p_B coincide with vertices, the barycentric vector lies on an edge; otherwise, it lies in a face.
- BarycentricVectorInFace($\overrightarrow{ij}, ijk$) — returns the barycentric vector defined by the endpoints of halfedge $\overrightarrow{ij}$, with coordinates expressed with respect to face ijk.
- BarycentricCoordsInFace(p, ijk) — returns the barycentric coordinates of the barycentric point p with respect to face ijk.
- BarycentricCoordsInSomeFace(p) — returns the barycentric coordinates of the barycentric point p with respect to one its containing faces, along with the face itself.
- BarycentricCoordsInFace(v, ijk) — returns the barycentric coordinates of the barycentric vector v with respect to face ijk.
- Norm(M, v) — returns the norm of the barycentric vector v defined on triangle mesh M.
- Dot(M, v_A, v_B) — returns the inner product $\langle v_A, v_B \rangle \in \mathbb{R}$ between two barycentric vectors v_A, v_B defined on triangle mesh M.
- Rotated90(M, v) — returns the barycentric vector v, rotated counterclockwise 90° in its local tangent plane on mesh M.
- SolveSparseSquare(A, b) — solves the sparse square linear system Ax = b, returning x.
- SolveSparsePositiveSemidefinite(A, b) — solves the sparse positive semidefinite linear system Ax = b, returning x (and picking an arbitrary shift if A has constants in its null space).

Algorithm 1 SolveGeneralizedSignedDistance(M, Ω, t, C)

```
Input: Points and/or curves $\Omega$ on a triangle mesh $M$, diffusion time $t$, and constraints $C$.
Output: The generalized signed distance function $\phi$ to $\Omega$.
1: $\mathsf{X}_t \leftarrow$ IntegrateVectorHeatFlow$(M, \Omega, t)$
2: $\mathsf{Y}_t \leftarrow$ Normalize$(\mathsf{X}_t)$
3: $\phi \leftarrow$ IntegrateVectorField$(M, \mathsf{Y}_t, C)$
4: return $\phi$
```

Algorithm 2 IntegrateVectorHeatFlow(M, Ω, t)

```
Input: Integrate the vector heat flow in Equation 12 for time $t$ on the triangle mesh $M = (V, E, F)$, with initial conditions defined by the geometry $\Omega$.
Output: The diffused vector field $\mathsf{X}_t \in \mathbb{C}^{|E|}$.
1: $\mathsf{L}^\nabla \leftarrow$ CrouzeixRaviartConnectionLaplacian$(M)$
2: $\mathsf{M} \leftarrow$ CrouzeixRaviartMassMatrix$(M)$
3: $\mathsf{X}_0 \leftarrow$ BuildSource$(M, \Omega)$
4: $\mathsf{X}_t \leftarrow$ SolveSparsePositiveSemidefinite$(\mathsf{M} + t\mathsf{L}^\nabla, \mathsf{X}_0)$
5: return $\mathsf{X}_t$
```

Algorithm 3 Normalize(M, X)

```
Input: A vector field $\mathsf{X} \in \mathbb{C}^{|E|}$ expressed in the edge basis defined in §5.2, defined on triangle mesh $M = (V, E, F)$.
Output: The normalized vector field $\mathsf{Y} \in \mathbb{R}^{|F| \times 3}$, sampled onto face barycenters and encoded via barycentric vectors.
1: $\mathsf{Y} \leftarrow 0^{|F| \times 3}$
2: for $pqr \in F$ do
3:     $\mathsf{y} \leftarrow 0^3$
4:     for $ijk \in C(pqr)$ do                      ▷C: circular shifts
5:         $s_{\overrightarrow{ij}} \leftarrow$ Orientation$(\overrightarrow{ij})$
6:         $\tau \leftarrow$ BarycentricVectorInFace$(\overrightarrow{ij}, pqr) \cdot s_{\overrightarrow{ij}}$
7:         $v \leftarrow$ Rotated90$(M, \tau)$
8:         $\tau$ /= Norm$(M, \tau)$
9:         $v$ /= Norm$(M, v)$
10:        $\lambda_\tau \leftarrow$ BarycentricCoordsInFace$(\tau, pqr)$
11:        $\lambda_v \leftarrow$ BarycentricCoordsInFace$(v, pqr)$
12:        $\mathsf{y}$ += Re$(\mathsf{X}_{ij}) \cdot \lambda_\tau$
13:        $\mathsf{y}$ += Im$(\mathsf{X}_{ij}) \cdot \lambda_v$
14:     $\mathsf{Y}_{pqr} \leftarrow \mathsf{y}$
15: return $\mathsf{Y}$
```

Algorithm 4 IntegrateVectorField(M, X, C)

```
Input: A vector field $\mathsf{X} \in \mathbb{R}^{|F| \times 3}$ defined on a triangle mesh $M = (V, E, F)$, and constraints $C$.
Output: The solution $\phi \in \mathbb{R}^{|E|}$ to the Poisson problem in Equation 13 satisfying the constraints $C$ (§7).
1: $\mathsf{L} \leftarrow$ CotanLaplacian$(M)$
2: $\mathsf{b} \leftarrow$ Divergence$(M, \mathsf{X})$
3: if $C = \varnothing$ then
4:     $\phi \leftarrow -$SolveSparsePositiveSemidefinite$(\mathsf{L}, \mathsf{b})$
5:     $\phi \leftarrow$ Shift$(\phi, \Omega)$
6:     return $\phi$
7: if $C =$ PreserveZeroLevelset then
8:     $\mathsf{A} \leftarrow$ ConstraintMatrix$(\Omega)$
9:     $\mathsf{u} \leftarrow -$SolveSparseSquare$\left(\begin{bmatrix} \mathsf{L} & \mathsf{A}^\mathsf{T} \\ \mathsf{A} & 0 \end{bmatrix}, \begin{bmatrix} \mathsf{b} \\ 0 \end{bmatrix}\right)$
10:    $\phi \leftarrow$ Shift$(\mathsf{u}_{:|E|}, \Omega)$
11:    return $\phi$
```

Algorithm 5 CrouzeixRaviartConnectionLaplacian(M)

```
Input: A triangle mesh $M = (V, E, F)$.
```

```
Output: The Crouzeix-Raviart connection Laplacian $L^\nabla \in \mathbb{C}^{|E|\times|E|}$ (§5.4).
1:  $L^\nabla \leftarrow 0^{|E|\times|E|}$                ▷initialize empty sparse complex matrix
2:  for $pqr \in F$ do
3:      for $ijk \in C(pqr)$ do                  ▷C: circular shifts
4:          $w \leftarrow 2\cot\theta_j^{ki}$
5:          $r_{ij,jk} \leftarrow$ EdgeRotation$(ij, jk)$
6:          $L^\nabla_{ij,ij}$ += $w$
7:          $L^\nabla_{jk,jk}$ += $w$
8:          $L^\nabla_{ij,jk}$ −= $w \cdot \overline{r}_{ij\to jk}$
9:          $L^\nabla_{jk,ij}$ −= $w \cdot r_{ij\to jk}$
10: return $L^\nabla$
```

Algorithm 6 CrouzeixRaviartMassMatrix(M)

```
Input: A triangle mesh $M = (V, E, F)$.
Output: The Crouzeix-Raviart mass matrix $M \in \mathbb{C}^{|E|\times|E|}$ (§B.3).
1:  $M \leftarrow 0^{|E|\times|E|}$                ▷initialize empty sparse complex matrix
2:  for $pqr \in F$ do
3:      for $ij < pqr$ do $M_{ij,ij}$ += $\frac{|ijk|}{3}$
4:  return $M$
```

Algorithm 7 CotanLaplacian(M)

```
Input: A triangle mesh $M = (V, E, F)$.
Output: The positive definite cotan Laplacian $L \in \mathbb{R}^{|V|\times|V|}$.
1:  $L \leftarrow 0^{|V|\times|V|}$                ▷initialize empty sparse matrix
2:  for $pqr \in F$ do
3:      for $ijk \in C(pqr)$ do                  ▷C: circular shifts
4:          $w \leftarrow \frac{1}{2}\cot\theta_k^{ij}$
5:          $L_{i,i}$ += $w$
6:          $L_{j,j}$ += $w$
7:          $L_{i,j}$ −= $w$
8:          $L_{j,i}$ −= $w$
9:  return $L$
```

Algorithm 8 Divergence(M, X)

```
Input: A triangle mesh $M = (V, E, F)$, and vector field $X \in \mathbb{C}^{|F|}$.
Output: The finite-element divergence $b := \nabla \cdot X \in \mathbb{R}^{|V|}$, defined per vertex.
1:  $b \leftarrow 0^{|V|}$
2:  for $i \in V$ do
3:      for $ijk > i$ do
4:          $v_A \leftarrow$ BarycentricVectorInFace$(\overrightarrow{ij}, ijk)$
5:          $v_B \leftarrow$ BarycentricVectorInFace$(\overrightarrow{ki}, ijk)$
6:          $d_A \leftarrow$ Dot$(M, v_A, X_{ijk})$
7:          $d_B \leftarrow$ Dot$(M, v_B, X_{ijk})$
8:          $b_i$ += $\frac{1}{2}\cot\theta_k^{ij} \cdot d_A + \frac{1}{2}\cot\theta_j^{ki} \cdot d_B$
9:  return $b$
```

Algorithm 9 ConstraintMatrix(Ω)

```
Input: Source geometry $\Omega$ considered as a set of barycentric points $\{p_i\}$ on triangle mesh $M = (V, E, F)$.
Output: The constraint matrix $A \in \mathbb{R}^{m\times|V|}$ defined in Equation 14, where $m$ is the number of constraints.
1:  $m \leftarrow 0$
2:  $\lambda_0, abc \leftarrow$ BarycentricCoordsInSomeFace$(p_0)$
3:  for $p \in \Omega$ do
4:      $ijk \leftarrow$ Face$(p)$
5:      $\lambda \leftarrow$ BarycentricCoordsInFace$(p, ijk)$
6:      for $l < ijk$ do $C_{m,l}$ += $\lambda_l$
7:      for $l < abc$ do $C_{m,l}$ −= $(\lambda_0)_l$
8:      $m$ += 1
9:  return $C$
```

Algorithm 10 BuildSource(M, Ω)

```
Input: Source geometry $\Omega = \{\Gamma, P\}$ consisting of a collection of curves $\Gamma$ and points $P$, defined on triangle mesh $M = (V, E, F)$ (§5.6).
Output: The r.h.s. $X_0 \in \mathbb{C}^{|E|}$ to Equation 12.
1:  $X_0 \leftarrow 0^{|E|}$                ▷initialize empty complex vector
2:  $X_0$ += BuildOrientedCurveSources$(M, \Gamma)$
3:  $X_0$ += BuildUnorientedPointSources$(M, P)$
4:  return $X_0$
```

Algorithm 11 BuildOrientedCurveSources(M, Γ)

```
Input: A collection of oriented curves $\Gamma = \{\gamma_i\}$ on triangle mesh $M = (V, E, F)$ consisting of linear segments $\gamma_i$, each defined by barycentric points sharing a face (§5.6).
Output: A source term $X_0 \in \mathbb{C}^{|E|}$ encoding $\Gamma$.
1:  $X_0 \leftarrow 0^{|E|}$                ▷initialize empty complex vector
2:  for $\gamma = (p_A, p_B) \in \Gamma$ do
3:      $\ell \leftarrow$ Length$(\gamma)$
4:      $\overrightarrow{ij} \leftarrow$ SharedHalfedge$(p_A, p_B)$
5:      if $ij$ = Null then
6:          $ijk \leftarrow$ SharedFace$(p_A, p_B)$
7:          for $ij < ijk$ do
8:              $(X_0)_{ij}$ += $\ell \cdot$ CurveNormal$(M, \gamma, ij)$
9:      else
10:         $n \leftarrow \imath \cdot$ Orientation$(\overrightarrow{ij})$
11:         $(X_0)_{ij}$ += $\ell \cdot n$
12: return $X_0$
```

Algorithm 12 BuildUnorientedPointSources(M, P)

```
Input: A collection of vertices $P$ on triangle mesh $M = (V, E, F)$.
Output: A source term $X_0 \in \mathbb{C}^{|E|}$ encoding $P$.
1:  $X_0 \leftarrow 0^{|E|}$                ▷initialize empty complex vector
2:  for $i \in P$ do
3:      ▷Compute angle sum.
4:      $\Theta \leftarrow 0$
5:      for $\,_i^{jk} < i$ do $\Theta$ += $\theta_i^{jk}$
6:      ▷Add contributions per-face.
7:      for $\,_i^{jk} < i$ do
8:          $s_{\overrightarrow{ij}} \leftarrow$ Orientation$(\overrightarrow{ij})$
9:          $s_{\overrightarrow{jk}} \leftarrow$ Orientation$(\overrightarrow{jk})$
```

```
10:         $s_{\overrightarrow{ki}} \leftarrow$ Orientation($\overrightarrow{ki}$)
11:         $r_{\overrightarrow{ij}\to\overrightarrow{jk}} \leftarrow$ HalfedgeRotation($\overrightarrow{ij}, \overrightarrow{jk}$)
12:         $r_{\overrightarrow{ki}\to\overrightarrow{ij}} \leftarrow$ HalfedgeRotation($\overrightarrow{ki}, \overrightarrow{ij}$)
13:         $n \leftarrow \frac{\iota(1-e^{\iota\theta_i^{jk}})}{\Theta}$
14:         $(\mathsf{X}_0)_{ij}$ += $s_{\overrightarrow{ij}} \cdot n$
15:         $(\mathsf{X}_0)_{jk}$ += $s_{\overrightarrow{jk}} \cdot \overline{r}_{\overrightarrow{ij}\to\overrightarrow{jk}} \cdot n$
16:         $(\mathsf{X}_0)_{ki}$ += $s_{\overrightarrow{ki}} \cdot r_{\overrightarrow{ki}\to\overrightarrow{ij}} \cdot n$
17: return $\mathsf{X}_0$
```

Algorithm 13 Shift(M, f, Ω)

Input: A function $\mathsf{f} \in \mathbb{R}^{|V|}$ and source geometry $\Omega = \{\Gamma, P\}$, defined on triangle mesh $M = (V, E, F)$.

Output: The function $\mathsf{g} \in \mathbb{R}^{|V|}$ shifted to average zero along Ω.

```
1: $c \leftarrow 0$
2: $L \leftarrow 0$
3: for $\gamma \in \Gamma$ do
4:     $\ell \leftarrow$ Length($M, \gamma$)
5:     $ijk, \lambda \leftarrow$ Midpoint($\gamma$)
6:     for $l < ijk$ do $c \leftarrow \ell \cdot \lambda_l \cdot \mathsf{f}_l$
7:     $L$ += $\ell$
8: for $p \in P$ do
9:     $ijk \leftarrow$ Face($p$)
10:    $\lambda \leftarrow$ BarycentricCoordsInFace($p, ijk$)
11:    for $l < ijk$ do $c$ += $\mathsf{f}_l \cdot \lambda_l$
12:    $L$ += 1
13: $c$ /= $L$
14: $\mathsf{g} \leftarrow \mathsf{f} - c \cdot \mathbf{1}^{|V|}$
15: return g
```

Algorithm 14 EdgeRotation(ij, jk)

Input: Two edges ij and jk in face ijk.

Output: The complex number encoding the smallest rotation from the local coordinate basis at edge ij to that of edge jk. (§5.4).

```
1: $r_{\overrightarrow{ij}\to\overrightarrow{jk}} \leftarrow$ HalfedgeRotation($\overrightarrow{ij}, \overrightarrow{jk}$)
2: $s_{ij\to jk} \leftarrow$ Orientation($\overrightarrow{ij}$) $\cdot$ Orientation($\overrightarrow{jk}$)
3: $r_{ij\to jk} \leftarrow s_{ij\to jk} \cdot \overline{r}_{\overrightarrow{ij}\to\overrightarrow{jk}}$
4: return $r_{ij\to jk}$
```

Algorithm 15 HalfedgeRotation($\overrightarrow{ij}, \overrightarrow{jk}$)

Input: Two halfedges $\overrightarrow{ij}$ and $\overrightarrow{jk}$ in face ijk.

Output: The complex number encoding the smallest rotation from $e_{\overrightarrow{ij}}$ to $e_{\overrightarrow{jk}}$.

```
1: $r_{\overrightarrow{ij}\to\overrightarrow{jk}} \leftarrow -e^{-\iota\theta_j^{ki}}$
2: return $r_{\overrightarrow{ij}\to\overrightarrow{jk}}$
```

Algorithm 16 CurveNormal(M, ij)

Input: A curve segment $\gamma = (p_A, p_B)$ specified by two barycentric points p_A and p_B, and edge ij defined on triangle mesh M.

Output: The complex number $n \in \mathbb{C}$ encoding the unit normal to γ, expressed w.r.t. the local basis of ij (§5.4).

```
1: $\beta \leftarrow$ BarycentricVector($i, j$)
2: $\tau \leftarrow$ BarycentricVector($p_A, p_B$)
3: $\nu \leftarrow$ Rotated90($M, \tau$)
4: $\tau$ /= Norm($M, \tau$)
5: $\nu$ /= Norm($M, \nu$)
6: $n \leftarrow$ Dot($M, \nu, \beta$) $+ \iota \cdot$ Dot($M, \tau, \beta$)
7: return $n$
```

Algorithm 17 Length(M, γ)

Input: A curve segment $\gamma = (p_A, p_B)$ specified by two barycentric points p_A and p_B, defined on the triangle mesh M.

Output: The length of γ.

```
1: $\nu \leftarrow$ BarycentricVector($p_A, p_B$)
2: $\ell \leftarrow$ Norm($M, \nu$)
3: return $\ell$
```

Algorithm 18 Midpoint(γ)

Input: A curve segment $\gamma = (p_A, p_B)$ specified by two barycentric points p_A and p_B.

Output: The barycentric point at the midpoint of γ, expressed via its containing face ijk and barycentric coordinates w.r.t. ijk.

```
1: $ijk \leftarrow$ SharedFace($p_A, p_B$)
2: $\lambda_A \leftarrow$ BarycentricCoordsInFace($p_A, ijk$)
3: $\lambda_B \leftarrow$ BarycentricCoordsInFace($p_B, ijk$)
4: return $ijk, \frac{1}{2}(\lambda_A + \lambda_B)$
```

Received January 20XX