Algorithms for Generalized Signed Distance and Winding Numbers

Nicole Feng

CMU-CS-26-104

June 2026

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Keenan Crane, Chair
Ioannis Gkioulekas
Nancy Pollard
Christopher Wojtan (Institute of Science and Technology Austria)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by: the National Science Foundation under award numbers 1943123, 2212290 and 2504890.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: geometry processing, signed distance, winding numbers, reconstruction, discrete differential geometry

Abstract

This thesis presents algorithms for generalized inside/outside computation (via *winding numbers*) and signed distance computation. By "generalized", we mean that these algorithms make geometric inferences from imperfect data comprising incomplete, inaccurate, or ambiguous observations or representations of shapes. In other words, these algorithms generalize from imperfect data and implicitly approximate the true underlying curve or surface. A theme is that generalization can often be achieved by processing globally-defined functions encoding the geometry of interest, rather than the original, defective curve or surface. For both inside/outside and signed distance computation we can unlock further control over geometry and topology by processing higher-order derivatives of these functions. In many cases, we can also re-cast our algorithms, formulated in terms of smooth functions, onto different discretizations and geometric data structures. Another theme is that inside/outside and signed distance computation are closely related problems; towards this end, we provide a formalization of their relationship that justifies the design of our algorithms.

Acknowledgments

First, thank you to my advisor Keenan. He invested a huge amount of time, effort, and energy into teaching me, including and especially through COVID. In the world of science, I appreciate his curiosity and ability to get the details right while championing accessibility.

Thank you also to Nancy, Yannis, and Chris, the other members of my committee. Nancy has a great ability to get to stuff that matters, and has been supportive of me throughout my PhD. Yannis is the source of most of my knowledge about modern rendering. Chris asks me hard questions and doesn't seem to let go of a loose end. I would also like very much to thank Pierre Alliez, who generously hosted me twice in Sophia-Antipolis during my PhD, and is a great generator of ideas and enthusiasm.

My time at CMU has been one of the most formative in my life in terms of personal growth and happiness, thanks to the members of the Geometry Collective: Nick, Thea, Kai, Daniel, Hesper, Ethan, Rohan, Mark, Hossein, Olga, Denise, Zoë, Josua, Simon, Etienne, Jacopo, Josh, and Jiří. I can't thank you all enough for your relentless intellectual curiosity and empathy you've shared with me. My growth has been driven primarily by being around you all. The Geometry Collective is a very special environment — one I'll probably be trying to re-create for the rest of my life. The CMU Graphics Lab in general is full of fun people (and animals): Arjun, Bharath, Arjun and Snowy, Dorian, Jenny, Jim, Oscar and cats, Sreekar, Tanli, and Yuichi. There are many others not listed here. Outside of CMU, I owe a lot of my happiness to the Pittsburgh Pharaoh Hounds, especially Laura, Julia, and the Hounds women. Thank you to Tolson as well.

Thank you to Al Barr, Peter Schröder, Julie Dorsey, and Thomas Haley for believing in me. Peter and Julie are responsible for building the foundation of my research skills during my undergrad. Al and Peter were extremely patient and generous with their guidance even though I wasn't a star student, and yet they gave me as much attention as anyone else. I would also like to thank Holly Rushmeier for doing the same, and introducing me to a lot of graphics when I was just getting started.

Finally, I'm lucky to have the best family: Lijuan Shi, Xuwu Feng, Emily, Kathleen, and Nick.

Contents

1 Introduction **1**

2 Background **3**
- 2.1 Notation . . . 3
- 2.2 Just the basics of differential geometry . . . 4
- 2.3 The inside and outside of curves and surfaces . . . 7
  - 2.3.1 Curve orientations . . . 8
  - 2.3.2 Homology & cohomology . . . 8
  - 2.3.3 Winding numbers . . . 10
- 2.4 Geodesic distance . . . 12

3 Theoretical foundations of generalized signed distance **15**
- 3.1 A unification of winding numbers, Poisson surface reconstruction, and signed distance . . . 16
- 3.2 Convolutional distance approximations . . . 19
- 3.3 Fundamental limitations of convolutional distance . . . 21
- 3.4 Relation to kernel methods . . . 22

4 Generalized signed distance through vector diffusion **27**
- 4.1 The asymptotics of vector diffusion . . . 27
- 4.2 Vector diffusion for generalized signed distance . . . 29
- 4.3 The signed heat method . . . 31
- 4.4 Signed heat method results . . . 33
- 4.5 Accuracy and performance of the signed heat method . . . 36

5 Pointwise evaluation of generalized signed distance **39**
- 5.1 The points as tori (PAT) method . . . 42
  - 5.1.1 Torus fitting . . . 42
  - 5.1.2 Learning per-neighborhood coefficients . . . 43
  - 5.1.3 Evaluating signed distance . . . 44
- 5.2 Results of PAT . . . 46
- 5.3 Accuracy and performance of PAT . . . 46
- 5.4 Limitations of PAT . . . 49

5.4.1 Robustness to large amounts of noise or outliers . . . 49
5.4.2 Non-conservative distance fields for perfect geometry . . . 50
5.4.3 Out-of-distribution behavior . . . 50
5.4.4 Enforcing constraints . . . 50
5.5 Design choices behind PAT . . . 51
5.5.1 Failed approaches . . . 51
5.5.2 Why learning? . . . 52
5.5.3 Why not end-to-end learning? . . . 53
5.5.4 Why tori? . . . 53
5.5.5 An appeal for small, judicious use of learning . . . 54
5.5.6 Future extensions of PAT . . . 54

6 Generalized inside/outside **57**
6.1 Surface winding numbers . . . 57
6.1.1 Differentiating and integrating jump harmonic functions . . . 59
6.1.2 Derivative and curve decomposition . . . 61
6.2 Boundary behavior of surface winding numbers . . . 62
6.3 Accuracy and performance of surface winding numbers . . . 63
6.4 Extensions to the surface winding numbers algorithm . . . 65
6.4.1 Nonbounding loops and discontinuous distance . . . 67

7 Practical notes **69**
7.1 Winding numbers vs. generalized signed distance for surface reconstruction . . . 69
7.2 Empirical conditions on input . . . 73
7.3 Alternative approaches for perfect geometry . . . 77

8 Conclusion **79**
8.1 Using geometric methods to inform data-driven methods, and vice versa . . . 80
8.2 Lingering questions . . . 82

9 Bibliography **85**

List of Figures

2.1 Is the point p "inside" or "outside" the curve Γ? On surfaces, this question does not always have a meaningful answer. 8

2.2 For curves Γ in the plane, the winding number function $w_\Gamma(p)$ gives the number of times the curve Γ wraps around any given point p. 10

3.1 As the screening amount λ goes to 0, the solution w to Equation 3.8 converges to a jump harmonic function describing inside-outside (*left*). As λ increases, w converges to a signed distance function via a log transform (*right*). However, this formula only works for closed geometry and fails for sampled geometry like point clouds (Section 3.3). 18

3.2 Regularizing the kernel in Equation 3.15 is equivalent to choosing the function h, which unfortunately has no effect asymptotically. Here, we use an exponential kernel with no regularization (*left*), with isotropic Gaussian regularization (*center*), and with anisotropic Gaussian regularization (*right*), which all produce non-generalized distance for $\lambda = 100$. 21

3.3 In naive convolutional distance formulas (Equation 3.15), the parameter λ is coupled to both regression quality and distance accuracy, in opposite directions: large λ values are needed to obtain better eikonality, but overfits distance to the point set (*left*); using smaller λ at least interpolates the data points, but isn't close to a distance function (*center, right*). These figures use Equation 3.13 with regularization [Chen et al. 2024a] using $\varepsilon = 0.2$, for stability; even so, it is easy to get artifacts when λ is smaller. 22

4.1 A Dirac-delta distribution gets blurred out by scalar heat flow. 28

4.2 *Left:* An initial vector (large) is diffused in $\mathbb{R}^d$, yielding exponential decay in its magnitude but preserving its orientation. *Right:* The same picture intuitively applies on manifolds. 28

4.3 The three steps of the signed heat method. 30

4.4 *Top:* Short-time scalar diffusion concentrates data tightly around the initial source, preventing generalization. *Bottom:* The signed heat method instead relies on short-time *vector* diffusion. While the magnitudes of the vectors remain tightly concentrated, their *orientations* are accurate — so one can normalize and integrate this vector field to obtain generalized signed distance. 30

4.5 *Left:* The signed heat method extends to polygon meshes, point clouds, and digital surfaces. *Right:* The method extends to volumetric domains, such as regular grids in $\mathbb{R}^3$. Contouring this function yields well-behaved and evenly-spaced offset surfaces. Digital surface meshes are from Coeurjolly and Levallois [2015]. 31

4.7 The signed heat method is 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. 32

4.6 Since our formulation is purely intrinsic, we can trivially improve accuracy and robustness by invoking *intrinsic Delaunay refinement* [Gillespie et al. 2021], without changing the implementation. 32

4.8 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. 33

4.9 *Left:* One can compute signed distance to broken curves that arise from attempting to draw curves on surfaces of high genus, with overhangs, and with holes and scanner noise. (Scanned bench from Choi et al. [2016].) *Right:* 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. 33

4.10 We can mix and match signed and unsigned distance. 34

4.11 The unsigned heat method exhibits bias near the domain boundary (figure reproduced from Crane et al. [2013b], Figure 11). Using their proposed boundary condition heuristic improves results at the cost of solving an additional linear system. In contrast, our method has correct boundary conditions without special treatment. 34

4.12 We can fit a signed distance function to several partial level sets. *Left*: Without constraints, isolines deviate slightly from source geometry. *Center*: 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. 35

4.13 *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, using either ADMM or PDHG. Here we start with a large diffusion time ($t = 100h^2$) to visually emphasize the effect. 36

4.14 Distribution of error in distance approximation for a perfect, unbroken curve on a high- and low-quality planar mesh *(top/bottom)*. Both meshes have about 100k faces. Inset numbers on SDF and error plots indicate compute time and mean error (*resp.*). 36

4.16 The signed heat 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. 37

4.15 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. 37

4.17 The tet mesh discretization of the volumetric domain yields more faithful reconstruction than using a grid with a similar number of DOFs, since its DOFs are adapted to the input. A tet mesh can also be made to be constrained to the input shape boundary, enabling exact enforcement of zero set constraints. 38

5.1 In $\mathbb{R}^3$, SDFs can be approximated by blending the SDFs of oriented tori. Tori have simple closed-form SDF expressions, and can capture locally spherical, ellipsoidal, and saddle-shaped surfaces, as well as cylindrical and planar surfaces as limiting behavior. 40

5.2 Madan and Levin [2022] suggest estimating unsigned distance to a point cloud P using $d(x) = -1/\lambda \log\left(\sum_{i=1}^{|P|} w_i(x) \exp\left(-\lambda \|x - p_i\|\right)\right)$ with $w_i = 1$, an instance of Equation 3.15. However, it is difficult to choose a single λ that yields a good reconstruction, and easy to accidentally merge two features. We experimented with using area weights, but could not tune this method effectively. Note that spatially varying λ compromises eikonality in LogSumExp-type methods. 41

5.3 Our method allows directly visualizing surfaces underlying point clouds through sphere tracing. Here we sphere-trace level sets in a shader; surface normals are simply given by the gradient of Equation 5.1. 41

5.4 *Left:* The ith torus's major and minor radii are aligned with the principal curvatures and directions of the estimated polynomial surface at point p_i, and its equator passes through the shifted point $Q_i^*(0,0)$. *Right:* The sign of each torus is determined by whether the torus osculates the estimated surface from the shape's interior or exterior. 42

5.5 In 2D, our method would use circles to approximate a 1D curve (*left*); in 3D, we use tori to approximate 2D surfaces, since tori generalize circles to two directions of curvature (*right*). 42

5.6 A neural network applied to each size-k neighborhood, using a series of transformer blocks that learn which points in each neighborhood are important for locally fitting a torus. 43

5.7 *Left:* Boolean operations are applied to two point clouds. *Right:* Morphological operations can be applied directly to the surfaces underlying point clouds, without meshing. 44

5.9 Some reconstructions produced by PAT of dense point cloud outputs from COLMAP, without any point cloud preprocessing. Our method yields reasonable reconstructions despite not being trained on any data with noise or outliers. . . 45

5.8 Offset surfaces to point clouds, including to an unevenly sampled point cloud (*bottom*). 45

5.10 General implicit functions can be turned into signed distance functions by applying our method to a sampling of their zero level set. Here we use the method of Sharp and Jacobson [2022] to sample neural implicits by casting rays. Query times are measured without GPU acceleration. 46

5.11 These 3D Gaussians have inconsistent geometry and orientations; nevertheless, our SDF behaves well in the far-field. Query times are measured without GPU acceleration. The results of SPSR [Kazhdan and Hoppe 2013] are shown for comparison (green). 46

5.12 On average, PAT has better signed distance accuracy than other convolutional formulas, and even a global method (SHM) for sparse point cloud data. It is also less prone to catastrophic error (Figure 5.13) 47

5.13 PAT tends to give better results on sparse point clouds, probably because our network was trained on relatively sparse point clouds (2048 points); here we use point clouds with 512 points. On some shapes, we observed better results using smaller λ than provided by the simple heuristic presented in Section 5.1.3, suggesting further tuning may be possible. NN-VIPSS can yield higher-quality reconstructions, especially when using ground-truth normals, though at a higher cost (Figure 5.14) 48

5.14 PAT is almost as fast as naive convolutional methods, and orders of magnitude faster than the signed heat method (SHM) while providing good signed distance quality on our datasets to sparse point clouds. For SPSR and NN-VIPSS, "precomputation" refers to reconstruction and meshing, while "evaluation" refers to distance evaluation to the meshed surface. Times correspond to sequential queries. 48

5.15 Precompute and query times for point clouds of size 110k, 880k, 7.3M, and 29M. Even for point clouds reaching tens of millions, each query of PAT's SDF remains on the order of milliseconds. 49

5.16 Though we use a simple, fixed heuristic, further tuning λ can lead to better reconstruction even using the same fitted tori. For example, if noise is high, setting λ lower leads to smoother reconstructions. Alternatively, if noise is low, setting λ higher might yield better detail recovery. 49

5.17 Classic point set methods often use a weighted least squares approach that solves for a best-fit quadratic polynomial around each point p_i, using Gaussian weights $\exp\left(-\|p_i - p_j\|^2/\sigma^2\right)$ for each neighbor p_j. In this example, we additionally fit tori to the fitted polynomials to get signed distance. Without further tuning, this process is incredibly brittle, with no values of σ yielding accurate results. 52

6.1 On surfaces, contouring the solution to Equation 6.2, equivalent to generalized winding numbers (GWN) and unregularized Poisson surface reconstruction (PSR), can yield regions that do not follow the input curves, and/or jump across nonbounding curves. 58

6.2 A collection of loops can be decomposed into bounding and nonbounding components in many different ways. We look for the decomposition whose residual is shortest (*middle*). 61

6.3 *Left:* Generalized winding number (solid angle) in $\mathbb{R}^2$. *Center, left:* The solution u of Equation 6.2. *Center, right:* Using vector diffusion, the boundary behavior more closely matches the free-space solution. *Right:* Alternatively, one can simply move the curve farther away from the domain boundary, though this is not always easily possible. 62

6.4 Some examples demonstrating the utility and robustness of SWN. 63

6.5 Even on meshes with low element quality, SWN can produce reasonable region labels (*center*). Since our formulation is intrinsic, any remaining artifacts can be eliminated via *intrinsic Delaunay refinement* (*right*). 63

6.6 As Γ becomes less broken, w approaches the expected winding number function, and the coefficients on nonbounding loops g approach 1. 63

6.8 Error rates for SWN (*left*) compared to naive rounding of u (*right*). The two example pairs show how naive rounding can fail to filter out nonbounding loops (in red). 64

6.9 On topologically trivial surfaces, SWN involves only a single sparse linear solve (*left*), and on surfaces with nontrivial topology solves an additional linear program (*right*). See the discussion below for a faster alternative. 64

6.7 Four of the 934 test cases in our synthetic benchmark. Each model is assigned ground truth region labels (indicated by colors), along with broken boundaries for those regions (black), and additional broken nonbounding loops (red). 64

6.10 The curves on this model represent five homologous loops; given their orientations, there are actually multiple possible valid curve decompositions. SWN always returns the one with the shortest collection of nonbounding curves (Figure 6.2). 64

6.11 A reduced-size linear program uses a path-finding heuristic to reduce the number of degrees of freedom from $|F|$ to just a few connected components; otherwise one follows the same procedure as the rest of the SWN algorithm. 65

6.12 The reduced-size linear program (*right*) is on average about 100 times faster than the original linear program (*left*) for topologically non-trivial surfaces, with comparable accuracy for region classification. (Compare with Figure 6.9). . . . 65

6.13 *Left:* The signed heat method (Chapter 4) produces extremely warped isolines when a curve fundamentally has no inside and outside. *Right:* Allowing piecewise continuous functions produces well-behaved, evenly-spaced isolines despite the fundamental sign ambiguity. 67

7.1 *Top:* Generalized winding number cannot be used for offset surfaces, since it provides only a smooth indicator function — not distance. *Bottom:* Generalized signed distance provides much nicer offsets on the same broken mesh. 69

7.2 For corrupted geometry, winding numbers are often worse than generalized signed distance (SHM) at classifying inside/outside. 70

7.3 *Left:* Winding number completes surfaces with saddle-shaped harmonic patches that exhibit poor normal continuity with the observed geometry (across many contour values). *Right:* PAT gives better reconstructions than generalized winding number or its regularized variant, especially for sparse data. 71

7.4 Winding numbers tend to cap off open curves, whereas the signed heat method tends to continue them. 71

7.5 We show the shape completion behavior of the three methods as a flat hole grows in size. *Top:* In 2D, winding numbers always connect curve endpoints with circular arcs (in 3D, more counter-intuitive saddle-shaped surfaces may emerge). *Middle:* The signed heat method considers normal information and hence has flatter completions, though still exhibits some bulging as the ratio of horizontal vs. vertical segments shrinks; the completed shape also exhibits a sudden change between the rightmost and second rightmost examples, when horizontal segments disappear entirely. *Bottom:* PAT works purely locally rather than solving a global reconstruction problem, and hence can exhibit discontinuity where neighboring fitted tori disagree. 72

7.6 SHM's diffusion time affects shape completion, interpolating between the linear prior used by the pseudonormal test (as $t \to 0$) and the harmonic prior used by generalized winding numbers (as $t \to \infty$). 72

7.7 Landscapes of signed distance error as point clouds of a cylinder become progressively worse under four different types of corruption, and varying cylinder aspect ratio. Colored circles match sets of conditions to visualizations of the corresponding contoured shape. 74

7.8 An analogous set of experiments to those in Figure 7.7, using a sphere instead of a cylinder. 75

7.9 The fast marching method is accurate and efficient when the input curve and surface are perfect, but otherwise can easily propagate sign and distance errors. 78

Chapter 1

Introduction

In many ways, our lives are defined by geometry — meaning shapes and their spatial relationships, and our study and manipulation of them. Examples of geometry and geometry-driven systems include both natural phenomena (material structure, plant growth, geological formation, etc.) and human-generated data (manufactured objects, art, digital assets). Geometry is the language in which we describe the physical world, both for ourselves and for the computer systems we work with. For example, every day, each of us probably interacts with hundreds if not thousands of objects, either physical or digital, that have been designed, simulated, or manufactured; fundamentally, these processes are enabled at scale by computer modeling and manipulation of 3D geometry. More broadly, computer algorithms for manipulating geometry drive technological progress across numerous domains including (just to name a few) physics simulation, robotics, medical imaging, animation, and design-based industries.

Unsurprisingly, problems involving geometry permeate the sciences and engineering. Naturally, problems involving geometry can be hard to solve, though not necessarily because the problem itself is difficult to model or understand, but simply because of the difficulty in doing computation with geometry. In particular, computers work with particular representations of geometry: for example, a smooth surface is often approximated by a collection of polygons (a "mesh") or even just a point set ("point cloud"). These instances of *digital geometry* very often suffer from quality issues, due to imperfect acquisition, reconstruction, or modeling: for instance, point clouds produced by 3D scanners have missing data, modelers create self-intersecting surfaces or invert elements (either intentionally or not), and so on. When this happens, even basic operations like simple inside/outside queries suddenly become ambiguous and undefined.

Unfortunately, these low-level defects like missing data, noise, self-intersections, and non-manifold features are unpredictable and easily frustrate higher-level design and optimization tasks such as toolpath planning, reliable physics analysis, or machine learning pipelines that demand large quantities of perfect, clean data. Explicit repair of data is possible but time-consuming and tedious, with no guarantees of perfection; we instead need algorithms that remain stable under perturbations in their input and hence allow direct computation of geometric quantities from defective data, which requires answering fundamental questions about geometry. In other words, we need *robust geometry processing*: we need algorithms that work reliably

across varying degrees of quality in their input.

This thesis specifically addresses two fundamental geometry problems: determining the inside and outside of a shape, and computing shortest distances to a shape, given an imperfect representation of the shape. We describe our methods as producing "generalized" inside/outside and signed distance, to distinguish our work from other types of robust algorithms (*e.g.* methods tuned for high numerical precision, such as exact predicates). Specifically, our algorithms generalize from imperfect observations, such as point sets and broken meshes, to answer queries about the underlying curve or surface. The thesis is organized as follows:

- Chapter 2 gives a self-contained overview of the mathematical background needed to understand our treatment of inside/outside and (signed) distance functions, and also defines the notation used throughout this document.
- Chapter 3 establishes theory underpinning inside/outside and signed distance computation, draws theoretical connections between the two, and describes fundamental tradeoffs to be made by algorithms for generalized signed distance.
- Chapter 4 describes an algorithm for computing generalized signed distance to defective curves and surfaces in 2D and 3D, whose generalization ability can be explained by the insights in Chapter 3.
- Chapter 5 extends generalized signed distance computation to fast, pointwise evaluation without spatial discretization, with particular focus on the especially difficult case of point clouds.
- Chapter 6 addresses the ambiguities of what it even means to be "inside" or "outside" a curve or surface in the first place, and uses *winding numbers* to compute inside/outside on general surface domains of arbitrary topology.

Finally, Chapter 7 gives practical comparisons between winding numbers- vs. signed distance-based shape completion as well as suggestions for practitioners, and Chapter 8 ends with some open questions. The content in this thesis largely draws from the following published papers:

- Nicole Feng, Mark Gillespie, and Keenan Crane. "Winding Numbers on Discrete Surfaces". *ACM Transactions on Graphics* 42.4 (2023)
  Project page with code: nzfeng.github.io/research/WNoDS/index.html
- Nicole Feng and Keenan Crane. "A Heat Method for Generalized Signed Distance". *ACM Transactions on Graphics* 43.4 (2024)
  Project page with code: nzfeng.github.io/research/SignedHeatMethod/index.html
- Nicole Feng, Ioannis Gkioulekas[†], and Keenan Crane[†]. "Points as Tori: Fast Pointwise Signed Distance for Point Clouds". *ACM Transactions on Graphics* 45.4 (2026)
  Project page with code: nzfeng.github.io/research/PointsAsTori/index.html

Chapters 3, 4, 5, and 6 inevitably contain descriptions similar to those in the above papers, though they generally contain livelier explanations than what can be afforded by a journal paper, as well as previously unpublished commentary and extensions. The other chapters contain some new results and write-ups.

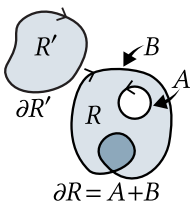
Chapter 2

Background

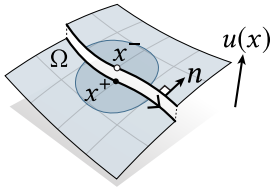
This chapter presents mathematical background for understanding inside/outside classification and signed distance computation, the two fundamental geometric quantities that form the core of this thesis.

2.1 Notation

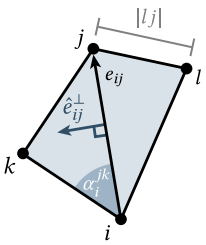
Throughout, we use $|\cdot|$ and $\langle \cdot, \cdot \rangle$ to denote the standard Euclidean norm and inner product for vectors in $\mathbb{R}^2$ and $\mathbb{R}^3$. For any function $f(t)$ of a single parameter t, we let $\dot{f}(t) := \frac{d}{dt} f(t)$. We use Δ to denote the negative-semidefinite *Laplace-Beltrami* operator on M, which locally behaves like the ordinary Laplace operator $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$. A function $u : M \to \mathbb{R}$ is *harmonic* if it is in the kernel of the Laplacian, *i.e.*, if $\Delta u = 0$.



Throughout we use Ω to denote a curve or surface, though we also use Γ to specifically denote a 1D curve. We also frequently discuss the boundaries of regions. Intuitively, the boundary ∂R of a 3D region R is the oriented surface enclosing R, and may have multiple components. Similarly, the boundary of a surface is the oriented curve enclosing the surface, and the boundary of a curve is its set of oriented endpoints. We call Ω *closed* if $\partial \Omega = \varnothing$.



One-sided limits. We make frequent use of signed functions whose value depends on from which side we approach a given curve or surface Ω. We denote one-sided limits of such functions as follows. On $\mathbb{R}^d$, we let $x^{\pm} := \lim_{s \to 0} x \pm s n(x)$ denote the point $x \in \Omega$ as approached from the positive or negative side of Ω. The vector $n(x)$ is a canonical choice of outward-pointing normal of $\Omega \subset \mathbb{R}^d$ at x that points from the negative to positive side of Ω (see inset). We then denote the corresponding one-sided limits of a function $u(x)$ as $u^{\pm}(x) := u\left(x^{\pm}\right) := \lim_{s \to 0} u\left(x \pm s n(x)\right)$. For functions defined on smooth manifolds (Section 2.2), an analogous definition holds where the limits are instead taken under an appropriate chart that maps from the surface to $\mathbb{R}^2$.



Triangle meshes. We often represent a 2D surface domain in the discrete setting as a triangle mesh $M = (V, E, F)$, with no restrictions on connectivity; we use C for the set of all triangle corners, and ∂M to denote the boundary of M. 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 ${}_i^{jk} \in C$. These indices are also used to express quantities stored on mesh elements — for instance, corner angles are denoted by α_i^{jk}. We use $|\cdot|$ to denote the volume of a simplex — for example, $|ij|$ is an edge length and $|ijk|$ is a triangle area. For brevity, we often assume that any interior, manifold, oriented edge ij is contained in two triangles labeled ijk, jil, where k and l sit to the left and right of ij, *resp.*

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. 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 $e_{ij}^{\perp}$ be the 90° rotation of e_{ij} in the counter-clockwise direction, and use $\hat{e}_{ij}, \hat{e}_{ij}^{\perp}$ for the corresponding unit vectors.

2.2 Just the basics of differential geometry

This section establishes the differential geometric foundations necessary for understanding the geometry processing algorithms developed in subsequent chapters. We focus on the key concepts of smooth manifolds and differential forms, which enable our function-based approach to geometry processing. Here we give only a high-level overview of the concepts needed to understand the algorithms in this thesis; for thorough treatment of these concepts, see Lee [2012] or do Carmo [1992]. A great introduction to differential forms includes Dray [2014, Section III].

Manifolds. Manifolds are topological spaces that locally look like $\mathbb{R}^n$. In particular, a *topological manifold* M has enough topological structure to allow uniqueness of limits and a meaningful notion of function continuity; M is also "locally Euclidean" in the sense that each point $p \in M$ lies in an open subset $V \subset M$ homeomorphic to an open subset of $U \subset \mathbb{R}^n$, *i.e.* there exists a continuous bijection $\varphi : U \to V$ with continuous inverse.

These homeomorphisms are called *coordinate charts* and let one locally analyze manifolds by comparing pieces of M with pieces of the more well-understood space $\mathbb{R}^n$. However, not all concepts from ordinary analysis can be consistently extended to topological manifolds this way. For example, one might think to call a function $f : M \to \mathbb{R}$ differentiable at $p \in M$ if its image under a coordinate chart is differentiable in $\mathbb{R}^n$. However, for purely topological manifolds, this notion of differentiability depends on the particular choice of coordinate chart. Thus we consider *smooth manifolds*, topological manifolds that can be covered with a collection of charts such that any two charts are *smoothly compatible*: that is, for any two charts $\varphi_\alpha : U_\alpha \subset \mathbb{R}^n \to M, \varphi_\beta : U_\beta \subset \mathbb{R}^n \to M$ such that $W := \varphi_\alpha(U_\alpha) \cap \varphi_\beta(U_\beta) \neq \varnothing$, the transition map $\varphi_\beta^{-1} \circ \varphi_\alpha : \varphi_\alpha^{-1}(W) \to \varphi_\beta^{-1}(W)$ is a smooth diffeomorphism. This compatibility ensures that the notion of differentiability is

well-defined, in the sense that the differentiability of functions does not depend on the choice of coordinate chart, so calculus can be done consistently on the manifold.

Tangent spaces and vector fields. Intuitively, *tangent vectors* are vectors that "lie flat" along a smooth manifold M (though they can be defined intrinsically). In more detail, each point $p \in M$ can be associated with a *tangent space* T_pM, an n-dimensional vector space isomorphic to $\mathbb{R}^n$ that captures all possible directions of motion through p. Tangent vectors can be characterized, for example, as equivalence classes of smooth curves through p, where two curves γ_1, γ_2 are equivalent if they have the same tangent vectors at a given point when pulled back to pieces of the plane. Because the smooth structure of M means derivatives of curves on M are well-defined, we can define tangent vectors on M.

The *tangent bundle* TM of M takes a union of all tangent spaces on M, and has a smooth structure inherited from the smooth structure of M. A *smooth vector field* can then be defined as a *section* of TM, meaning a smooth map $X : M \to TM$, where X assigns to each point $p \in M$ a tangent vector $X(p) \in T_pM$ such that the assignment varies smoothly.

Differential forms. Differential forms provide a framework for integration on manifolds, and are built up of objects called *covectors* that act as "length-ometers"[1] with which to measure the signed length of tangent vectors along certain directions.

In more detail, given an n-dimensional vector space V, a *covector* is a linear function $\omega : V \to \mathbb{R}$, and the space of all covectors forms a dual space V^* to V. Each point $p \in M$ of a smooth manifold can be associated with a *cotangent space* $T_p^*M := (T_pM)^*$ which consists of linear functionals $T_pM \to \mathbb{R}$, and the collection of all cotangent spaces on M is called the *cotangent bundle* T^*M; a smooth section of T^*M is called a smooth *covector field* or *1-form*. A 1-form ω can be used to integrate along a curve $\gamma : [a, b] \to M$ using $\int_\gamma \omega = \int_a^b \omega_{\gamma(t)}(\gamma'(t))\, \mathrm{d}t$.

More generally, one can define *(covariant) k-tensors*, multilinear functions that take in k vectors, and define *k-covectors* as *alternating* k-tensors; "alternating" means the k-tensor is negated when two arguments are swapped. Higher-degree k-covectors can be formed from lower-degree covectors using the *exterior product*, or *wedge product*. At each $p \in M$, one can consider the space $\Lambda^k(T_p^*M)$ of all k-tangent covectors, and consider a union of all such spaces to form the vector bundle $\Lambda^k(T^*M) = \coprod_{p \in M} \Lambda^k(T_p^*M)$; a section of $\Lambda^k(T^*M)$ is called a *(differential) k-form*, and the set of all k-forms on M is often denoted $\Omega^k(M)$. In words, a k-form, evaluated at $p \in M$, measures signed k-dimensional volumes in k-dimensional linear subspaces of T_pM. Given forms $\omega \in \Omega^k(M)$ and $\eta \in \Omega^\ell(M)$, their wedge product $\omega \wedge \eta \in \Omega^{k+\ell}(M)$ can be defined pointwise as $(\omega \wedge \eta)_p = \omega_p \wedge \eta_p$ for all $p \in M$.

The integration of differential forms on M also depends on the *orientability* of M: the manifold M is *orientable* if it has an atlas of coordinate charts whose transition maps all have positive Jacobian determinant. Classic examples of non-orientable manifolds include the Möbius strip and Klein bottle, where attempting to define a consistent orientation while traversing certain loops leads to contradictions.

[1] Phrase from Prof. Robin Neumayer at CMU.

Hodge duality. In $\mathbb{R}^n$, one can wedge together all basis 1-forms to form a unit n-form called a *volume form*. Given a volume form dV and a k-form ω, one can associate with it a $(n-k)$-form $*\omega$ such that their product is dV, *i.e.* $\omega \wedge *\omega = dV$. More generally, for two k-forms ω and η, one has $\omega \wedge *\eta = \langle \omega, \eta \rangle dV$, where $\langle \cdot, \cdot \rangle$ is the inner product on k-forms [Lee 2012, Ch. 14].

In this thesis, we often implicitly make use of Riemannian manifolds, so we can talk about geometric quantities like distances. A *Riemannian manifold* (M, g) is a smooth manifold equipped with a *Riemannian metric*, which is a smoothly varying choice of inner product for each tangent space: in terms of the above definitions, a Riemannian metric is a smooth section of the bundle $T^2(TM)$ of (covariant) 2-tensors on M that is positive-definite, symmetric, and bilinear. On Riemannian manifolds, the Riemannian volume form dV_g is the unique positively oriented n-form with unit norm with respect to the inner product $\langle \cdot, \cdot \rangle_g$ determined by g. The Hodge star operator can be defined as the unique homomorphism $\star : \Lambda^k(T^*M) \to \Lambda^{(n-k)}(T^*M)$ satisfying $\omega \wedge *\eta = \langle \omega, \eta \rangle_g dV_g$ for all k-forms ω, η [Lee 2012, Ch. 16].

Closed and exact forms. The *exterior derivative* d acts on k-forms to produce $(k+1)$-forms, and generalizes the differential of a function (a.k.a. a 0-form). For clarity, d acting on k-forms is sometimes denoted as d_k. On a smooth manifold M, the exterior derivative is the unique differential operator satisfying several fundamental properties, such as linearity over $\mathbb{R}$ and $d \circ d = 0$. The latter property, nilpotency, gives rise to the distinction between forms ω that are *closed* ($d\omega = 0$) and exact ($\omega = d\alpha$ for some $(k-1)$-form α), which is central to *de Rham cohomology* (Section 2.3.2).

The operator $\delta := *d*$, called the *codifferential*, is the adjoint of d with respect to the L^2 inner product, and acts on $(k+1)$-forms to produce k-forms. A $(k+1)$-form ω is called *coexact* if it can be expressed as the image of the codifferential (meaning there exists β such that $\omega = \delta\beta$), and *co-closed* if $\delta\omega = 0$. A k-form is *harmonic* if it is both closed and co-closed.

The fundamental relationship between differentiation and integration is captured by *Stokes' theorem*: for an oriented smooth n-manifold M with boundary and $(n-1)$-form ω with compact support, $\int_{\partial M} \omega = \int_M d\omega$.

Hodge decomposition. We make use of *Hodge decomposition* acting on 1-forms. In particular, any 1-form ω on a closed Riemannian manifold can be uniquely decomposed into a sum of an exact component $d\alpha$, coexact component $\delta\beta$, and a harmonic component γ,

$$\omega = d\alpha + \delta\beta + \gamma.$$

On manifolds with boundary, the *Hodge-Friedrichs-Morrey decomposition* applies [Schwarz 2006]. As detailed in Crane et al. [2013a, Chapter 8], this decomposition can be computed by solving a pair of Poisson equations

$$\Delta_0 \alpha = \delta_1 \omega \qquad \text{and} \qquad \Delta_2 \beta = d_1 \omega, \tag{2.1}$$

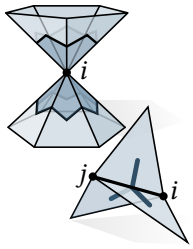
where $\Delta_0 := *_0^{-1} d_0^T *_1 d_0$ and $\Delta_2 := d_1 *_1^{-1} d_1^T *_2$ are the discrete 0- and 2-form Laplacians, *resp.*, with their usual zero-Neumann boundary conditions.

Hodge decomposition is a generalization of *Helmholtz decomposition* from vector calculus, and implies that solving for the scalar potential f that gives the best least-squares approximation of a given vector field X,

$$f \coloneqq \operatorname{argmin}_{f'} \|\nabla f' - X\|^2,$$

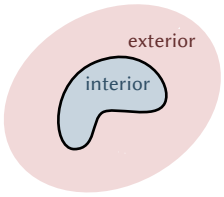
has a residual $\nabla f - X$ that consists of at most a coexact plus a harmonic component.

Discrete differential forms. On triangle meshes, we use *discrete exterior calculus* to represent differential forms on simplicial complexes [Desbrun et al. 2005; Crane et al. 2013a]. On triangle meshes, k-forms are represented as values assigned to k-dimensional mesh elements: 0-forms correspond to values on vertices, 1-forms on edges, and 2-forms on faces. The *discrete exterior derivative* d_k maps discrete k-forms to discrete $(k+1)$-forms, and can be represented as a sparse matrix equal to the transpose of the boundary operator ∂_{k+1} that maps $(k+1)$-dimensional complexes to k-dimensional ones. The *discrete Hodge star* $\star_k$ takes k-forms to $(n-k)$-forms, incorporating geometric information arising from edge lengths. These discrete operators preserve the fundamental structure of their smooth counterparts: for example, $d_{k+1} \circ d_k = 0$, and discrete versions of Stokes' theorem hold.



For general, possibly nonmanifold triangle meshes, we follow Sharp et al. [2019a] and define the discrete Hodge star operators by taking volume ratios involving all incident elements (see inset), yielding diagonal matrices with entries $(\star_0)_i \coloneqq \frac{1}{3}\sum_{ijk\in F} |ijk|$ for all $i \in V$, $(\star_1)_{ij} \coloneqq \mathrm{w}_{ij}$ for all $ij \in E$, where $\mathrm{w}_{ij} \coloneqq \frac{1}{2}\sum_{ijk\in F} \cot \alpha_k^{ij}$ are *cotan weights* [MacNeal 1949, Section 3.2], and $(\star_2)_{ijk} \coloneqq 1/|ijk|$ for all $ijk \in F$. Otherwise, we use the standard discrete exterior derivative matrices d_k; the *discrete codifferential* is then $\delta_k \coloneqq \star_{k-1}^{-1} d_{k-1}^T \star_k$.

2.3 The inside and outside of curves and surfaces

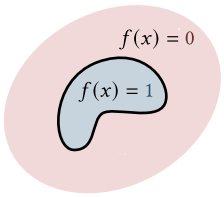


Intuitively, closed objects — such as a capped bottle, soccer ball, or sealed building — separate space into an "inside" region comprised of all the points which cannot be connected to the "outside" without intersecting the object. This intuition can be formalized, for instance using the *Jordan curve theorem* which says that every simple closed curve in the plane — that is, every non-self-intersecting continuous loop — decomposes $\mathbb{R}^2$ into two connected components: a region R bounded by the curve, called the "interior", and its unbounded complement $\mathbb{R}^2 \setminus R$, called the "exterior" of the curve. The *Jordan–Brouwer separation theorem* gives an analogous statement in higher dimensions.

Determining the interior (or equivalently, determining the exterior) of a curve or surface is a fundamental problem in computer vision and computer graphics. For example, *surface reconstruction* is a perennial task whose goal is to infer, or reconstruct, the surface from which a collection of discrete points or polygons were sampled; the output is a well-defined interior and exterior of a shape [Berger et al. 2014]. Other applications that depend on reliable interior/exterior

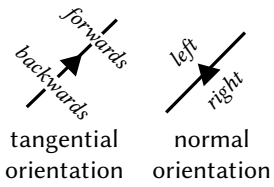
computation include motion planning, computer-aided design, and geographic information systems.

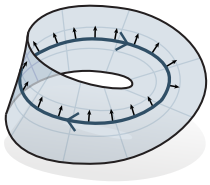
The interior and exterior of a shape are often represented using a scalar function, such as an *occupancy function*, which typically has value 1 inside the shape and 0 outside (inset). In Section 2.3.3 and Chapter 6, we consider the more general *winding number function*, which gives not just a binary inside/outside classification, but rather an integer-valued function that counts how many times a curve or surface encloses a given point.



2.3.1 Curve orientations

For a surface embedded in $\mathbb{R}^d$, we can specify its orientation as a choice of a continuously-varying surface normal $n(x)$ at every point. For curves, one can also specify orientation via normal direction, picking "left" and "right" sides of the curve. We can also orient along the curve's tangent direction, picking forward and backward directions (inset). For curves on an oriented surface, the two notions of orientation are equivalent: the normal of the curve, and hence the left/right sides, is usually defined via a 90° counterclockwise rotation of the tangent.





However, if M is non-orientable, then one cannot pick a consistent counterclockwise direction at all points of the surface, and the two types of orientation are not equivalent: for example, the inset shows a curve on a Möbius strip that cannot be given a consistent normal orientation, but can be given a consistent tangent orientation.

2.3.2 Homology & cohomology

On domains besides $\mathbb{R}^d$, a closed curve or surface may not necessarily be the boundary of a well-defined region (inset). The failure of a curve to be a boundary of a region can be studied using *homology* and *cohomology* (Munkres [1984, Chs. 1 & 5]).

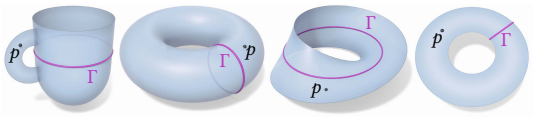
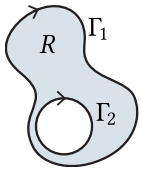
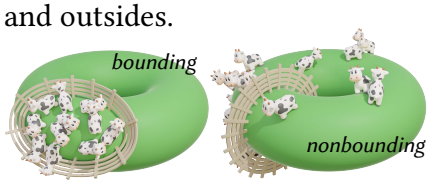


Fig. 2.1: Is the point p "inside" or "outside" the curve Γ? On surfaces, this question does not always have a meaningful answer.

Homology. On orientable surfaces without boundary, two closed curves Γ_1 and Γ_2 are *homologous* if they form the boundary of a region R, meaning if $\Gamma_1 - \Gamma_2 = \partial R$ (inset). A curve that is the boundary of a region is called *nullhomologous*. In the plane, all closed curves are nullhomologous and hence have well-defined insides and outsides.

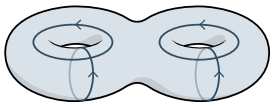




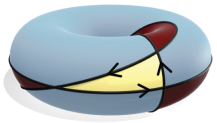
However, on surfaces or even in subsets of the plane, there may be closed curves which are not the boundary of any region; that is, these curves are not congruent to zero in

the homology group $H_1(M) = \ker(\partial_1)/\mathrm{Im}(\partial_2)$. For clarity, we call a closed curve *bounding* if it is nullhomologous, and *nonbounding* if it is non-nullhomologous (a.k.a. a *separating cycle*.)

The *homology class* of a curve Γ is the set of all curves homologous to Γ. A set of *homology generators* for M is a set of closed curves $\{\eta_i\}$ from which we can construct a curve in every homology class. On a closed surface of genus g there are $2g$ homology generators, which can be organized in pairs around each handle (inset).

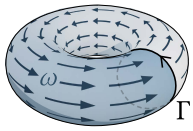


Even if individual loops do not bound regions, however, they can still conspire to define a meaningful partition — see the right inset, which reproduces an example from Riso et al. [2022, Figure 4].



Cohomology. Alternatively, we can study curves and their bounding properties by studying 1-forms dual to the curves. In *de Rham cohomology*, curves Γ are replaced by 1-forms ω, and the boundary operator is replaced by the *exterior derivative* d. A 1-form ω is *closed* if $d\omega = 0$. Two closed 1-forms ω_1, ω_2 are said to be *cohomologous* if $\omega_1 - \omega_2 = d\alpha$, mirroring the condition for curves. Whereas the *Poincaré lemma* states that all 1-forms in $\mathbb{R}^2$ are cohomologous to zero, there can be multiple cohomology classes on surfaces. A set of *cohomology generators* for M is a collection of 1-forms $\{\omega_i\}$ allowing us to represent the cohomology class of any closed 1-form ξ via a sum of generators. On a closed surface of genus g, there are $2g$ cohomology generators, matching the number of homology generators.

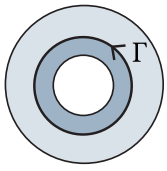
A 1-form ω can be paired with a curve Γ via integration, yielding a value $\int_\Gamma \omega$. Using this pairing, a 1-form ω is *dual* to a closed curve Γ if integration against ω counts intersections with Γ, *i.e.* if $\int_{\Gamma'} \omega$ is the signed number of intersections between Γ and Γ' for any closed Γ'.

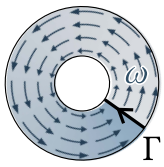


Cohomology is closely related to the theory of harmonic functions and differential forms. In particular, by the *Hodge theorem*, on compact, oriented Riemannian manifolds there is a unique harmonic form ("harmonic representative") in each cohomology class, allowing us to represent cohomology classes concretely using harmonic forms. We make use of this duality in Chapter 6.

Relative homology & cohomology. So far we have only discussed surfaces M with no boundary, in which case nonbounding (non-nullhomologous) curves are categorized by the *absolute homology group* $H_1(M)$ [Erickson and Whittlesey 2005].

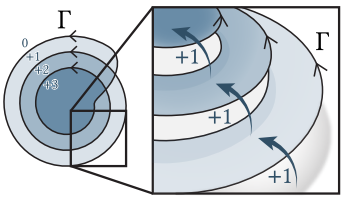
On surfaces with boundary, however, curves may be nonbounding in the sense that they separate the domain into two components, but cannot be represented as the boundary of some region in an absolute sense. For instance, the annulus has a single homology generator: a loop Γ wrapping around the hole in the middle. While Γ separates the annulus into two components, it is not the boundary of any region of the annulus since the boundary of *e.g.* the inside component includes the inner circle of the annulus' boundary in addition to Γ. On surfaces with boundary, nonseparating cycles are instead described by the *relative homology group* $H_1(M, \partial M)$, whose elements are closed loops in the space obtained by collapsing all of ∂M to a single point [Munkres 1984, §9].





On surfaces with boundary, one must also distinguish between absolute and relative cohomology. The absolute cohomology group $H^1(\Omega)$ consists of harmonic 1-forms tangent to the boundary, whereas the relative cohomology group $H^1(\Omega, \partial\Omega)$ consists of harmonic 1-forms normal to the boundary [Poelke and Polthier 2016]. The dual of a relative homology generator is an absolute cohomology generator and vice versa; for example, the dual of the nonseparating relative homology generator is the 1-form which circulates around the annulus, tangent to the boundary (inset).

2.3.3 Winding numbers



Winding numbers are a basic concept from differential geometry that give a natural extension of the binary notion of inside/outside to more than two regions [Do Carmo 2016, Section 5.7]. In the plane, winding number is a piecewise constant function that jumps by +1 as one crosses a curve Γ from the right (Figure 2.2, inset). The classic winding number is a special case of the signed *solid angle* function, which is itself a particular *harmonic function*, *i.e.*,

$$\text{winding number} \subset \text{solid angle} \subset \text{harmonic functions}.$$

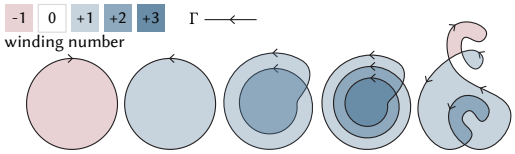
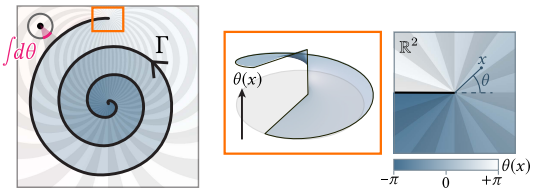


Fig. 2.2: For curves Γ in the plane, the winding number function $w_\Gamma(p)$ gives the number of times the curve Γ wraps around any given point p.

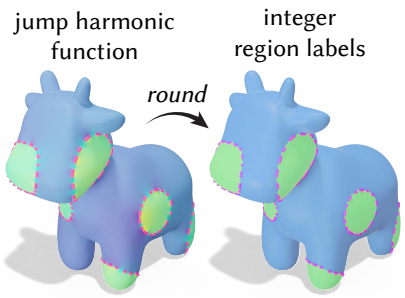
For broken curves, we must consider the more general solid angle function, which Jacobson et al. [2013] call the *generalized winding number (GWN)*. In turn, solid angle is not well-defined for curves on surfaces, leading us to consider the broader class of harmonic functions with jumps, or *jump harmonic functions* for short. For a non-self-intersecting curve Γ, a jump harmonic function u satisfies a Laplace equation with jump boundary conditions, which we call a *jump Laplace equation*:

$$\begin{aligned} \Delta u &= 0, && \text{on } M \setminus \Gamma, \\ u^+ - u^- &= 1, && \text{on } \Gamma, \\ \partial u^+/\partial n &= \partial u^-/\partial n, && \text{on } \Gamma. \end{aligned} \tag{2.2}$$



Harmonic functions continuous up to jumps also arise naturally in surface parameterization, for instance, as conjugate harmonic functions in conformal mapping [Gu and Yau 2003; Sawhney and Crane 2017]; our treatment of such functions is similar to Tong et al. [2006]. A jump harmonic function u is in fact harmonic everywhere, including at points on the curve, modulo the integer jumps; the angle-valued function u only fails to be harmonic at curve endpoints, where there is a branch-point singularity (inset) — see [Krutitskii 2001] for a careful treatment in the case $M = \mathbb{R}^2$.

If the domain M is simply-connected, meaning there cannot exist nonbounding loops on M, then we can simply solve Equation 2.2 for the function u. If Γ is closed, u is analogous to the ordinary winding number in the plane, which yields a piecewise constant, integer-valued function (inset). If Γ is not closed, but M is still simply-connected, then u yields a "soft" real-valued (rather than purely integer-valued) indicator function, analogous to solid angle.



History of winding numbers and solid angle. Connections between winding numbers, solid angles, and harmonic functions have long appeared in mathematics, physics, and scientific computing [Binysh and Alexander 2018]. Both Euler [1781] and Lagrange [1798] give formulas for the solid angle of a triangle; Gauss [1838, Sections 37-38] notes the relationship of solid angle to magnetic potential; Maxwell [1881, Articles 409-11, 417-21] further makes connections to jump conditions. Methods for approximating solid angles also play an integral role in *boundary element methods (BEM)* for the Laplace equation [zhi Ning et al. 2010].

In computer graphics, winding numbers were first applied to *point-in-polygon queries* [Shimrat 1962; Haines 1994], and continue to play a key role in general point-in-shape queries [Lengyel 2017; Spainhour et al. 2024; Martens and Bessmeltsev 2025; Spainhour and Weiss 2026]. Solid angle plays a key role in rendering, *e.g.*, for finite element radiosity [Goral et al. 1984] or importance sampling for direct illumination [Veach and Guibas 1995, Section 2.1]. In geometry processing, the utility of the solid angle function for broken geometry has been rediscovered twice, via both *Poisson surface reconstruction (PSR)* [Kazhdan et al. 2006] and the *generalized winding number (GWN)* [Jacobson et al. 2013]. These methods are in turn key components of a wide variety of applications [Hu et al. 2018; Zhou et al. 2016; Chi and Song 2021; Müller et al. 2021; Dvořák et al. 2022; Collet et al. 2015; Chang et al. 2017].

Owing to their rich history and prevalence in math and physics, winding numbers can be characterized through several different perspectives [Feng et al. 2023, suppl.]. These perspectives do not, however, have a standard extension to curves on surfaces, due to the possible presence of nonbounding loops (Section 2.3.2). Several authors have instead considered generalizations of *turning number* on surfaces, a quantity distinct from winding number despite the historical confusion of terminology [Reinhart 1960, 1963; Chillingworth 1972; Humphries and Johnson 1989; McIntyre and Cairns 1993]. McIntyre and Cairns [1993, Lemma 2] does describe a function that behaves like the winding number for bounding curves, but for nonbounding curves must introduce arbitrary discontinuities to keep this function piecewise constant, and Chernov and Rudyak [2009] define a so called *affine winding number* useful only for curves within a common homotopy class. More recently, Riso et al. [2022] give a method for computing winding numbers but only on perfectly closed curves already partitioned into distinct loops. In Chapter 6, we give an algorithm that gives a well-behaved generalization of winding number, given minimal structure on the input curves.

2.4 Geodesic distance

Distance computation on curved spaces requires additional geometric structure beyond the purely topological notion of manifolds. We consider an n-dimensional *Riemannian manifold* (M, g) with metric g, which for each $p \in M$ gives rise to the norm of tangent vectors $v \in T_pM$ through $\|v\|_g := g_p(v, v)^{1/2}$, and angle between two tangent vectors $v, w \in T_pM$ through $\cos\theta = \frac{g_p(v,w)}{\|v\|_g \|w\|_g}$.

The metric lets us measure lengths of curves: for a (piecewise) smooth curve $\gamma : [a, b] \to M$, its length is

$$L_g(\gamma) := \int_a^b \|\gamma'(t)\|_g \, dt$$

(This definition is independent of parameterization.) The *Riemannian distance* between points $p, q \in M$ is then defined as the infimum of lengths over all curves connecting p and q. (Note that "distance" is also almost always used to refer to minimum distance, a convention we will also adopt throughout this thesis unless otherwise noted.) Curves that are locally length-minimizing are called *geodesics*. In $\mathbb{R}^n$, geodesics are straight lines; geodesics generalize straight lines to arbitrary Riemannian manifolds.

Parallel transport. Geodesics can also be defined to be curves whose tangent vectors remain parallel to the curve as they are transported along the curve. To define what it means for vectors to "remain parallel" across different tangent spaces, we need the concept of a *connection*, which provides a way to differentiate vector fields along curves. The *covariant derivative* generalizes the notion of a directional derivative to curved manifolds, and an *affine connection* ∇ on M assigns to a pair of vector fields X, Y a new vector field $\nabla_X Y$ that gives the covariant derivative of Y in the direction X. The fundamental theorem of Riemannian geometry states that given a Riemannian metric g, there exists unique affine connection on M, the *Levi-Civita connection*, that is both *metric-compatible* ($\nabla g = 0$) and *torsion-free*.

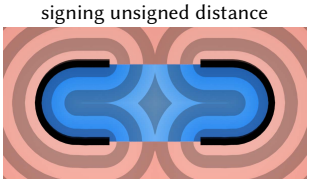
The connection enables definition of *parallel transport* along curves: given a curve $\gamma : I \to M$ and initial vector $V_0 \in T_{\gamma(t_0)}M$, there exists a unique vector field V along γ such that $V(t_0) = V_0$ and $\nabla_{\dot\gamma} V = 0$. Parallel transport of V along γ maps tangent vectors in $T_{\gamma(t_0)}M$ to tangent vectors in $T_{\gamma(t_1)}M$. A geodesic is a curve γ whose velocity vector field γ' is parallel-transported along itself, meaning $\nabla_{\dot\gamma}\dot\gamma = 0$.

Signed distance. Consider a codimension-1 submanifold $\Omega \subset M$ (for example, curves on a surface, or surfaces within a volume). If Ω separates M into an interior A and an exterior $M \setminus A$, we can define a *signed distance function (SDF)* $\phi : M \to \mathbb{R}$ as

$$\phi(x) = \begin{cases} d_g(x, \Omega) & x \in M \setminus A \\ -d_g(x, \Omega) & x \in A \end{cases}$$

where the sign of $\phi(x)$ indicates whether x is in the interior or exterior of Ω. SDFs hence encode both geometric information (how far x is from Ω) and topological information (on which side of Ω lies x) in a single scalar function.

Signed distance is essential to many problems across graphics and vision, forming 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. 2024] and computer vision [Vicini et al. 2022]. For watertight geometry in $\mathbb{R}^n$, there are many ways to compute signed distance: for example, unsigned distance can first be computed via fast, exact closest point queries [Sawhney 2021], then 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]; a few methods consider unsigned distance to curves embedded in surfaces [Bommes and Kobbelt 2007; Trettner et al. 2021], which can then be signed using, *e.g.*, flood fill.



For non-watertight, noisy, self-intersecting, or otherwise broken geometry, the operations of computing distance and signing no longer commute. For one, simply signing unsigned distance can yield a function quite different from the SDF to completed geometry (inset). Likewise, wavefront-based methods like *fast marching* [Kimmel and Sethian 1998] and learning-based variants [Lichtenstein et al. 2019; Huberman et al. 2023] propagate sign errors. Some past works consider regularized signed distance for broken geometry, though they suffer from various downsides that affect their accuracy, robustness, or generality [Bærentzen 2005; Mullen et al. 2010; Calakli and Taubin 2011; Xu and Barbič 2014; Brunton and Rmaileh 2021]. In Chapter 4, we present an algorithm for signed distance that is robust to broken geometry, and generalizes to curved surfaces and alternative spatial discretizations. This notion of "generalized signed distance" (sometimes called "signed distance reconstruction" [Hubert-Brierre et al. 2025]) directly approximates signed distance to a well-behaved reconstruction of the input geometry — that is, to an approximation of the true geometry underlying a possibly corrupted observation.

Chapter 3

Theoretical foundations of generalized signed distance

Reconstruction of curves and surfaces, and signed distance computation, represent two fundamental problems in geometry processing. Of course, the two problems are highly related, because a meaningful notion of signed distance depends on a meaningful notion of inside and outside. This thesis in fact develops two algorithms for computing generalized signed distance that work by essentially achieving simultaneous reconstruction and distance.

The connection between reconstruction and distance computation can be formalized: in Section 3.1, we show that winding numbers-based reconstruction and Poisson surface reconstruction are related to signed distance via a relatively simple change of variables. This relationship gives rise to *convolutional distance approximations*: a class of algorithms that approximate the minimum distance to a given shape through a summation of kernels concentrated on the shape's boundary. Such approximations for unsigned distance have been re-discovered several times throughout image processing, signal processing, computer vision, and computer graphics. Signed variants can also be derived, and through these signed variants we formalize the connection between winding numbers and signed distance. At a high level, these (un)signed distance approximations may be obtained as viscosity solutions to eikonal equations, or through asymptotic analysis using Laplace's method, encompassing the well-known Hopf-Cole transformation for PDEs, Varadhan formulas for geodesic distance, and LogSumExp methods and softmax functions used in machine learning.

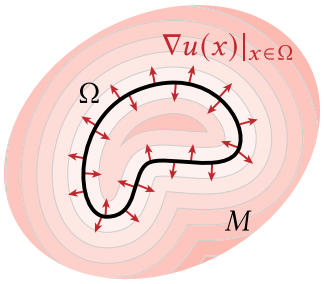
However, while such convolutional distance approximations produce good results for densely sampled boundaries such as polygon meshes and image contours, we show that no convolutional approximation can yield good results on sparsely sampled boundary data consisting of isolated point samples, such as point clouds. Instead, it is necessary to decouple regression from distance computation to some extent, resulting in the multi-step but especially robust *signed heat method* described in Chapter 4. We further extend generalized signed distance to fast, pointwise evaluation in Chapter 5. This section also discusses prior work on distance computation, reconstruction, and kernel methods.

3.1 A unification of winding numbers, Poisson surface reconstruction, and signed distance

In this section, we describe the connection between winding numbers, Poisson surface reconstruction, and signed distance.

We start by considering the following eikonal equation for unsigned distance:

$$\begin{aligned} \|\nabla u\|^2 &= 1 & x \notin \Omega \\ u(x) &= 0 & x \in \Omega \\ \tfrac{\partial u^{\pm}}{\partial n}(x) &= \pm 1 & x \in \Omega. \end{aligned} \tag{3.1}$$



This equation describes an unsigned distance function $u(x)$ to Ω in some domain M (inset). The two-sided boundary conditions consist of Dirichlet boundary conditions that state the distance should be zero at the source geometry Ω, and double-sided Neumann conditions that state $u(x)$ should be increasing on either side away from Ω.

Distance functions, however, are not differentiable at the cut locus of the source geometry Ω, the locus of points at which the minimizer of distance to Ω is non-unique. For x on the cut locus, $\nabla u(x)$ is hence undefined and the meaning of Equation 3.1 is unclear. Distance functions instead satisfy Equation 3.1 in a "viscosity sense": whereas a true distance function is not everywhere differentiable, it can be obtained as the limit of the solution to a perturbed version of the eikonal equation as the amount of perturbation goes to zero. In particular, one can perturb Equation 3.1 by adding a small amount of scalar diffusion, obtaining a "viscous" eikonal equation

$$\begin{aligned} \|\nabla u\|^2 - 1 &= \tfrac{1}{\lambda}\Delta u(x) & x \notin \Omega \\ u(x) &= 0 & x \in \Omega \\ \tfrac{\partial u^{\pm}}{\partial n}(x) &= \pm 1 & x \in \Omega. \end{aligned} \tag{3.2}$$

The seminal work by Crandall and Lions [1983] introduced *viscosity solutions* as a type of generalized solution to Hamilton-Jacobi equations that satisfies uniqueness.

Equation 3.2 is an example of a time-independent, viscous *Burgers' equation*, for which there is a well-known change of variables called the *Hopf-Cole transformation* (sometimes called *Cole-Hopf transformation*) [Hopf 1950; Cole 1951]. Evans [1998, Section 4.4] gives a derivation of the transformation, which through exponentiation turns a nonlinear time-dependent viscous Burgers' equation into linear heat equation. In our context, the transformation

$$w(x) = \exp(-\lambda u(x)) \tag{3.3}$$

turns the nonlinear, viscous eikonal equation in Equation 3.2 into a linear screened Laplace equation, where the viscosity now acts as a screening term that controls the amount of damping on a diffusive process [Belyaev and Fayolle 2015]:

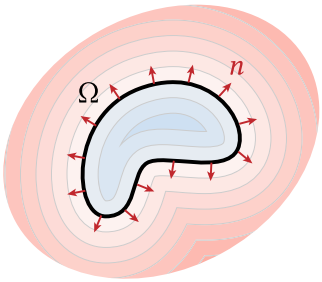
$$\begin{aligned} \Delta w(x) - \lambda^2 w(x) &= 0 & x \notin \Omega \\ w(x) &= 1 & x \in \Omega \\ \tfrac{\partial w^{+}}{\partial n}(x) &= -\tfrac{\partial w^{-}}{\partial n}(x) & x \in \Omega. \end{aligned} \tag{3.4}$$

The Hopf-Cole transformations, as applied to the heat equation and the screened Laplace equation in Equation 3.4, correspond to the two formulas for geodesic distance presented by Varadhan [1967].

We now consider an eikonal equation which solves for signed rather than unsigned distance:

$$\begin{aligned} \|\nabla u(x)\|^2 &= 1 \quad x \notin \Omega \\ u(x) &= 0 \quad x \in \Omega \\ \tfrac{\partial u}{\partial n}(x) &= 1 \quad x \in \Omega. \end{aligned} \tag{3.5}$$

Compared to Equation 3.1 for unsigned distance, Equation 3.5 has Neumann conditions that are continuous across Ω (inset).



We can likewise consider a signed eikonal equation with viscosity,

$$\begin{aligned} \operatorname{sign}_\Omega(x)\left(\|\nabla u(x)\|^2 - 1\right) &= \tfrac{1}{\lambda}\Delta u(x) \quad x \notin \Omega \\ u(x) &= 0 \quad x \in \Omega \\ \tfrac{\partial u}{\partial n}(x) &= 1 \quad x \in \Omega, \end{aligned} \tag{3.6}$$

and a *signed* variant of the Hopf-Cole transformation [Lipman 2021]

$$w(x) = \operatorname{sign}_w(x) \exp\left(-\lambda \operatorname{sign}_w(x) u(x)\right). \tag{3.7}$$

Applying Equation 3.7 to Equation 3.6 yields a *jump* screened Laplace equation:

$$\begin{aligned} \Delta w(x) - \lambda^2 w(x) &= 0 \quad x \notin \Omega \\ w^{\pm}(x) &= \pm 1 \quad x \in \Omega \\ \tfrac{\partial w^+}{\partial n}(x) &= \tfrac{\partial w^-}{\partial n}(x) \quad x \in \Omega. \end{aligned} \tag{3.8}$$

Compared to the screened Laplace equation in Equation 3.4, the jump screened Laplace equation in Equation 3.8 has boundary conditions that make the solution jump across the source geometry Ω. A derivation of the signed Hopf-Cole transformation, valid for both closed and open Ω, is given in Feng et al. [2026, App. A].

The relationship between distance functions and screened Laplace solutions via the Hopf-Cole transformation can be described intuitively as follows. The jump screened Laplace equation in Equation 3.8 diffuses double-sided boundary values from Ω, where larger values of λ means greater screening (damping) so that boundary data is diffused less and instead more concentrated around Ω. Diffusion yields a function whose decay is roughly exponential in distance, so applying the inverse signed Hopf-Cole transformation

$$u(x) = -\tfrac{1}{\lambda} \operatorname{sign}_w(x) \log|w(x)| \tag{3.9}$$

essentially reverses this exponential decay by taking a log, giving an approximation of distance as $\lambda \to \infty$ (Figure 3.1, *right*). The jump screened Laplace equation in Equation 3.8 can further be rewritten as the screened Poisson equation

$$\Delta w(x) - \lambda^2 w(x) = -2\left(\nabla \cdot n(x)\right) \mu_\Omega(x) \tag{3.10}$$

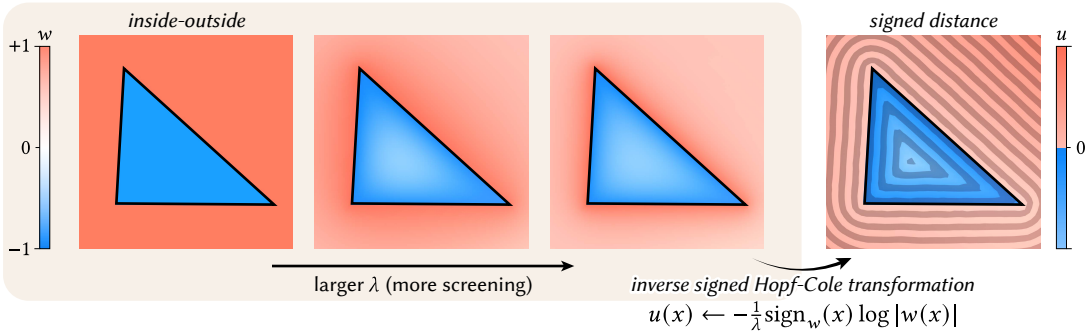


Fig. 3.1: As the screening amount λ goes to 0, the solution w to Equation 3.8 converges to a jump harmonic function describing inside-outside (*left*). As λ increases, w converges to a signed distance function via a log transform (*right*). However, this formula only works for closed geometry and fails for sampled geometry like point clouds (Section 3.3).

where $n(x)$ denotes the outward-pointing unit normals to Ω, and μ_Ω is a measure concentrated on Ω. A derivation of Equation 3.10 is given in Feng et al. [2026, App. B].

The free-space Green's function $G^\lambda(x, y)$ of the screened Laplace operator, called the *Yukawa potential* or *screened Coulomb potential*,

$$G^\lambda(x, y) := \frac{\exp\left(-\lambda\|x - y\|\right)}{4\pi\|x - y\|} \tag{3.11}$$

yields the normal derivative (a.k.a. *Yukawa double-layer potential*)

$$\nabla G^\lambda(x, y) \cdot n = \frac{(\lambda\|x - y\| + 1)\,\langle x - y, n\rangle \exp\left(-\lambda\|x - y\|\right)}{2\pi\|x - y\|^3} \tag{3.12}$$

which can be used to express the solution $w(x)$ of Equation 3.8 at all $x \notin \Omega$ as the boundary integral

$$w(x) = 2\int_\Omega \frac{\partial G^\lambda(x, z)}{\partial n(z)}\,\mathrm{d}z = \int_\Omega \frac{(\lambda\|x - z\| + 1)\,\langle x - z, n(z)\rangle}{2\pi\|x - z\|^3} \exp\left(-\lambda\|x - z\|\right)\,\mathrm{d}z, \quad x \notin \Omega. \tag{3.13}$$

Unlike the distance function u for which $|u(x)| \to \infty$ as $\|x\| \to \infty$, the solution w of Equation 3.8 exhibits exponential decay at infinity, and so can be represented by the boundary integral in Equation 3.13.

Fascinatingly, these derivations establish a close relationship between classic occupancy methods and signed distance. As the screening parameter $\lambda \to 0$, Equation 3.8 becomes a jump Laplace equation whose solutions — so-called *jump harmonic functions* (Section 2.3.3) — describe the generalized winding number (Figure 3.1, *left*)[1]. In turn, the generalized winding number is a special case of *Poisson surface reconstruction* [Kazhdan et al. 2006]: Poisson surface

[1]Note that screening that appears in Equations 3.4, 3.8, and 3.10 is a volumetric screening term distinct from the surface-based screening used in screened Poisson surface reconstruction [Kazhdan and Hoppe 2013].

reconstruction is equivalent to a regularized version of winding numbers, which corresponds to convolving the right-hand side of the Poisson equation in Equation 3.10 with a Gaussian [Chen et al. 2024a], and taking $\lambda \to 0$. In summary, we can obtain signed distance from occupancy methods simply by introducing a screening term into their partial differential equations (PDEs) — at least for perfect geometry.

The insights in this section are inspired by the viscous formulation of the signed eikonal equation in Lipman [2021], and Feng et al. [2026, App. A] derives the boundary conditions necessary for formalizing the connection between winding numbers, regularized winding numbers, and Poisson surface reconstruction.

Practical issues. We will soon see in Section 3.3 that Hopf-Cole-based distance approximations are fundamentally flawed when acting on sampled data, such as point clouds, severely limiting their direct applicability to real-world data. But let's assume we have perfect data — are there still challenges?

On discrete surfaces, Crane et al. [2013b] actually advocate against approximating geodesic distance by solving a screened Laplace or heat equation then applying a Hopf-Cole transformation. In particular, for the distance approximation to be accurate, one needs to take the diffusion time $t \to 0$, but on discrete surfaces one cannot take t too small else the approximation converges to the graph metric [Crane et al. 2013b, Appendix A]. On $\mathbb{R}^n$, however, we can use the boundary integral in Equation 3.13: this boundary integral uses not a discrete Laplacian, but Green's function for the screened Laplace operator without discretizing $\mathbb{R}^n$, and hence we can take $t = \lambda^{-2}$ much smaller (barring precision issues).

A more significant drawback of applying Hopf-Cole transformations/Varadhan's formulas is that they can be numerically unstable. First, the distance approximation can be inaccurate near Ω where $|w| \leq 1$, and $\log |w|$ hence has a very large derivative – so any existing errors near Ω become exacerbated after applying a log transform. These errors may arise, for example, due to the Yukawa potential's singular behavior at Ω. Second, one quickly runs into numerical precision issues far away from Ω due to the exponential decay of the Yukawa potential and heat kernel, though a simple shifting of the exponents buys precision (see Equation 5.8).

3.2 Convolutional distance approximations

For perfect geometry, signed distance can be obtained by convolving an exponential kernel over the source geometry Ω (Equation 3.13), then applying an appropriate log transformation (Equation 3.9). The kernel used for convolution, however, does not have to be the Yukawa double-layer potential in Equation 3.12. In fact, for *any* exponential kernel with parameter λ inside the exponential, and for any continuous function $h : \mathbb{R}^d \to \mathbb{R}$ and twice-differentiable function $\varphi : \mathbb{R}^d \to \mathbb{R}$, we have the asymptotic behavior

$$\int_\Omega h(z) \exp\left(-\lambda \varphi(z)\right) \, \mathrm{d}z \overset{\lambda \to +\infty}{\sim} (2\pi/\lambda)^{d/2} \det\left(\nabla^2 \varphi(x^*)\right)^{-1/2} h(x^*) \exp\left(-\lambda \varphi(x^*)\right) \tag{3.14}$$

where $x^* := \operatorname{argmin}_{z\in\Omega} \varphi(z)$ is the minimizer of the exponential argument φ, assumed to be unique [Belyaev and Fayolle 2024; Tibshirani et al. 2024, Equation 11]. The observation in Equation 3.14 is an example of *Laplace's method*, a classic technique in asymptotic analysis [Evans 1998, §4.5], [Bender and Orszag 1999, §6.4].

Intuitively, the integral on the left-hand side of Equation 3.14 becomes increasingly peaked where φ is smallest, such that in the limit all other contributions become *subdominant*, that is, exponentially small with respect to this peak contribution. By applying $-\frac{1}{\lambda}\log(\cdot)$ to both sides of Equation 3.14, asymptotically we obtain a direct estimate of the minimum of φ:

$$-\frac{1}{\lambda}\log\left(\int_\Omega h(z)\exp\left(-\lambda\varphi(z)\right)\,\mathrm{d}z\right) \sim \varphi(x^*) + \frac{d}{2}\frac{\log\lambda}{\lambda} - \frac{\log h(x^*)}{\lambda} + O(\lambda^{-1}), \quad \lambda\to+\infty.$$

All terms beyond the first go to 0 as $\lambda\to\infty$.

Taking $\varphi(z) = \|x-z\|$, and taking the domain of integration to represent the source geometry Ω to which we compute distance, we define

$$\boxed{\widetilde{d}^\lambda(x) = -\frac{1}{\lambda}\log\left(\int_\Omega h_x(z)\exp\left(-\lambda\|x-z\|\right)\,\mathrm{d}z\right)} \tag{3.15}$$

as the general form of a *convolutional distance formula* that approximates the minimum distance from a point x to Ω. A special case of Equation 3.15 is Equation 3.13 we derived in Section 3.1 via the Hopf-Cole transformation: there, $h_x(z) = \frac{(\lambda\|x-z\|+1)\langle x-z, n(z)\rangle}{2\pi\|x-z\|^3}$. Equation 3.15 also subsumes unsigned distance approximations like *LogSumExp distance* [Madan and Levin 2022], the *Kreisselmeier-Steinhauser* function [Kreisselmeier and Steinhauser 1980], *Varadhan's formula* [Varadhan 1967], and the *Schrödinger distance transform* [Gurumoorthy and Rangarajan 2009] as special cases. If x is on the cut locus of Ω, then φ no longer has a unique minimizer, in which case the distance approximation in Equation 3.15 has additional asymptotic error that gets absorbed into the $O(\lambda^{-1})$ term. Note that while Madan and Levin [2022] show that Equation 3.15 can give a conservative distance field — that is, it underestimates the true unsigned distance to Ω — their proof only holds assuming the input discretized geometry is *exactly* the true surface Ω, which is rarely the case (especially for point clouds).

We can also obtain a *self-normalized* convolutional formula (terminology we take from Belyaev and Fayolle [2024])

$$\boxed{\widehat{d}^\lambda(x) = \frac{\int_\Omega g_x(z)\exp\left(-\lambda\|x-z\|\right)\,\mathrm{d}z}{\int_\Omega \exp\left(-\lambda\|x-z\|\right)\,\mathrm{d}z}} \tag{3.16}$$

that gives a smooth approximation of $g_x(x^*)$ as $\lambda\to\infty$, where $x^* = \operatorname{argmin}_{z\in\Omega}\varphi_x(z)$ is again the minimizer of $\varphi_x(z) = \|x-z\|$. Equation 3.16 can be obtained either by applying Laplace's method twice (once to the numerator, and once to the denominator), or by taking the derivative of Equation 3.15 with respect to λ.

Equation 3.16 is an example of a *kernel density estimator*, and offers greater flexibility than Equation 3.15 because it instead interpolates the function g, which need not even be scalar-valued. Equation 3.16 can also be seen as a partition-of-unity method that computes an expected value of $g(x, z)$, where the local estimate at $z = z'$ has probability $\exp(-\lambda\|x-z'\|)/\int_\Omega \exp(-\lambda\|x-z\|)$.

3.3 Fundamental limitations of convolutional distance

As $\lambda \to \infty$, the convolutional distance formulas in both Equations 3.15 and 3.16 are completely determined by the behavior of φ or g (*resp.*) around the global minimizer x^* of the exponential argument φ. The fact that convolutional distance approximations are essentially single-point approximations severely hinders their generalizability to point clouds: knowing a good local function φ or g that gives globally accurate signed distance to the true geometry at x^* is just as hard as the general global problem of signed distance reconstruction.

Assuming no noise or errors in the point cloud, convolutional distance approximations do converge to the correct solution as sampling becomes increasingly dense. But Equation 3.15 fares especially poorly on point-sampled surfaces, where one obtains distance to the *sampled* geometry, rather than to the underlying surface from which the discrete geometry is sampled (Figure 3.3). Thus winding numbers and Poisson surface reconstruction take $\lambda \to 0$, achieving reconstruction at the expense of distance. But as $\lambda \to \infty$, the distance function simply "snaps" to the closest point in the input point set. Unfortunately, regularizing the exponential kernel is futile. Chen et al. [2024a] use regularized kernels to avoid the numerical and interpolation issues of ordinary winding numbers, which can be interpreted as adopting a stochastic model of the point cloud geometry. However, the choice of regularizing kernel is equivalent to the choice of h, which has no effect asymptotically as $\lambda \to \infty$ (Figure 3.2). In other words, while we need very high λ to get good distance properties, the asymptotic properties of the exponential function rapidly negate the effect of any regularization or "blurring" we might choose.

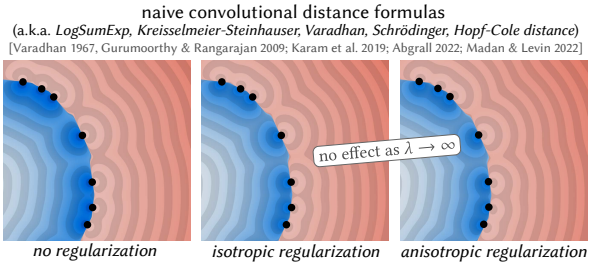


Fig. 3.2: Regularizing the kernel in Equation 3.15 is equivalent to choosing the function h, which unfortunately has no effect asymptotically. Here, we use an exponential kernel with no regularization (*left*), with isotropic Gaussian regularization (*center*), and with anisotropic Gaussian regularization (*right*), which all produce non-generalized distance for $\lambda = 100$.

Ultimately, we can either obtain better regression quality, or better eikonality, but not both: the parameter λ is coupled to both regression quality and distance accuracy, in opposite directions. Empirically, there are no values of λ that yield acceptable results in both. On one end of the spectrum, λ is small and one obtains winding number methods, Poisson surface reconstruction, and the method of Madan and Levin [2022], methods which essentially sacrifice distance accuracy to focus on regression quality (reconstruction). On the other end of the spectrum, one has the convolutional distance transforms applied in image processing [Gurumoorthy and Rangarajan 2009; Sethi et al. 2012; Karam et al. 2019], where issues of robustness likely never occurred: these

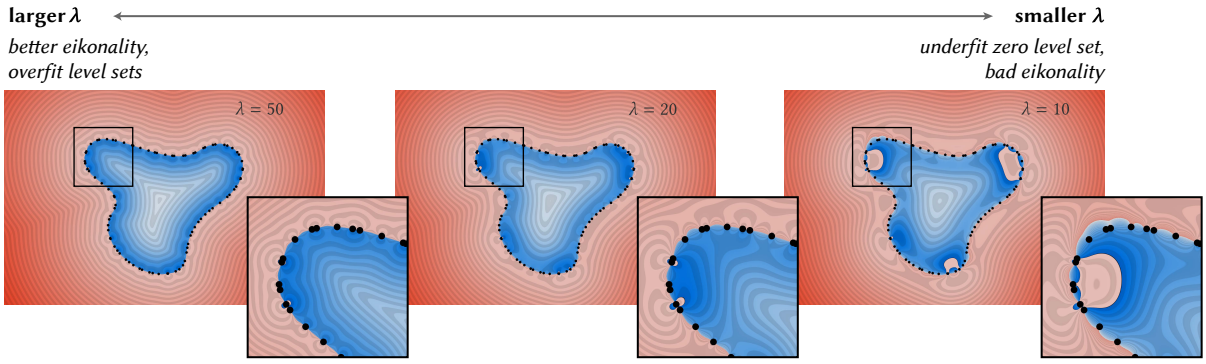
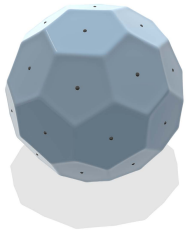


Fig. 3.3: In naive convolutional distance formulas (Equation 3.15), the parameter λ is coupled to both regression quality and distance accuracy, in opposite directions: large λ values are needed to obtain better eikonality, but overfits distance to the point set (*left*); using smaller λ at least interpolates the data points, but isn't close to a distance function (*center, right*). These figures use Equation 3.13 with regularization [Chen et al. 2024a] using $\varepsilon = 0.2$, for stability; even so, it is easy to get artifacts when λ is smaller.

transforms operate on dense image contours whose sampling resolution matches the resolution of the image on which distance is being computed.

We also cannot alter the exponential part of the kernel, because the exponential factor is the key to the asymptotic behavior that enables distance approximation. Kernels that decay slower — for example, kernels of the form $1/\|x-z\|^\lambda$ as in *Shepard interpolation* — need even larger values of λ to achieve good distance, while suffering from the same drawbacks as essentially approximations of the exponential. Kernels that decay faster, such as the Gaussian $\exp\left(-\lambda\|x-z\|^2\right)$, suffer immensely from numerical instability, and anisotropic kernels alter the metric with which distance is measured. Spatially varying λ in Equation 3.15 compromises eikonality.

Instead, the self-normalized variant of convolutional distance approximations (Equation 3.16) holds more promise because we can control the manifold's landscape at the closest point via the functions $g(x,z)$. Many authors [Alexa et al. 2001; Boissonnat and Cazals 2002; Kolluri 2008; Öztireli et al. 2009; Yang et al. 2025] use the naive signed planar distance $g(x,z) = \langle x - z, n(z)\rangle$ (a.k.a. tangent plane approximation, plane test, or *pseudonormal distance* [Bærentzen and Aanæs 2005]) yielding simple linear continuation of surfaces (inset). Chapter 5 describes a method that instead fits second-order surfaces, and Chapter 4 describes a method that instead uses vector-valued g to obtain generalized signed distance.



3.4 Relation to kernel methods

Equation 3.15 and Equation 3.16 construct global distance functions by blending exponentially-weighted local distance approximations centered on the geometry Ω, in the style of countless other classic meshless interpolation methods that use exponential radial basis functions, such as moving least squares surfaces, partition-of-unity methods, and meshless methods for solv-

ing PDEs (*e.g.* smoothed particle hydrodynamics). Sharp et al. [2019c] also use a version of Equation 3.16 to estimate closest-point interpolation of both scalar- and vector-valued data on manifolds.

More generally, kernel methods find broad use in regression tasks. Kernel methods are used, for example, in *manifold learning*, where one assumes that high-dimensional data lies on a low-dimensional subset. More recently, kernel density estimators underpin the "attention" mechanism in transformer neural network architectures [Vaswani et al. 2017]. In graphics, the idea of constructing a smooth interpolant or regressor by convolving basis functions against a finite number of samples has been a powerful paradigm in modeling. However, although such methods can be efficient, prior to this thesis they had not yet been designed to give good-quality signed distance for point clouds.

Implicit surface modeling. Blinn [1982] introduced implicit modeling with radial basis functions to the graphics community, where they became known as "blobs" or "metaballs" and inspired many variations [Bloomenthal and Shoemake 1991; Muraki 1991; Tigges et al. 1999; Morse et al. 2005]. Later this implicit approach was adopted for surface reconstruction, an approach often called *(implicit) moving least squares* or *partition of unity surfaces* [Lancaster and Salkauskas 1981; Turk and O'Brien 1999; Carr et al. 2001; Boissonnat and Cazals 2002; Amenta and Kil 2004; Shen et al. 2004; Dey and Sun 2005; Ohtake et al. 2005; Kolluri 2008; Öztireli et al. 2009; Zagorchev and Goshtasby 2012; Fuhrmann and Goesele 2014]. Methods that use basis functions to interpolate data are also referred to as *meshless interpolation* [Belytschko et al. 1996; Nguyen et al. 2008]. Recent works have adopted the implicit moving least squares framework as a differentiable shape representation [Liu et al. 2021; Yang et al. 2025]; but like other neural implicit methods, these works do not provide globally accurate signed distance reconstructions.

Unlike classic implicit modeling or distance blending methods, our method does not suffer from bulging artifacts [Bloomenthal 1997; Biswas and Shapiro 2004; Gourmel et al. 2013; Madan and Levin 2022] because we combine kernels in a way that is designed to give distance to the true geometry. Furthermore, our methods are designed to give a globally accurate signed distance field that remains well-behaved in the presence of holes and noise.

Other convolutional distance methods. Instances of the two convolutional formulas in Equation 3.15 and Equation 3.16 appear widely across mathematics, computer science, and engineering. For instance, Varadhan [1967] uses similar asymptotic analysis as in Section 3.2 to derive two formulas for geodesic distance with the same exponential change-of-variables as Hopf-Cole. Many authors have used Equation 3.15 as a smooth approximation for unsigned distance [Gurumoorthy and Rangarajan 2009; Sethi et al. 2012; Karam et al. 2019; Abgrall 2022; Madan and Levin 2022]. Close to our work, Madan and Levin [2022] use a kernel method to approximate a distance field to meshes; unfortunately, Equation 3.15 is difficult to balance reconstruction and distance for point clouds (Figure 5.2). More recently, the relationship between eikonal and screened Laplace equations has been used in neural reconstruction methods [Lipman 2021; Wang et al. 2025a]. Other variants include *p-Poisson* or L_p *distances*, which operate on similar asymptotic principles [Belyaev and Fayolle 2015]. However, none of these methods can directly provide signed distance to point clouds, as discussed in Section 3.3.

More generally, the exponential asymptotics that underlie convolutional distance formulas also underlie common logistic regression techniques used for clustering and classification. Equation 3.15, when applied to discrete data, is a generalization of the *LogSumExp* function, commonly used as a smooth relaxation of the minimum or maximum operator in machine learning. The LogSumExp function is known in the systems and control community as the *Kreisselmeier-Steinhauser function* [Kreisselmeier and Steinhauser 1980]. The gradient of the LogSumExp function, called the *softmax function* in machine learning, is an instance of Equation 3.16. Tibshirani et al. [2024] gives an excellent survey of further connections between Laplace's method and smooth minimizers throughout the fields of convex optimization, statistics, and machine learning.

Exponential asymptotics are useful for clustering problems, since the incredibly strong attraction of the exponential kernel to minimizers aids the partition of data into discrete clusters. For example, applying *maximum log-likelihood estimation* to *Gaussian mixture models*, a method common in statistical learning, yields an iterative kernel density estimator. Similarly, the SoftMax function is used as an activation function in classifier neural networks precisely because it is likely to pick a single "winning" category. The same principle underlies the method of *exponential tilting* and other sampling methods motivated by statistical mechanics. On the other hand, exponential asymptotics can be detrimental for non-clustering regression problems. Kolluri [2008] derives sampling requirements under which their reconstruction converges, though does not propose a robust algorithm; other authors suggest anisotropic kernels [Levin 1998; Adamson and Alexa 2006; Zagorchev and Goshtasby 2012], spatially-varying kernel bandwidths [Wang et al. 2008; Öztireli et al. 2009; Fuhrmann and Goesele 2014], hierarchical schemes [Ohtake et al. 2003], or regularized kernels [Chen et al. 2024a], but these regularizations hinder or have no effect on signed distance reconstruction (Section 3.3). Huang et al. [2023]; Williams et al. [2022] use learned neural kernels for surface reconstruction, but do not provide signed distance.

Generative modeling. One class of regression problems where exponential asymptotics really hurt are generative models. One such class of generative models aim to sample from a target probability distribution ρ_1 by transforming samples from an easy-to-sample distribution ρ_0. *Diffusion models*, for example, parameterize the transformation as a time-dependent differential equation

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \frac{1}{t}z + \left(\frac{1}{t} - 1\right)\nabla \log \rho_t(z) \qquad\qquad t \in (0, 1]$$

where z is a random variable that represents a sample $x \sim \rho_1$ that has been perturbed by some noise; one usually applies this model to training data to learn how to sample from the target distribution [Liu et al. 2022; Scarvelis et al. 2023].

When the source distribution is a normal distribution from which we sample Gaussian noise, at each timestep the transformed distribution is simply a mixture of (increasingly lower-variance) Gaussians centered at the training points, and the score function has a closed-form expression

where

$$\nabla \log \rho_t(z) \propto \frac{\sum_{i=1}^N t x_i \exp\left(-\frac{1}{2}\frac{\|z-tx_i\|^2}{(1-t)^2}\right)}{\sum_{i=1}^N \exp\left(-\frac{1}{2}\frac{\|z-tx_i\|^2}{(1-t)^2}\right)} - z$$

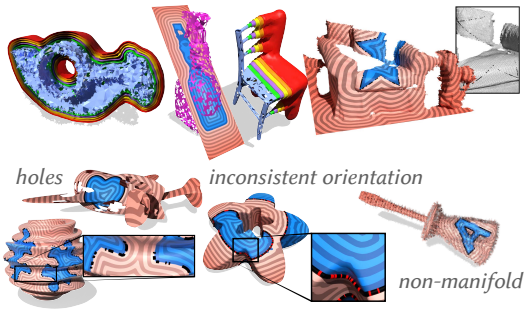
where the first term can be seen as an approximation of $\operatorname{argmin}_{tx_i} \|z-tx_i\|^2$. Hence this expression represents a vector that points from the current sample z to an estimate of $\operatorname{argmin}_{tx_i} \|z-tx_i\|^2$, meaning the above ODE can be interpreted as an over-relaxation step that gradually pushes samples towards the closest training point.

Hence flow-based generative models such as *diffusion models* and *flow matching* amount to simple kernel regression formulas based on Equation 3.16 that push samples towards the closest datapoint seen in training [Scarvelis et al. 2023; Gao and Li 2024; Kamb and Ganguli 2025]. They can only "memorize" their training data, simply reproducing the discrete training distribution rather than the true distribution from which the training data is sampled — exactly the phenomenon underlying the fundamental limitation of naive convolutional distance formulas (Section 3.3). This observation about generative models has been made by many authors [Liu et al. 2022; Somepalli et al. 2023; Pidstrigach 2022; Yoon et al. 2023; Carlini et al. 2023; Jain et al. 2025; Gu et al. 2025; Biroli et al. 2024]. The current state-of-the-art achieves generalization by instead training expensive neural networks to approximate the score function, rather than using the closed-form formula: generalization doesn't come from the flow model, but rather from the neural network (over)regularizing the score function. Scarvelis et al. [2023] develop a closed-form generative model by regularizing the kernel in a manner reminiscent of Chen et al. [2024a], though adding noise to the model can easily lead to lower-quality output [Arjovsky et al. 2017].

The upshot is that these asymptotics are based on scalar diffusion: to gain robust behavior, we must use higher-order information. Bamberger et al. [2025] improve generalization of flow matching by estimating the tangent space of the learned manifold at each datapoint, and replacing isotropic Gaussians with anisotropic ones aligned to the manifold. In our case, we aim to not only obtain better "coverage" of the learned manifold, as in generative modeling, but also optimize the shape of the level sets of a globally-defined SDF.

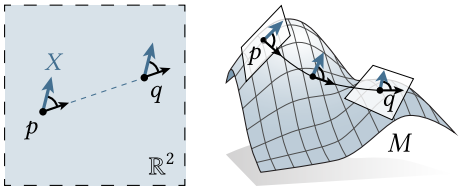
Chapter 4

Generalized signed distance through vector diffusion



We now present our first algorithm for computing generalized signed distance directly from broken geometry, which we call the *signed heat method (SHM)*. At a high level, the SHM is based on *short-time heat diffusion*, which, like the convolutional distance formulas introduced in Chapter 3, relies on convolution with an exponential kernel. However, the SHM introduces an intermediate step that avoids the diffusion-based pitfalls of naive convolutional distance approximations (Section 3.3). As a PDE-based approach, the SHM can be used to compute geodesic distance on curved domains, applies to many discretizations (Figure 4.5), and is robust to broken input with corrupted topology, geometry, and orientation (Figure 4.16) and to errors in the underlying domain (Figure 4.7). More broadly, its variational, PDE-based approach enables extensions not possible with other geodesic distance algorithms. A further discussion of the SHM's advantages and disadvantages relative to other methods is contained in Chapter 7.

4.1 The asymptotics of vector diffusion



We consider an n-dimensional Riemannian manifold M with metric g, and want to compute the signed distance function ϕ for a submanifold $\Omega \subset M$.

The signed heat method begins by considering *parallel transport* of vectors along shortest geodesics, or shortest paths, between points on Ω and points in M. In the plane, for example, the shortest geodesic $\gamma_{p\to q}$ between two points p and q is simply a straight line connecting the two points. Parallel-transporting a tangent vector $X \in T_p\mathbb{R}^2$ along $\gamma_{p\to q}$ means

transporting X so that it remains parallel to its initial state throughout its travel, and thus amounts to translating the initial vector X along $\gamma_{p\to q}$ (inset, *left*). On a Riemannian manifold M, we can similarly ask that a tangent vector $X \in T_pM$ maintain a constant angle with the tangent of the geodesic during transport (inset, *right*). (For more description of parallel transport, see Chapter 2.)

A remarkable fact is that *vector heat diffusion* yields parallel transport along shortest geodesics, as diffusion time goes to zero [Berline et al. 1992, Theorem 2.30]. To understand this fact intuitively, it's worth understanding *scalar* heat flow on $\mathbb{R}^d$: the scalar heat equation states that a temperature distribution $q_t(x) : \mathbb{R}^d \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ changes in time according to its Laplacian,

$$\frac{\mathrm{d}}{\mathrm{d}t} q_t = \Delta q_t, \quad t > 0. \tag{4.1}$$

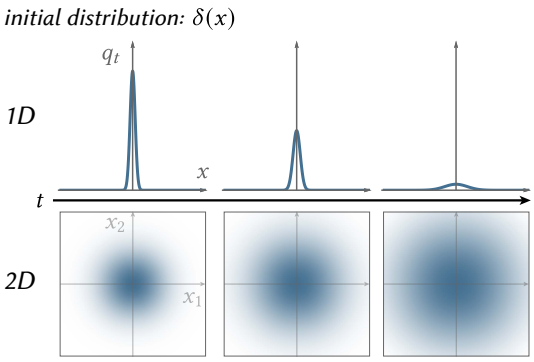


Fig. 4.1: A Dirac-delta distribution gets blurred out by scalar heat flow.

The Laplacian has many interpretations, for example, as providing a local measure of curvature, or average deviation from one's neighbors. Thus flowing the heat equation for some amount of time smooths out "bumps" in the initial distribution q_0; in particular, q_t can be expressed as a convolution of q_0 with the scalar *heat kernel*

$$\kappa_t(x, y) = (4\pi t)^{-d/2} \exp\left(-\frac{1}{4t}\|x - y\|^2\right), \tag{4.2}$$

so that $q_t(x)$ looks like a blurred version of q_0 for $t > 0$, characterized by exponential decay (Figure 4.1).

We can likewise define a vector heat equation that acts on vector-valued data,

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = \Delta X_t, \quad t > 0. \tag{4.3}$$

In $\mathbb{R}^d$, vector heat diffusion is equivalent to simply applying the scalar heat diffusion component-wise to an initial vector field, meaning vectors get translated throughout $\mathbb{R}^d$, while their magnitudes decay according to the scalar heat kernel in Equation 4.2 (Figure 4.2, *left*). On manifolds, where the tangent space at every point locally looks Euclidean, the vector heat kernel has the asymptotic expansion

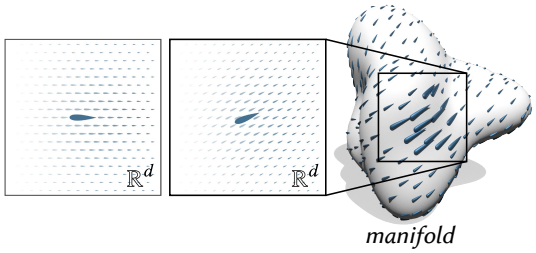
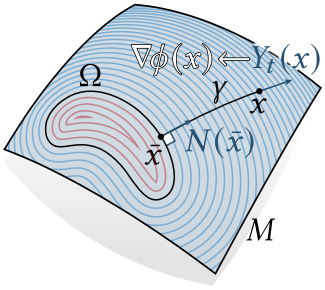


Fig. 4.2: *Left:* An initial vector (large) is diffused in $\mathbb{R}^d$, yielding exponential decay in its magnitude but preserving its orientation. *Right:* The same picture intuitively applies on manifolds.

$$k_t^{\nabla} \sim (4\pi t)^{-d/2} \exp\left(-\mathrm{dist}(x, y)^2/4t\right) c(x, y) \sum_{i=0}^{\infty} t^i \Phi_i^{\Delta}(x, y) \tag{4.4}$$

where the magnitude is governed by the same exponential decay that describes the scalar heat kernel in Euclidean space, and the leading term $\Phi_0^\Lambda(x,y)$ is the parallel transport map $P_{\gamma_{x\to y}}$ that takes vectors at a point x to vectors at another point y along the shortest geodesic $\gamma_{x\to y}$ connecting the two ($c(x,y)$ is a surface-related scalar-valued term). Using Equation 4.4, Sharp et al. [2019c] use short-time vector diffusion to perform parallel transport along shortest geodesics on triangle meshes; they also enable closest-point interpolation of vectors and their magnitudes by normalizing diffused vectors by diffused scalar data, essentially generalizing the self-normalized convolutional formula in Equation 3.16 to manifolds.

4.2 Vector diffusion for generalized signed distance



In our context, we recognize the following implications for computing signed distance. In particular, Berline et al. [1992, Theorem 2.30] implies that as $t \to 0$, the diffused vector $X_t(x)$ at each point $x \in M$ aligns with the vector obtained via parallel transport of the normal $N(\bar{x})$ at the closest point $\bar{x} \in \Omega$, along a minimal geodesic γ. 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. (These properties generalize a well-known property of SDFs in $\mathbb{R}^d$: in $\mathbb{R}^d$, at points $x \in M$ away from the cut locus of Ω, the gradient of an SDF is equal to the normal of Ω at the closest point to x on Ω.) We can hence normalize X_t to obtain an approximation $Y_t := X_t/\|X_t\|$ of the signed distance gradient.

Hence the first step of our algorithm is to diffuse the normals N of Ω to the rest of the domain M for a short time $t > 0$. In particular we solve a vector-valued diffusion equation

$$\begin{aligned} \tfrac{\mathrm{d}}{\mathrm{d}t} X_t &= \Delta^\nabla X_t, \quad t > 0, \\ X_0 &= N\mu_\Omega \end{aligned} \tag{4.5}$$

where Δ^∇ denotes the negative-definite *connection Laplacian* on M, and $N\mu_\Omega$ is a vector-valued measure equal to zero away from Ω, and determined by N for points in Ω [1]. Intuitively, solving Equation 4.5 extends information about surface orientation to the rest of the domain, though is not merely a heuristic: the approach is motivated by a key observation from differential geometry (Equation 4.4).

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

$$\min_{\phi:\, M\to\mathbb{R}} \int_M \|\nabla\phi - Y_t\|^2. \tag{4.6}$$

[1] Intuitively, we use μ_Ω to denote a measure concentrated on Ω, similar in spirit to an indicator function. 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.

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

$$\begin{aligned} \Delta\phi &= \nabla \cdot Y_t \text{ on } M \\ \frac{\partial\phi}{\partial n} &= n \cdot Y_t \text{ on } \partial M, \end{aligned} \tag{4.7}$$

where Δ denotes the negative-definite Laplace-Beltrami operator on M. The solution of Equation 4.7 is determined up to a constant shift, leaving us the freedom to enforce either exact or approximate interpolation of Ω by a level set of ϕ (Section 4.4).

In summary, we arrive at the following algorithm:

1. Solve a vector diffusion equation $\frac{d}{dt}X_t = \Delta^\nabla X_t$ (Equation 4.5), which diffuses the normals of Ω for a small time $t > 0$.
2. Evaluate the vector field $Y_t = X_t / \|X_t\|$, yielding a unit vector field that approximates the gradient of the (unknown) SDF.
3. Solve a Poisson equation $\Delta\phi = \nabla \cdot Y_t$ (Equation 4.7), to find the function whose gradient best matches Y_t.

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 ensures that we recover a distance approximation, rather than just a smooth function.

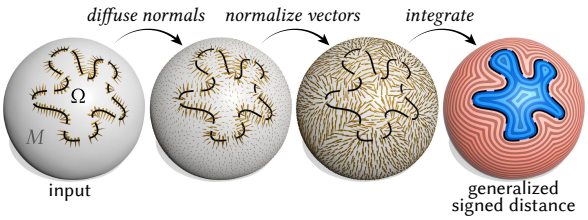


Fig. 4.3: The three steps of the signed heat method.

The SHM ultimately relies on the exponential asymptotics of the heat kernel, much in the same way as convolutional distance approximations (Chapter 3). However, whereas convolutional distance approximations tightly couple both distance accuracy and reconstruction, and thus sacrifice their quality in both (Section 3.3), the SHM instead achieves good quality in both. The SHM's intermediate step of normalizing gradient vectors, like the original heat method for unsigned distance [Crane et al. 2013b], partially decouples distance and reconstruction quality, while achieving especially good reconstruction by diffusing vector-valued rather than scalar-valued data (Figure 4.4). While the asymptotic behavior of short-time vector diffusion means that in the limit $t \to 0$, points simply inherit the normal at the closest point like in pseudonormal distance approximations (Section 3.3), in practice

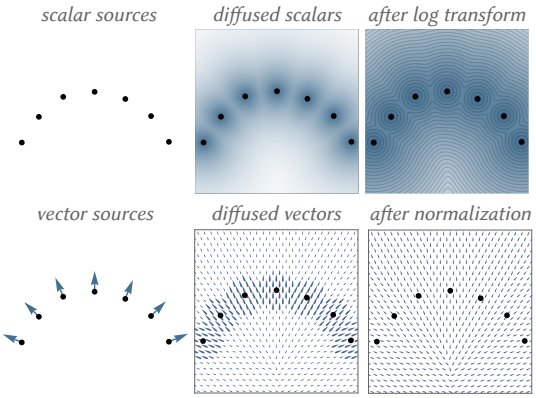


Fig. 4.4: *Top:* Short-time scalar diffusion concentrates data tightly around the initial source, preventing generalization. *Bottom:* The signed heat method instead relies on short-time *vector* diffusion. While the magnitudes of the vectors remain tightly concentrated, their *orientations* are accurate — so one can normalize and integrate this vector field to obtain generalized signed distance.

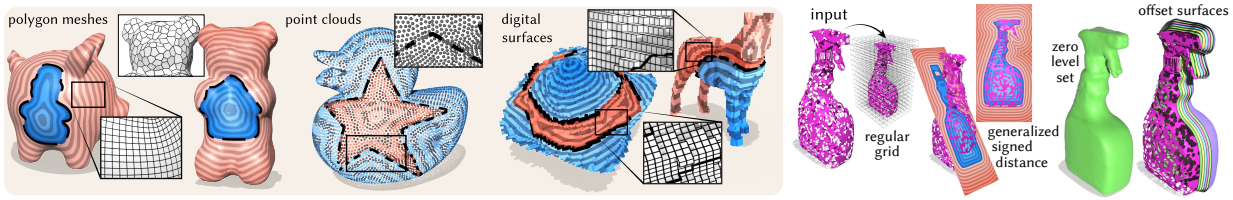


Fig. 4.5: *Left:* The signed heat method extends to polygon meshes, point clouds, and digital surfaces. *Right:* The method extends to volumetric domains, such as regular grids in $\mathbb{R}^3$. Contouring this function yields well-behaved and evenly-spaced offset surfaces. Digital surface meshes are from Coeurjolly and Levallois [2015].

we take finite diffusion times $t > 0$ and perform a global L^2 integration that yields well-behaved shape completions.

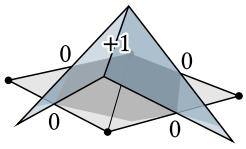
4.3 The signed heat method

The signed heat method [Feng and Crane 2024] mainly involves solving two sparse linear systems, corresponding to Steps 1 and 3.

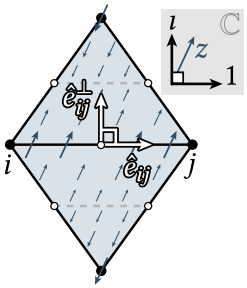
Time discretization. As in past heat methods, Equation 4.5 is discretized in time via one step of backward Euler [Crane et al. 2013b; Sharp et al. 2019c], and solve a linear equation

$$(\mathrm{id} - t\Delta^{\nabla})X_t = X_0 \tag{4.8}$$

for a single, fixed time $t > 0$ (where id is the identity). On meshes, this time is set simply to the square of the mean spacing between nodes.



Spatial discretization on triangle meshes. On triangle meshes, we use edge-based operators for the vector diffusion step (Step I), which makes it straightforward to discretize curve sources: we don't have to map from the tangent space of curve segments, which generically lie within triangle faces, to the tangent space of vertices. We can instead discretize within a single face using only intrinsic operations, opening the possibility of further improving accuracy and robustness using *intrinsic Delaunay refinement* [Gillespie et al. 2021]. In particular, we use *Crouzeix-Raviart (CR) basis functions*, which associate each edge $ij \in E$ with a facewise linear function $\varphi_{ij} : M \to \mathbb{R}$ interpolating the value 1 at the midpoint of ij, and 0 at all other edge midpoints (inset).



As in Djerbetian [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). In particular, a basis for vector fields is expressed by identifying tangent vectors with complex numbers. At each edge ij we choose a coordinate system such that 1 and the imaginary unit $\imath$ correspond to $\hat{e}_{ij}$ and $\hat{e}_{ij}^{\perp}$, *resp.* The function $\psi_{ij} := \varphi_{ij} + 0\imath$

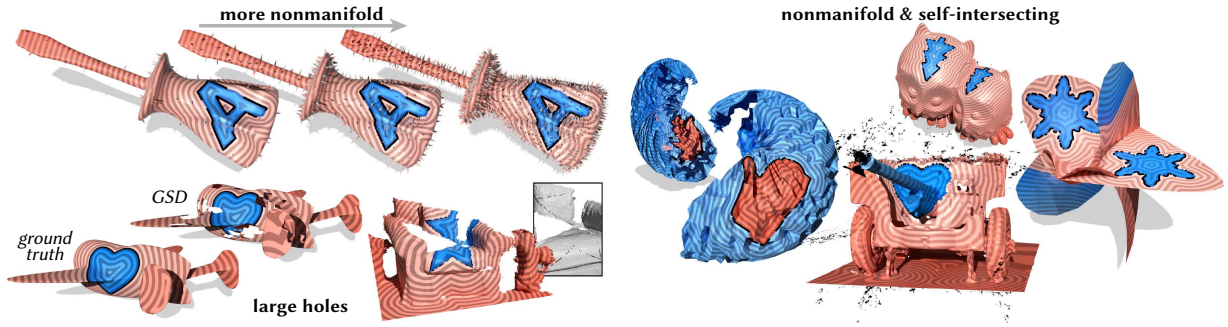


Fig. 4.7: The signed heat method is robust not only to errors in the source geometry, but also in the domain mesh itself. Here we obtain well-behaved SDFs even for meshes found "in the wild," such as amateur-created 3D scans [Choi et al. 2016]. Even in cases where a notion of inside and outside is meaningless (such as the rightmost mesh), our method fails gracefully — still producing a good signed distance approximation near the input curve.

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).

The discrete algorithm amounts to solving two sparse linear systems. First we solve the discrete vector heat equation, which uses the discrete Crouzeix-Raviart connection Laplacian L^∇ and mass matrix M,

$$(\mathsf{M} + t\mathsf{L}^\nabla)\mathsf{X} = \mathsf{X}_0, \tag{4.9}$$

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.

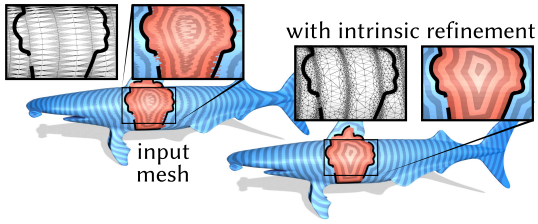
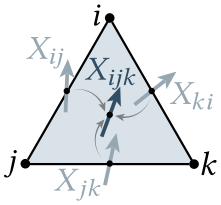


Fig. 4.6: Since our formulation is purely intrinsic, we can trivially improve accuracy and robustness by invoking *intrinsic Delaunay refinement* [Gillespie et al. 2021], without changing the implementation.



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), 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

$$\mathsf{C}\phi = \mathsf{b} \tag{4.10}$$

where C is the cotan Laplacian, $\mathsf{b} \in \mathbb{R}^{|V|}$ is a vector of discrete divergences. For full detail on the implementation of the Laplacian, connection Laplacian, and divergence operator on triangle meshes, as well as the discretization of signed and unsigned curve and point sources (Figure 4.10), see Feng and Crane [2024, Section 5].

Additional discretizations. Since the method is based purely on intrinsic PDEs, these equations can be applied not just on triangle meshes, but virtually any data structure (Figure 4.5): all we need is a divergence operator and Laplacian defined on the data structure, and we simply

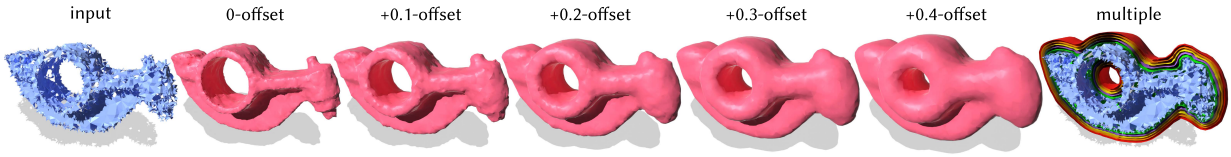


Fig. 4.8: By extracting level sets of generalized signed distance, we can convert broken, noisy, and nonmanifold input geometry *(far left)* into closed, regular, manifold surfaces and evenly-spaced offset surfaces.

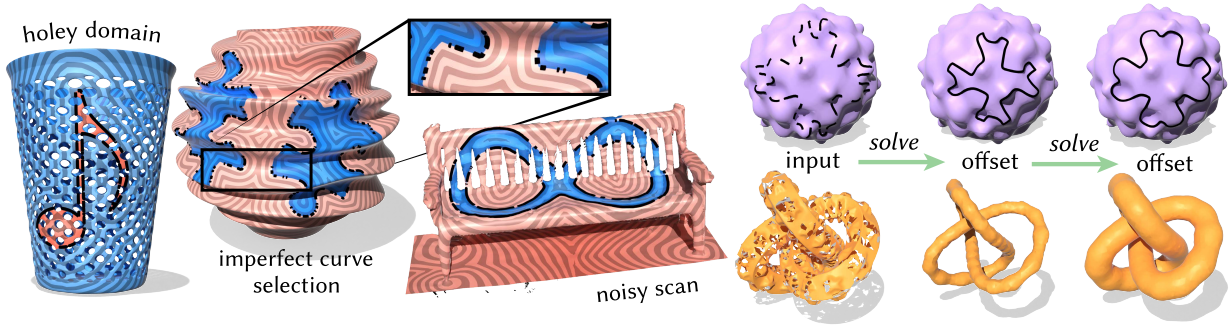
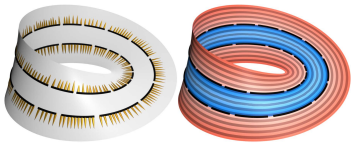


Fig. 4.9: *Left:* One can compute signed distance to broken curves that arise from attempting to draw curves on surfaces of high genus, with overhangs, and with holes and scanner noise. (Scanned bench from Choi et al. [2016].) *Right:* One can simplify high-frequency features of broken curves and surfaces by taking consecutive positive/negative offsets of a generalized signed distance function, akin to dilation/erosion.

take advantage of Laplacians developed in previous work [Bunge et al. 2020; Coeurjolly and Lachaud 2022; Sharp and Crane 2020; Sharp et al. 2019c]. For example, beyond triangle meshes, our method extends to polygon meshes, point clouds, digital surfaces, regular grids, and tet meshes (Figures 4.5, 4.8).

4.4 Signed heat method results



Robustness. The signed heat method performs well on not only broken input with corrupted topology, geometry, and orientation (Figure 4.16) but also challenging surface domains (Figure 4.7). Since our method is purely intrinsic, 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 4.7, *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, but owing to our careful discretization, our method remains purely intrinsic (Section 4.3) and we can easily apply *intrinsic Delaunay triangulation* to get good-quality solutions (Figure 4.6). For surfaces in $\mathbb{R}^3$, our method remains robust, even for extremely corrupted input surfaces (Figures 4.5,4.8).

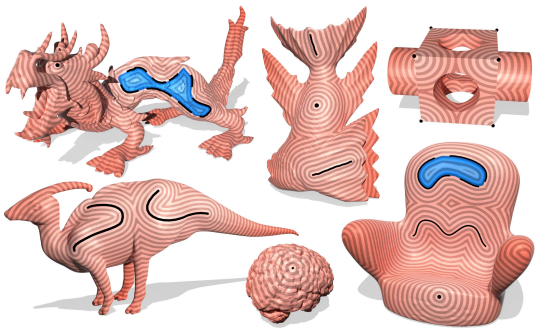


Fig. 4.10: We can mix and match signed and unsigned distance.

Examples and applications. One natural use case for generalized signed distance is to contour broken geometry, and generate accurate fixed-distance offsets (Figures 4.8; 4.5, *right*). 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 4.9, *right*).

Not only can the method compute signed distance to curves, but it can also compute unsigned distance simply by changing the direction of the normal vectors diffused in the first step. Hence the source set Ω can be a mix of oriented and unoriented curves, as well as isolated points, enabling one to combine both signed and unsigned distance in a single distance field (Figure 4.10).

Boundary behavior. Unlike the *unsigned heat method* [Crane et al. 2013b], the signed heat method exhibits the correct behavior at the boundary (Figure 4.11), 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 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 encodes zero-Neumann boundary conditions on the vector field itself [Gelfand et al. 2000, I.6], rather than a scalar potential u.

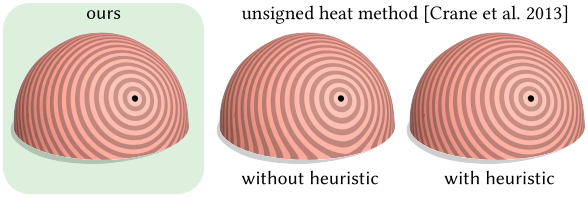


Fig. 4.11: The unsigned heat method exhibits bias near the domain boundary (figure reproduced from Crane et al. [2013b], Figure 11). Using their proposed boundary condition heuristic improves results at the cost of solving an additional linear system. In contrast, our method has correct boundary conditions without special treatment.

Preserving level sets. The global variational nature of the signed heat method also provides capabilities not possible with standard distance methods, such as iterative region-growing methods based on *MMP* [Mitchell et al. 1987] or *fast marching* [Kimmel and Sethian 1998]. For instance, one can add an explicit linear constraint to Equation 4.6 that ensures ϕ takes the same value at all points of Ω. This constraint ensures exact interpolation of the source geometry Ω, in contrast to a common strategy used in prior work that simply shifts ϕ by its average over Ω

[Kazhdan et al. 2006; Calakli and Taubin 2011; Crane et al. 2013b].

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 saddle-point problem

$$\begin{bmatrix} \mathsf{C} & \mathsf{A}^\mathsf{T} \\ \mathsf{A} & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \mu \end{bmatrix} = \begin{bmatrix} \nabla \cdot Y_t \\ 0 \end{bmatrix}, \tag{4.11}$$

where $\mu \in \mathbb{R}^m$ are Lagrange multipliers. 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 4.12, *right*).

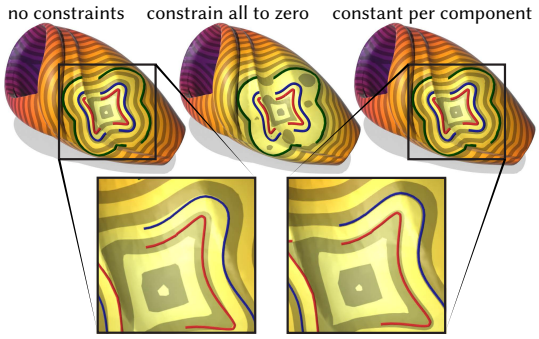


Fig. 4.12: We can fit a signed distance function to several partial level sets. *Left*: Without constraints, isolines deviate slightly from source geometry. *Center*: Constraining to zero along *all* curves grossly violates the distance property. *Right*: Constraining values to be constant along each component nicely matches the input geometry.

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]:

$$\begin{aligned} \max_{\phi} \; & \int_M \phi(x) \; \mathrm{d}x \\ \text{s.t. } & |\nabla\phi| \leq 1 \\ & \phi = 0 \text{ on } \Omega. \end{aligned} \tag{4.12}$$

Belyaev and Fayolle [2020] solve Equation 4.12 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 4.5), and more importantly, can compute only unsigned distance. If desired, however, one can "sharpen" our results using a generalization of Equation 4.12. We simply replace the objective in Equation 4.12 with

$$\max_{\phi} \int_M \operatorname{sign}(\phi_0(\mathrm{x}))\phi(\mathrm{x}) \; \mathrm{dx} \tag{4.13}$$

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 4.13. Note that robust sign information is available only thanks to our method — Equation 4.13 cannot be applied directly to broken curves.

ADMM is quite slow in practice since it requires repeated linear solves. We also explore the *primal-dual hybrid gradient method (PDHG)*, a first-order method, which is cheaper per iteration than ADMM [Chambolle and Pock 2011, Figure 5], though tends to take more iterations (though PDHG can easily be implemented in parallel). We find that setting the parameters of PDHG is not straightforward, however, and also seems more unstable (Figure 4.13,*right*), so this extension of our method would likely benefit from future work.

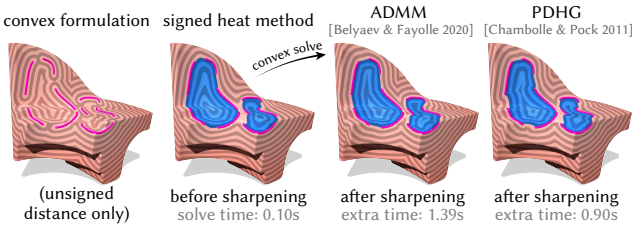


Fig. 4.13: *Left:* Methods based on convex optimization yield more accurate distances, but compute only unsigned distance. *Right:* Using our method as a warm start, we can "sharpen" distance while preserving the inside/outside classification, using either ADMM or PDHG. Here we start with a large diffusion time ($t = 100h^2$) to visually emphasize the effect.

4.5 Accuracy and performance of the signed heat method

We compare against the unsigned heat method (UHM) of Crane et al. [2013b], and also 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, and do not pre-factor any matrices. Note, however, that for multiple source terms, heat methods can achieve about two orders of magnitude speedup by omitting factorization [Crane et al. 2013b, Table 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 2021], and sign can unambiguously be determined via standard inside/outside tests [Haines 1994]. As seen in Figure 4.14, our method is slightly slower but slightly more accurate than UHM. In summary, our method is comparable to the original heat method, and less accurate

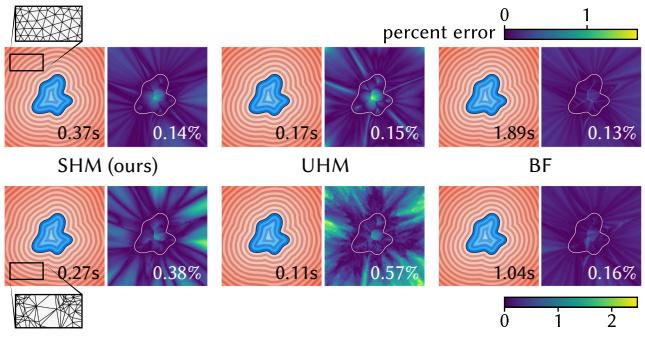


Fig. 4.14: Distribution of error in distance approximation for a perfect, unbroken curve on a high- and low-quality planar mesh *(top/bottom)*. Both meshes have about 100k faces. Inset numbers on SDF and error plots indicate compute time and mean error (*resp.*).

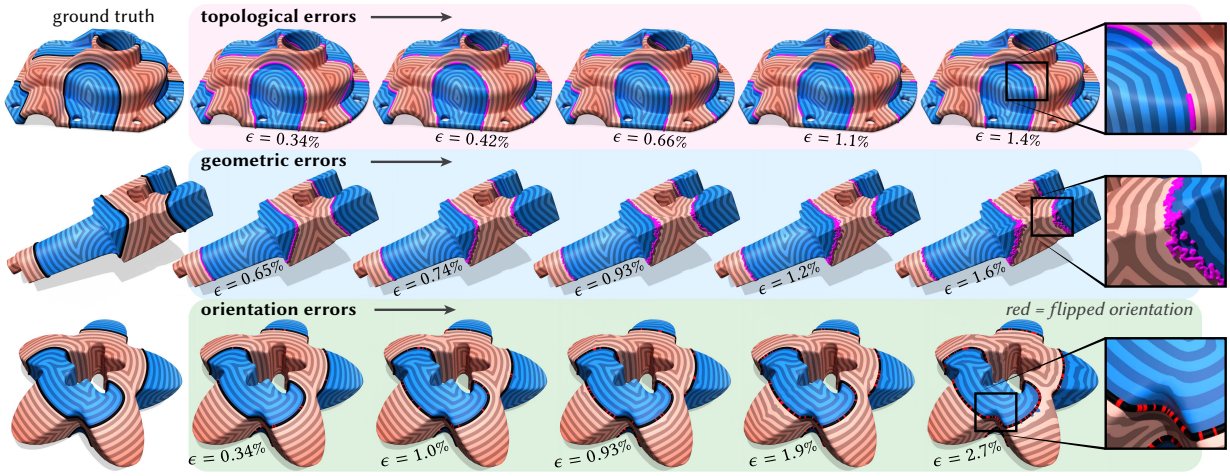


Fig. 4.16: The signed heat method provides robust and reliable signed distance approximation, failing gracefully in the presence of significant topological, geometric, or orientation errors. Errors ϵ in geodesic distance are displayed relative to the exact polyhedral SDF of a finely sampled version of the original curve.

than BF on the low-quality mesh, but about 4–5x faster. BF must also 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.*).

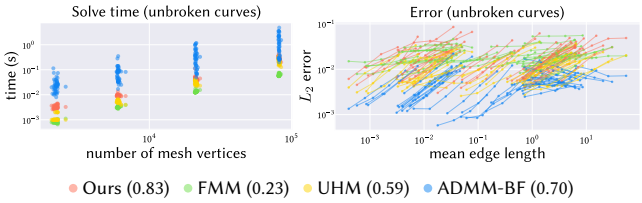


Fig. 4.15: We observe approximately linear convergence in distance accuracy on a benchmark of unbroken (closed) curves on 44 different meshes. The legend shows median orders of accuracy.

Surface domains. We next consider closed curves on surface meshes. On surface meshes, 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 from the surface mesh's approximation of the true surface. Here we additionally compare against the fast marching method (FMM). We plot convergence and solve times in Figure 4.15. 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 our experiments on planar domains. 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.

Error under progressive corruption. Figure 4.16 shows error in signed distance as the curve input becomes increasingly corrupted in topology, geometry, and orientation. The error ϵ in 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.

Future work: adaptive meshing. We present two possibilities for discretizing a volumetric domain in $\mathbb{R}^3$: either using a regular grid (Figure 4.5, *right*), or using a tet mesh (Figure 4.8). Regardless of how the domain is discretized, Figure 4.17 demonstrates the importance of adaptively meshing the domain so that resolution is concentrated near the input: the tet mesh discretization leads to more faithful surface reconstruction because the tet mesh concentrates its degrees of freedom on and near the input point cloud. (On the other hand, the far-field accuracy of the SDF on the tet mesh is lower where tets are coarse.) Future grid-based versions of the signed heat method, if used for reconstructing the zero level set in particular, should ideally also use an adaptive, non-uniform method.

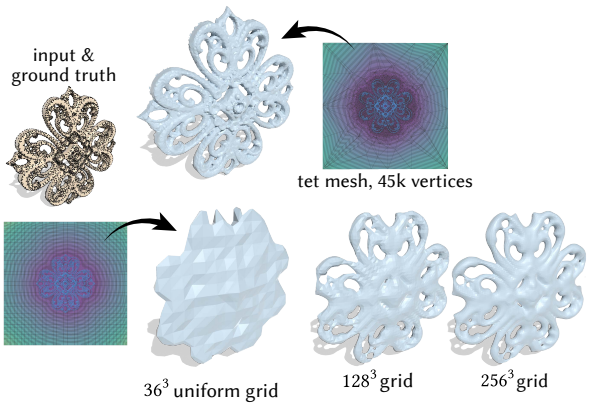


Fig. 4.17: The tet mesh discretization of the volumetric domain yields more faithful reconstruction than using a grid with a similar number of DOFs, since its DOFs are adapted to the input. A tet mesh can also be made to be constrained to the input shape boundary, enabling exact enforcement of zero set constraints.

Using a grid discretization has the advantage that constructing grids is cheaper and more straightforward than constructing a tet mesh. Using a tet mesh discretization has the advantage that zero set constraints can be enforced exactly, by generating a constrained tet mesh: in particular, one can generate a tet mesh whose faces include those of an input triangle mesh, or whose vertices include the points of an input point cloud. Enforcing these constraints exactly can also lead to more faithful surface reconstruction. Of course, if exact zero set constraints are not desired — for example, if the input geometry has noise — then zero set constraints can simply be ignored for both grid-based and tetrahedral discretization.

Chapter 5

Pointwise evaluation of generalized signed distance

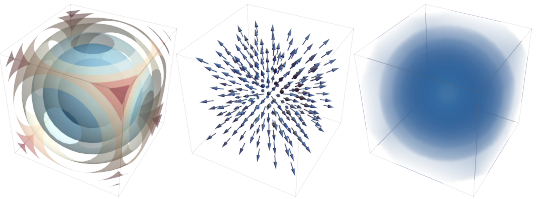
The robustness and generalization ability of the signed heat method in Chapter 4 comes from solving a global problem: in particular, by integrating a well-behaved, globally-supported extension of higher-order information about the input geometry. This global problem is solved via a linear system defined on a spatial discretization of the domain, a computational paradigm especially effective for dense evaluations of generalized signed distance (meaning simultaneous evaluation at many query points throughout the domain). In particular, solving a global system makes use of spatial coherence between query points, and allows improved amortized performance over repeated solves re-using the same matrix factorizations.

However, the signed heat method, as a finite-element-based method (FEM), inherits typical drawbacks of FEM, such as the requirement for high-quality meshing and for solving large systems of coupled variables. Many applications in computer vision, simulation, and rendering would better benefit from an *output-sensitive* algorithm for generalized signed distance that can answer isolated pointwise queries *without* global solves. Ideally, we'd also like to avoid spatial discretization of the domain, especially for large-scale or intricate scenes where meshing would be especially memory- and compute-heavy, perhaps even infeasible.

Computational limitations of the signed heat method. Unfortunately, directly adapting the signed heat method to support efficient pointwise queries proves extremely difficult. At first, it appears the signed heat method boils down to solving the Poisson problem

$$\Delta\phi = \nabla \cdot Y_t,$$

where $Y_t := X_t / \|X_t\|$ is the normalized version of the vector field solution obtained from the initial vector diffusion step. It's tempting to turn this Poisson equation into its corresponding integral equation to gain the computational flexibility we ask for, but the resulting integral is ill-posed.



As a concrete demonstration, consider a sphere of radius R centered at the origin in $\mathbb{R}^3$, which has signed distance function $\phi(x) = \|x\| - R$. The gradient of ϕ is $Y(x) = x/\|x\|$, and the divergence of Y is $(\nabla \cdot Y)(x) = 2/\|x\|$ (inset). One can verify that attempting to integrate $\phi(x) \overset{!}{=} \int_{\mathbb{R}^3} G(x,y)(\nabla \cdot Y)(y)\,dy$ yields ∞. The divergence $\nabla \cdot Y$ decays too slowly, and Green's representation formula can only be extended to infinite regions if the solution decays to zero at infinity [Brebbia et al. 1984, §2.10] — whereas a distance function goes to infinity at infinity.

Likewise, the integral equation corresponding to a finite domain M

$$\phi(x) = \int_{\partial M} \left[\frac{\partial G(x,z)}{\partial n(z)} \phi(z) - G(x,z) \frac{\partial \phi}{\partial n}(z) \right] dz + \int_M G(x,y)\,(\nabla \cdot Y_t)\,(y)\,dy,$$

using an appropriate fundamental solution $G(x,y)$, is recursive in ϕ: the first term assumes we already know the value of the solution ϕ on the domain boundary ∂M. (In contrast, the convolutional distance formulas in Chapter 3 do exist as boundary integrals, because through exponentiation the solution does have proper decay at infinity.)

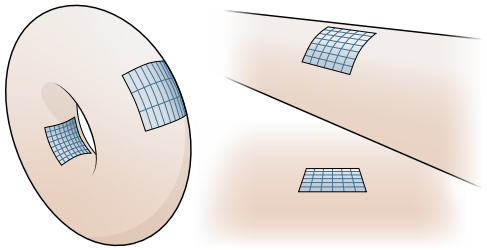


Fig. 5.1: In $\mathbb{R}^3$, SDFs can be approximated by blending the SDFs of oriented tori. Tori have simple closed-form SDF expressions, and can capture locally spherical, ellipsoidal, and saddle-shaped surfaces, as well as cylindrical and planar surfaces as limiting behavior.

A new approach. In this chapter, we focus on generalized signed distance from point clouds, which represent a significant portion of geometric data captured in the wild (*e.g.* via scanning or photogrammetry) or used for geometric processing; point clouds also represent a fairly general representation, in the sense that other representations (meshes, implicits, etc.) can be sampled into point clouds.

We revisit the convolutional distance formulas introduced in Chapter 3, which give wrong solutions when used naively, but are indeed fast and have the computational properties we want (pointwise evaluation, discretization-free). Chapter 3 showed that there are two such single-pass formulas for computing distance; in the end, we develop a novel variant of the kernel density estimator Equation 3.16 and evaluate signed distance $\phi(x)$ as

$$\phi(x) = \frac{\sum_{i=1}^{|P|} g_i(x) \exp\left(-\lambda \|x - p_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda \|x - p_i\|\right)}, \tag{5.1}$$

which gives an exponentially-weighted average of per-point functions g_i associated with a point cloud $P := \{p_i\}_{i=1}^{|P|}$. We define g_i as signed distance functions to tori, because torus SDFs have efficient closed-form expressions, and tori can locally approximate surfaces up to second order

(Figure 5.1). The parameter λ is automatically chosen with a simple expression. Our method provides the first convolutional formula that provides generalized distance: Chapter 3 shows why Equation 5.1 converges to the true SDF, and why other generalized distance formulas fail (Figure 5.2 shows one such comparison).

The main challenge in this method is fitting tori to a given point cloud P. We determine these tori by estimating the curvature and shift of a best-fit surface in the neighborhood of each point, an approach used by decades of classical point set methods. However, instead of relying on brittle heuristics or expensive robust statistics [Fleishman et al. 2005; Öztireli et al. 2009; Wei et al. 2023], we pre-train a neural regressor to *learn* surface parameters that give good signed distance to point clouds. Because we impose a local surface model, at inference time the solution is obtained via a simple analytical formula per point. At the same time, our method does not require expensive nonlinear or iterative solvers at inference time: robustness comes from the per-point learned coefficients.

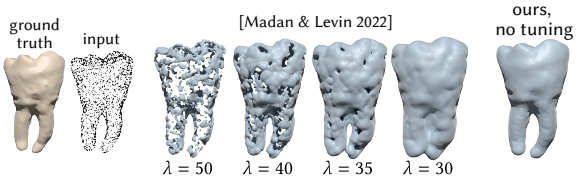


Fig. 5.2: Madan and Levin [2022] suggest estimating unsigned distance to a point cloud P using $d(x) = -1/\lambda \log\left(\sum_{i=1}^{|P|} w_i(x) \exp\left(-\lambda \|x - p_i\|\right)\right)$ with $w_i = 1$, an instance of Equation 3.15. However, it is difficult to choose a single λ that yields a good reconstruction, and easy to accidentally merge two features. We experimented with using area weights, but could not tune this method effectively. Note that spatially varying λ compromises eikonality in LogSumExp-type methods.

In summary, the method relies on a small pre-trained neural network that takes as input a size-k neighborhood of point p_i, and returns parameters yielding a "best-fit" torus intended to provide a good signed distance reconstruction of the point cloud (Section 5.1.2). Once this network is trained, one can infer signed distance from *any* point cloud with normals $P = \{p_i, n_i\}_{i=1}^{|P|}$ in two steps:

1. *Precomputation:* Pre-compute tori fitted to each point of P by forward-evaluating the pre-trained network on each point's neighborhood. This step needs to be done only once per point cloud, and can be parallelized over all points.
2. *Inference:* For a given query point x, evaluate $\phi(x)$ using the signed distance functions $\{g_i\}_{i=1}^{|P|}$ of the fitted tori in Equation 5.1.

The resulting method, which we call *points as tori (PAT)*, is less robust to noise and uneven sampling than the signed heat method, but is orders of magnitude faster (between 10^{-4} and 10^{-3} seconds for a single query to point clouds with millions of points), and more reliable in the sense that one is no longer at the mercy of volumetric meshing or linear solvers. The pointwise and differentiable nature of the method also enables interactive shader implementations (Figure 5.3) and possibly a useful representation

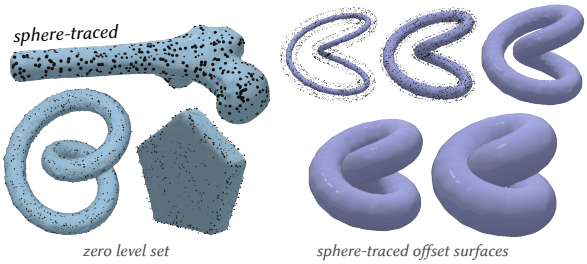


Fig. 5.3: Our method allows directly visualizing surfaces underlying point clouds through sphere tracing. Here we sphere-trace level sets in a shader; surface normals are simply given by the gradient of Equation 5.1.

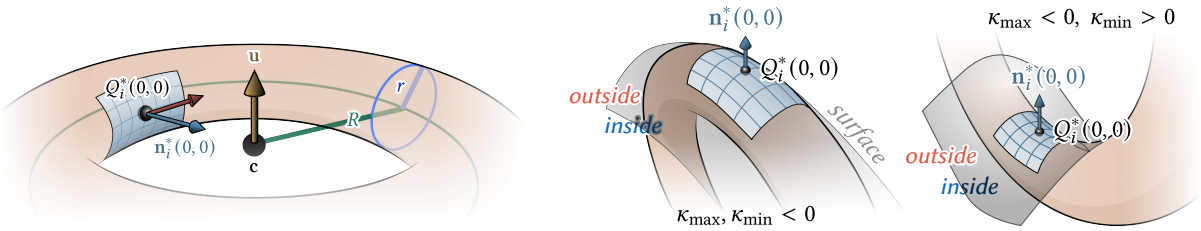


Fig. 5.4: *Left:* The ith torus's major and minor radii are aligned with the principal curvatures and directions of the estimated polynomial surface at point p_i, and its equator passes through the shifted point $Q_i^*(0,0)$. *Right:* The sign of each torus is determined by whether the torus osculates the estimated surface from the shape's interior or exterior.

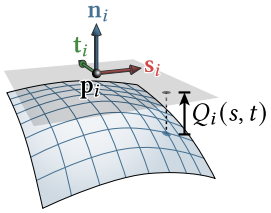
for inverse problems (Section 8.2).

5.1 The points as tori (PAT) method

PAT [Feng et al. 2026] takes as input a point cloud P with normals, and uses Equation 5.1 to estimate distance ϕ. The zero level set of the generalized signed distance function ϕ produces a surface reconstruction of the surface Ω.

5.1.1 Torus fitting

At a high level, tori are fitted such that they match the estimated principal curvatures and directions, as well as a shift coefficient, around each point. In more detail, let each point $p_i \in P$ be associated with a supporting plane defined by the normal n_i at p_i, passing through p_i (inset). Let $\{a_{n,m}\}_{n,m=0}^{2}$ be the coefficients of a polynomial describing the surface around p_i, expressed in its local coordinate frame at p_i:



$$Q_i^*(s,t) = p_i + s \cdot s_i + t \cdot t_i + Q_i(s,t) \cdot n_i,$$
$$\text{where } Q_i(s,t) = \sum_{n=0}^{2} \sum_{m=0}^{2} a_{n,m} s^n t^m. \tag{5.2}$$

A torus $\mathbb{T}_i$ is fit to each point p_i such that its equator passes through the shifted point $Q_i^*(0,0) := p_i + a_{0,0} n_i$, and its principal curvatures and directions align to those of the polynomial surface $Q_i^*(s,t)$ (Figure 5.4, *left*). In the local coordinate frame, the first and second fundamental forms of $Q_i^*(s,t)$ can be derived in closed form using standard Monge patch calculations [Feng et al. 2026, Section 4.1, Appendix

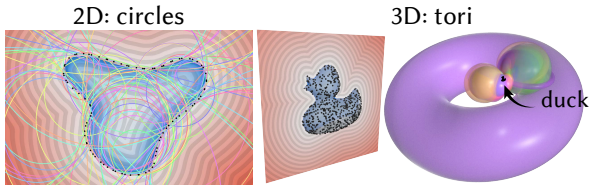


Fig. 5.5: In 2D, our method would use circles to approximate a 1D curve (*left*); in 3D, we use tori to approximate 2D surfaces, since tori generalize circles to two directions of curvature (*right*).

C]. In the end, fitting $\mathbb{T}_i$ requires knowing only the six coefficients $a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}, a_{0,2}, a_{2,0}$ in Equation 5.2 for each point p_i, and these coefficients determine the principal curvatures, principal directions, and surface normal $n_i^*(0,0)$.

With these parameters, one can derive closed-form expressions for the center c, axis of revolution u, major radius R, minor radius r, and principal directions, and surface normal that determine the SDF $\phi_{\mathbb{T}}(x)$ of a torus $\mathbb{T}$,

$$\phi_{\mathbb{T}}(x) = \|d\| - r, \quad d \coloneqq (\|(x-c)\times u\| - R, \langle x-c, u\rangle)\,. \tag{5.3}$$

Equation 5.3 is essentially obtained by applying the SDF to a circle, twice [Quilez 2025]. We take the major radius R to always be greater than the minor radius r; in particular, we define $\kappa_{\max}, \kappa_{\min}$ to the signed principal curvatures with the greater and lesser magnitudes corresponding to r and R, respectively.

There are two sign combinations of the principal curvatures $\kappa_{\max}, \kappa_{\min}$ corresponding to a positive SDF, and two combinations for negative (Figure 5.4, *right*). The final SDF of torus $\mathbb{T}_i$, to be blended using Equation 5.1, is

$$g_i(x) = \operatorname{sign}(\mathbb{T}_i)\,\phi_{\mathbb{T}_i}. \tag{5.4}$$

where $\operatorname{sign}(\mathbb{T}_i)$ is the sign of the torus's SDF.

5.1.2 Learning per-neighborhood coefficients

For signed distance inference, we train a small neural network that learns to predict the six polynomial coefficients used to determine the torus parameters at each point p_i. The choice to use a (small) neural network here is a careful and deliberate one: see Section 5.5 for a discussion of the algorithm's design choices. Importantly, the network is purely local, and needs to be pretrained only *once* before being applied to *any* neighborhood of *any* possible surface — in contrast to *e.g.* neural field methods that need to train a separate network for each input shape.

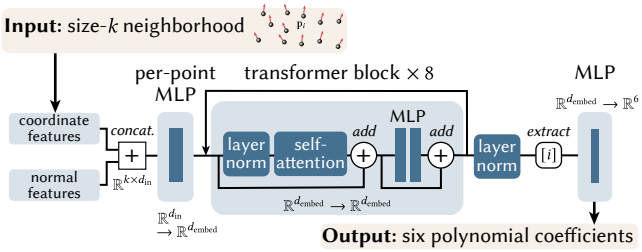


Fig. 5.6: A neural network applied to each size-k neighborhood, using a series of transformer blocks that learn which points in each neighborhood are important for locally fitting a torus.

As input, the network takes in a point set $\mathcal{N}_k(p_i)$ comprising a point p_i and its k nearest neighbors. As output, the network returns the coefficients $a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}, a_{0,2}, a_{2,0}$. We use the architecture shown in Figure 5.6, using transformer blocks using attention [Vaswani et al. 2017] to learn which points in the neighborhood are most important for fitting a local surface.

We train this network with a per-neighborhood loss function that penalizes the differences

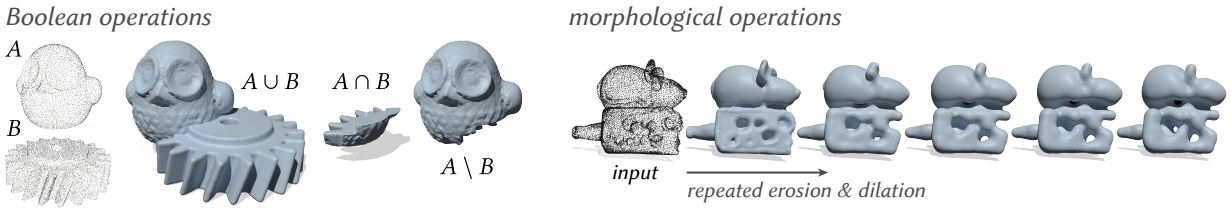


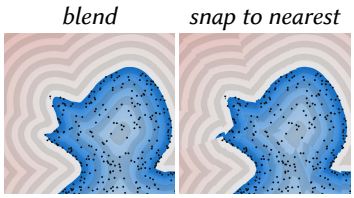
Fig. 5.7: *Left:* Boolean operations are applied to two point clouds. *Right:* Morphological operations can be applied directly to the surfaces underlying point clouds, without meshing.

between the SDF ϕ predicted using Equation 5.1 and ground-truth,

$$\begin{aligned} \mathcal{L} &:= \mathcal{L}_{\text{distance}} + \mathcal{L}_{\text{eikonal}}, \\ \text{where } \mathcal{L}_{\text{distance}} &= \frac{1}{Q}\sum_{j=1}^{Q} \left|\phi\left(q_j\right) - \phi_{\text{true}}\left(q_j\right)\right|, \\ \mathcal{L}_{\text{eikonal}} &= \frac{1}{Q}\sum_{j=1}^{Q} \left|1 - \|\nabla\phi\left(q_j\right)\|\right|. \end{aligned} \tag{5.5}$$

The eikonal loss implicitly encourages neighborhoods to blend together smoothly. As training data, we use a variety of geometries including CAD models [Koch et al. 2019] and procedurally-generated shapes, and sample them into fixed-size point clouds. In total, we used about 40 million training examples, and took about 45 hours to train on a single NVIDIA RTX 3090 GPU.

5.1.3 Evaluating signed distance



Once tori have been fitted, the global SDF $\phi(x)$ can be computed with Equation 5.1 using the fitted g_i, at arbitary query points x. Equation 5.1 technically requires area weights, as a discretization of the surface integral in Equation 3.16. However, omitting area weights improves efficency, and we hypothesize that we do not observe many ill effects because we fit surfaces with area, rather than singular kernels[1]. We also remark that blending torus SDFs via Equation 5.6 is important, since simply using the torus of the nearest point results in discontinuities (inset).

Setting λ. With a good local reconstruction in place via the fitted tori, we simply need to set a large λ in Equation 5.1 to get good distance. How large we can make λ depends on machine precision, and a simple shifting of the exponents buys precision without changing the result [Blanchard et al. 2020]. Empirically, we find good behavior using the following shifted formula

[1]Incorporating separately-optimizable confidence weights may be interesting future work.

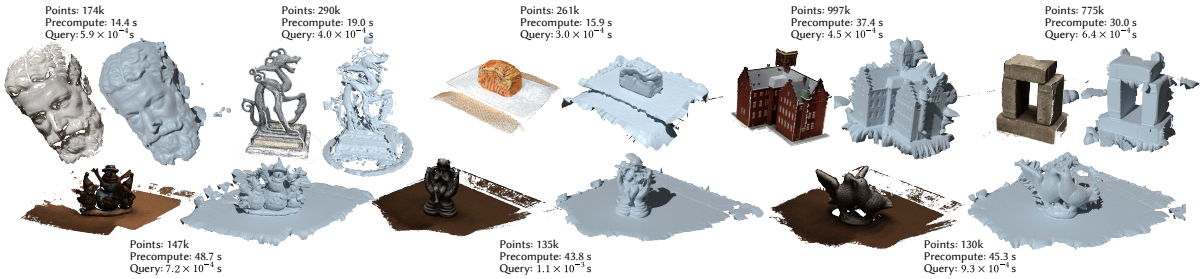


Fig. 5.9: Some reconstructions produced by PAT of dense point cloud outputs from COLMAP, without any point cloud preprocessing. Our method yields reasonable reconstructions despite not being trained on any data with noise or outliers.

instead of Equation 5.1:

$$\phi(x) = \frac{\sum_{i=1}^{|P|} g_i(x) \exp\left(-\lambda_x \left(\|x - p_i\| - \sigma_x\right)\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda_x \left(\|x - p_i\| - \sigma_x\right)\right)}. \tag{5.6}$$

We set the exponential shift and screening for each x as

$$\begin{aligned} \sigma_x &:= \frac{1}{2} \max\left(\{\|x - p_i\| \mid p_i \in P\}\right), \\ \lambda_x &:= \frac{C}{\max\left(\{\|x - p_i\| - \sigma_x \mid p_i \in P\}\right)} = \frac{C}{\sigma_x}, \end{aligned} \tag{5.7}$$

where C is a constant defined such that $\exp(-C)$ does not exceed machine precision; the maximum value of C is about 87 when using single-precision floating point. We use $C = 64$. Finally, we sum only over points within a fixed radius R_{eval} of each evaluation point x, falling back to summing over the 32 nearest points if no points lie within R_{eval}. We select R_{eval} by keeping λ constant for a given point cloud and adjust R_{eval} accordingly. It is possible to instead evaluate using a fixed-size evaluation neighborhood, and to also further tune λ to balance detail of reconstruction vs. noise or sparsity, though we have not yet explored these avenues.

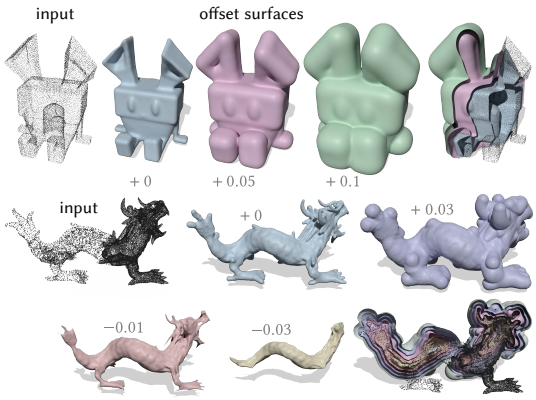


Fig. 5.8: Offset surfaces to point clouds, including to an unevenly sampled point cloud (*bottom*).

5.2 Results of PAT

PAT extends classic SDF operations to point cloud data: the method plugs directly into existing algorithms that rely on pointwise signed distance queries, allowing point clouds to be used directly in geometry routines. For example, we can apply Boolean operations to sparse point clouds (Figure 5.7, *left*), or apply the classical morphological operations of erosion and dilation to "deepen" the concave features of the surface underlying a point cloud (Figure 5.7, *right*). In Figure 5.8, we contour offset surfaces, including of a point cloud of varying density. As PAT produces signed distance fields, point cloud surfaces and their offsets can be visualized directly using sphere tracing [Hart 1996]. Figure 5.3 shows a few examples visualizing different offset surfaces in a shader.

Fig. 5.10: General implicit functions can be turned into signed distance functions by applying our method to a sampling of their zero level set. Here we use the method of Sharp and Jacobson [2022] to sample neural implicits by casting rays. Query times are measured without GPU acceleration.

PAT can robustly infer SDFs directly from acquired point clouds; Figure 5.9 shows how the proposed method can be applied directly from the output of computer vision methods that estimate 3D point clouds from unposed photos. These results are obtained without point cloud preprocessing or data-specific tuning. Beyond pure point cloud output, Figure 5.11 visualizes signed distance from Gaussian splat objects, an emerging 3D representation [Kerbl et al. 2023], and Figure 5.10 shows how more general implicit functions, such as existing neural implicits, can easily be redistanced.

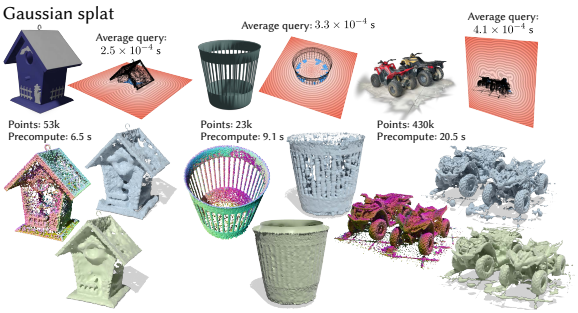


Fig. 5.11: These 3D Gaussians have inconsistent geometry and orientations; nevertheless, our SDF behaves well in the far-field. Query times are measured without GPU acceleration. The results of SPSR [Kazhdan and Hoppe 2013] are shown for comparison (green).

5.3 Accuracy and performance of PAT

This section establishes PAT's accuracy and performance relative to common signed distance and reconstruction methods, including the signed heat method (Chapter 4). For even more discussion comparing PAT's accuracy to that of the other methods presented in this thesis, see Chapter 7.

Accuracy. Figure 5.12 shows accuracy statistics on four datasets of size-512 point clouds: 100 watertight CAD models from the ABC dataset and 100 procedurally-generated shapes not seen during training, 45 surface meshes from the Thingi10k dataset [Zhou and Jacobson 2016], and 70 instances of analytically-defined SDFs. All meshes are centered so their centroids are

at the origin, and scaled to lie within the cube $[-1, 1]^3$. In addition to the signed heat method (SHM), we compare with two baselines: signed Hopf-Cole (a.k.a. signed LogSumExp, or screened winding number)

$$\begin{aligned} \phi_{\text{SHC}}(x) &= \text{sign}_w(x) \left(-\frac{1}{\lambda} \log |w(x)| + \sigma_x \right), \\ w(x) &:= \sum_{i=1}^{|P|} A_i \frac{(\lambda \|x - p_i\| + 1) \langle x - p_i, n_i \rangle}{2\pi \|x - p_i\|^3} \exp\left(-\lambda \left(\|x - p_i\| - \sigma_x\right)\right) \end{aligned} \tag{5.8}$$

as an instance of Equation 3.15; and the smoothed signed planar distance

$$\phi_{\text{SSPD}}(x) = \frac{\sum_{i=1}^{|P|} A_i \langle x - p_i, n_i \rangle \exp\left(-\lambda \left(\|x - p_i\| - \sigma_x\right)\right)}{\sum_{i=1}^{|P|} A_i \exp\left(-\lambda \left(\|x - p_i\| - \sigma_x\right)\right)} \tag{5.9}$$

as a baseline instance of Equation 3.16.

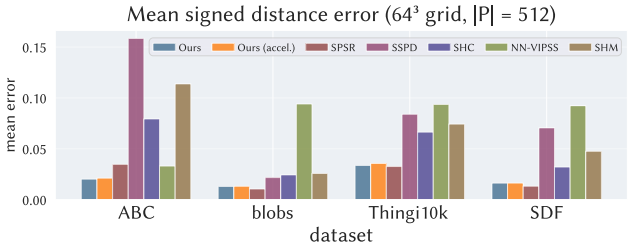


Fig. 5.12: On average, PAT has better signed distance accuracy than other convolutional formulas, and even a global method (SHM) for sparse point cloud data. It is also less prone to catastrophic error (Figure 5.13)

We also compare against an end-to-end pipeline of reconstructing the point cloud, meshing the surface, and computing signed distance to the meshed surface. For signed distance, we compute unsigned distance using `FCPW` [Sawhney 2021] and sign with fast winding number [Barill et al. 2018]. We compare using two different reconstruction methods: screened Poisson surface reconstruction (SPSR) [Kazhdan and Hoppe 2013] (using `Open3D`'s Python wrapper [Zhou et al. 2018]), and NN-VIPSS using input normals [Xia and Ju 2025]. To reflect typical usage, we preserve the contouring algorithms in the original codebases: SPSR uses an octree-based contouring method [Kazhdan et al. 2007], and NN-VIPSS uses an adaptive tetrahedral method that uses both function and gradient values [Ju et al. 2024]. We limit the maximum depth of SPSR's octree to correspond to the resolution of the grid used for evaluating and contouring the results of other methods.

Compared to the other point-based approaches and SHM, our method has consistently lower error across the four datasets (Figure 5.12). The SPSR-mesh-SDF pipeline has slightly lower error on three of four datasets, though we have lower error on the ABC dataset, probably because our network was mostly trained on other ABC shapes. We were surprised that SHM, a global method, had relatively high SDF error. We hypothesize that SHM produces poor or overly smooth reconstructions when data is sparse, and might benefit from higher grid resolution (at the cost of significantly more computation time), or using a tet mesh discretization with exact zero-set constraints at input points (Figure 4.17).

In Figure 5.13, we visualize the corresponding reconstructions of several shapes. Qualitatively, our method gives better reconstructions on sparse point clouds than other point-based methods and SHM. Signed Hopf-Cole distance and smoothed signed planar distance are prone to failing

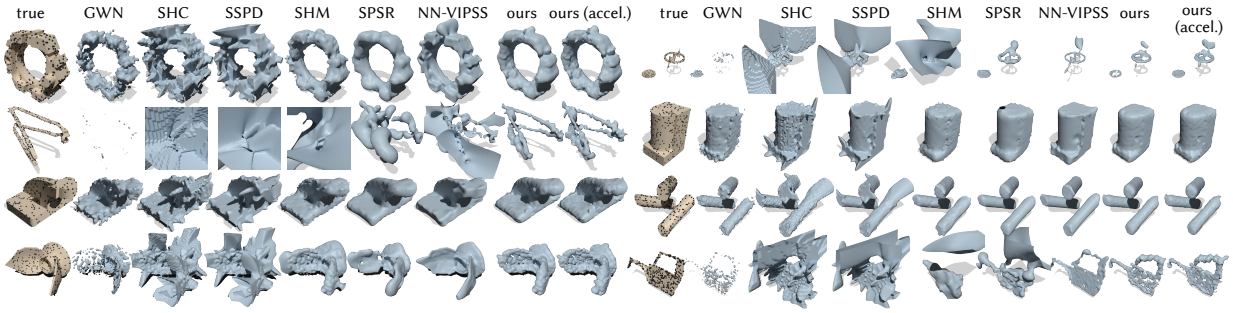


Fig. 5.13: PAT tends to give better results on sparse point clouds, probably because our network was trained on relatively sparse point clouds (2048 points); here we use point clouds with 512 points. On some shapes, we observed better results using smaller λ than provided by the simple heuristic presented in Section 5.1.3, suggesting further tuning may be possible. NN-VIPSS can yield higher-quality reconstructions, especially when using ground-truth normals, though at a higher cost (Figure 5.14)

catastrophically, owing to their brittle tangent-plane-based sign estimation. Even so, our method might still benefit from further tuning and exploration: anecdotally we observed better results on these very sparse point clouds using smaller λ, perhaps suggesting a more sophisticated heuristic than the one in Section 5.1.3. SHM can produce well-behaved reconstructions, at the cost of sometimes over-smoothing surfaces when data is sparse, but can also fail badly if data is too sparse; SPSR has similar downsides. NN-VIPSS, in contrast to the other methods, extracts surfaces using an adaptive threshold-based method that uses both function values and gradients; it generally yields higher-quality reconstructions especially when using input normals, with only a few catastrophic failures, though at a higher cost than our method.

Performance. During the precompute stage of PAT, the performance bottleneck is neural network evaluations for surface coefficient inference. Figure 5.14 shows timings, which were measured on a MacBook Pro laptop (M3 Max chip, 16-core CPU, 64 GB of RAM). For comparison, we evaluate fast winding number on the same data using the implementation in `libigl`, which uses parallelized code and hierarchical acceleration. Fast winding number averages 3.9×10^{-6} seconds per query, whereas PAT averages 7.4×10^{-5} seconds per query on the MacBook. On a Linux desktop with a 64-core AMD Ryzen Threadripper 3990X CPU and NVIDIA RTX 3090 GPU, precompute takes about 40 seconds for a point cloud with one million points. Without a dedicated GPU, precompute takes about a minute for point clouds with 50k points, and eight minutes for point clouds with one million points on the MacBook.

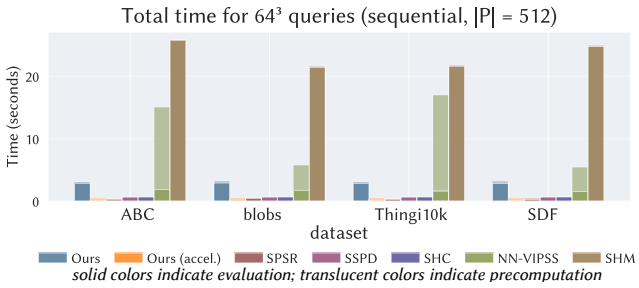


Fig. 5.14: PAT is almost as fast as naive convolutional methods, and orders of magnitude faster than the signed heat method (SHM) while providing good signed distance quality on our datasets to sparse point clouds. For SPSR and NN-VIPSS, "precomputation" refers to reconstruction and meshing, while "evaluation" refers to distance evaluation to the meshed surface. Times correspond to sequential queries.

Figure 5.15 shows PAT's performance as point cloud size increases, using a 3D scan of *David* from the Digital Michelangelo Project [Levoy et al. 2000]. Both precompute time and query time grow linearly with point cloud size. The largest point cloud has 29 million points: there, precompute takes about 12.5 minutes, and a single SDF evaluation takes about four milliseconds. In contrast, the signed heat method can struggle on large, intricate scenes in $\mathbb{R}^3$ where the number of DOFs naively grows cubically with volume. On the 29-million-points David example, the publicly-available signed heat method implementation [Feng 2025, v1.1.0] throws an error after two hours because the positive definite linear solver received an indefinite system. It is possible to combat indefiniteness by mollifying the linear system or adjusting the linear solver, and to address the naive cubic scaling by meshing only where one wants to evaluate signed distance (for example, in a narrow band around the point cloud), but these workarounds add another layer of complexity relative to PAT.

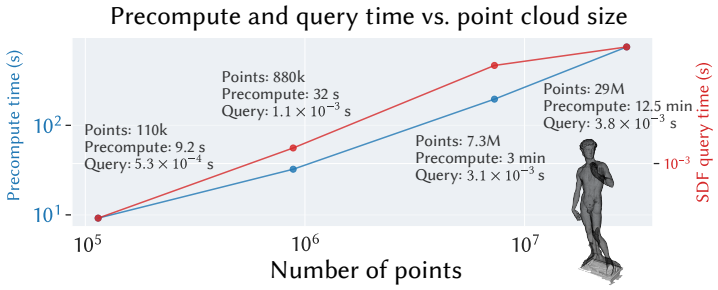


Fig. 5.15: Precompute and query times for point clouds of size 110k, 880k, 7.3M, and 29M. Even for point clouds reaching tens of millions, each query of PAT's SDF remains on the order of milliseconds.

5.4 Limitations of PAT

The points as tori method shows that using a neural network to learn local kernel parameters is a powerful approach for signed distance estimation, yielding better results than fast methods based on classical closed-form kernels. However, there is room for improvement.

5.4.1 Robustness to large amounts of noise or outliers

PAT is less robust relative to the signed heat method for point clouds with significant outliers or noise (Chapter 7). On the one hand, in principle it is straightforward to simply generate a broader range of point distributions for the network to train on; in this problem, one is limited primarily by the representational power of the network. On the other hand, we found that our network had difficulty learning when trained on point cloud neighborhoods with noise (Section 5.5.1). Future approaches for making PAT's predictions more robust might entail improved feature tuning, incorporating more global context about the whole point cloud, a hierarchical approach, or simply more extensive training. Recent neural reconstruction methods like NKSR [Huang et al. 2023] address the challenges of

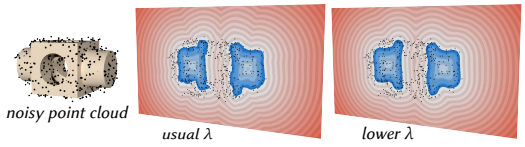


Fig. 5.16: Though we use a simple, fixed heuristic, further tuning λ can lead to better reconstruction even using the same fitted tori. For example, if noise is high, setting λ lower leads to smoother reconstructions. Alternatively, if noise is low, setting λ higher might yield better detail recovery.

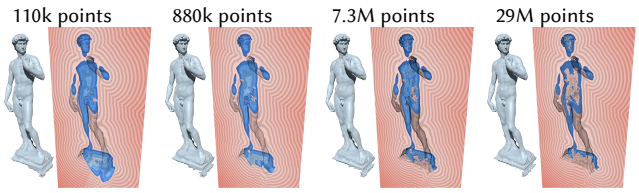
robust reconstruction with data preprocessing and voxel size tuning, though ideally one should minimize the amount of manual cleanup and human decision. Recent point-based methods also improve initial reconstructions through per-example image-based optimization [Chen et al. 2024b; Huang et al. 2024], a topic natural for future work.

Our method in principle also allows scaling of λ to balance reconstruction detail vs. noise; we currently use a simple, fixed heuristic to set λ regardless of local sampling density or noise, and leave sophisticated tuning to future work (Figure 5.16). Future work on optimally and efficiently tuning λ seems especially promising given that bandwidth tuning is common to kernel-based reconstruction methods such as NKSR [Huang et al. 2023] and Poisson surface reconstruction [Kazhdan et al. 2006].

5.4.2 Non-conservative distance fields for perfect geometry

In the case of perfect, closed geometry without noise, all self-normalized convolutional distance formulas, including ours, are not conservative: they give an overestimate rather than an underestimate of the magnitude of the true distance for any finite λ (see Section 2.2). In practice, the error goes away with high λ, and potential overestimation is likely insignificant compared to the uncertainty inherent to point cloud data; it is also hard to define what it means for the distance approximation to be an overestimate since the true shape is unknown.

5.4.3 Out-of-distribution behavior

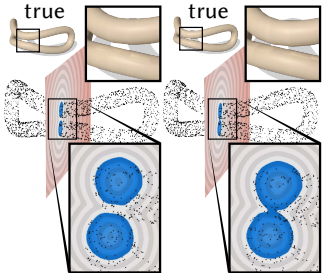


Like all data-driven approaches, our method can yield unexpected results when given point clouds that deviate significantly from the training data. A prominent example we found was the points as tori method's relative sensitivity to sampling density (inset) compared to *e.g.* the signed heat method. The figure visualizes SDF slices and reconstructions of increasingly dense samplings of the shape used in Figure 5.15. Though non-negative level sets always behave well, the interior of the SDF can paradoxically get worse as the sampling gets denser. We hypothesize that as neighborhoods of size $k = 64$ get smaller with increasing sampling density, the network can sometimes simply fit compact tori. Future work might tweak our network by introducing additional input features or geometric regularizations, subsample point clouds, implement a (learned) hierarchical framework or feature-size estimation, or simply train on point clouds of varying densities — we only train on point clouds within a very small range of densities (all point clouds used for training had 2048 points within a bounding box of $[-1, 1]^3$).

5.4.4 Enforcing constraints

Using the convolutional formula in Equation 3.16 means the reconstructed surface is always C^∞. Moreover, around locally planar or cylindrical surfaces, we fit tori with very large radii rather

than fit a true plane or cylinder, though we did not encounter numerical trouble approximating cylindrical (Figures 5.2, 5.3, 5.8 dragon whiskers) or planar features (Figure 5.9). Future work can try extracting true sharp features, perhaps by fitting extra types of primitives.



Because of the huge amount of work on meshless methods across statistics, geometry processing, engineering, and simulation, there might also be variants of our problem model tailored for particular tasks. For example, Gotsman and Keren [1998] and Keren and Gotsman [1999] solve for algebraic curves guaranteed to satisfy certain topological properties. One might also try to explicitly enforce physical constraints tied to knowledge about the point cloud acquisition process: for example, if one knows that the point cloud came from a noise-less sensor, then the zero level set of the final SDF cannot enclose other points. The method also cannot guarantee exact interpolation. Exact interpolation is often not desired, since real point clouds often have non-zero noise, but nonetheless could be a useful option that may avoid unintentional merging (inset).

5.5 Design choices behind PAT

This section explains key design choices behind the points as tori method.

5.5.1 Failed approaches

We tried many alternative approaches that didn't work. One idea was to use established tools from manifold learning, for example diffusion maps [Coifman and Lafon 2006], to learn spatially-varying, possibly anisotropic diffusion parameters associated with kernels locally aligned with the geometry. Unfortunately, this process proved brittle around edges and corners, where it is easy to incorrectly learn anisotropy aligned with only one side of the sharp feature. We tried using the quadric error metric of Garland and Heckbert [1997] instead of learning arbitrary anisotropic kernels, but the quadric error metric was extremely sensitive to the chosen neighborhood size. We tried hierarchical approaches, but choosing parameters optimal for torus-based signed distance reconstruction was tricky.

We also tried approaches that directly optimized for torus parameters, but found that the problem was ill-conditioned. Then, similar to our proposed method, we tried a neural network that for each point learned the optimal per-neighbor weights with which to solve a weighted least-squares minimization for the best-fit quadratic surface in the neighborhood. Learning per-neighbor weights is more stable, and perhaps better-suited for transformers, than directly learning torus parameters. However, under this model it was difficult to provide a good initialization of the neural network. Ultimately, we decided to learn only the six polynomial coefficients needed to derive the shift and curvatures used to fit a torus, which both enabled good network initialization, and bypassed the need to solve a weighted least squares problem in the forward pass (for example, as done by Ben-Shabat and Gould [2020]).

We additionally experimented with learning additional per-neighborhood scaling factors for λ, and learning using k-nearest neighbors-based acceleration for SDF evaluations, but found that both approaches did not converge. We also found that our network could not learn well on point clouds with any noise, which we hypothesize is because curvature estimation is a relatively ill-conditioned problem.

5.5.2 Why learning?

Inferring the true surface, even locally within a small neighborhood, is a difficult problem. In fact, simply choosing what this local neighborhood should be is difficult, and classic point set methods are extremely dependent on domain-specific heuristics for neighborhood size and shape. For example, a classic paradigm from the point set literature is to use an exponential kernel (usually Gaussians) to determine a "soft" neighborhood size, an approach we find incredibly brittle (Figure 5.17): there is no good kernel bandwidth, and hence neighborhood size, that yields good results, and so the downstream SDF breaks down.

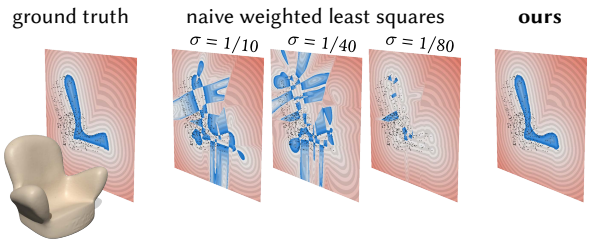


Fig. 5.17: Classic point set methods often use a weighted least squares approach that solves for a best-fit quadratic polynomial around each point p_i, using Gaussian weights $\exp\left(-\|p_i - p_j\|^2/\sigma^2\right)$ for each neighbor p_j. In this example, we additionally fit tori to the fitted polynomials to get signed distance. Without further tuning, this process is incredibly brittle, with no values of σ yielding accurate results.

Because of how difficult even local surface fitting can be, there has been a huge amount of literature proposing endless variations of similar kernel-based methods for reconstruction (keywords: moving least squares surfaces, partition-of-unity surfaces, radial basis functions). As described above, it is easy to completely mis-classify features without tuning the method just right, and more sophisticated statistical methods come at a cost. Furthermore, for any chosen method, one can probably think of a class of data for which the assumed model breaks down. Hence rather than manually tweak the parameters of a single, particular surface-fitting model, we chose to use fixed-size neighborhoods but with a data-driven approach that can fit to an extremely large set of examples — far more examples, and with a far more sophisticated data-fitting model, than we could ever craft purely by hand. Still, the network is fairly lightweight by modern standards: pre-training took about 45 hours on a single GPU (NVIDIA RTX 3090), and the network takes about 6.4MB on disk. We can design and train this network easily, precisely because we've broken down the problem into one that is dominated by local surface behavior.

Non-neural methods like the signed heat method tend to have less failure when given point clouds that are severely "out of distribution" (meaning input that is significantly different than data seen in training), but you pay more with each application of the method. In this method, you pay more up front (via pretraining), and get much better performance on in-distribution data.

5.5.3 Why not end-to-end learning?

In contrast to other learning-based approaches that try to directly learn SDFs from entire point clouds, we intentionally invoke the minimum amount of learning needed — relying on classical results for convergence to true signed distance, and using learning only for the one component with no easy analytical solution.

Recently, neural fields have become popular as implicit shape representations. Many works attempt to train such fields to be SDFs, which is challenging because the eikonal equation $\|\nabla\phi\|^2 = 1$ is nonlinear and has many local minima, and because the implicit function values depend nonlinearly on neural network parameters. As a result, neural fields are not true SDFs, but simply SDF-like signed implicit functions that aid reconstruction of the zero level set [Calakli and Taubin 2011; Atzmon and Lipman 2019; Sitzmann et al. 2020; Chibane et al. 2020; Ma et al. 2021; Pumarola et al. 2022]. More recent works therefore introduce additional regularizations using various distance properties, to encourage SDF-like behavior [Gropp et al. 2020; Zhang et al. 2022; Park et al. 2023; Ben-Shabat et al. 2022; Yang et al. 2023; Wang et al. 2023; Coiffier and Béthune 2024] or build on the heat method [Weidemaier et al. [n. d.]]. Some methods use the signed Hopf-Cole transformation mentioned in Section 2.2 [Lipman 2021; Wang et al. 2025a], though they cannot provide signed distance to point clouds.

Significantly, these neural SDF-based shape representations work by fitting a neural network to a given shape. In other words, they must train a new network for every shape. Other neural methods that have adopted interpolation-based, local-to-global approaches for constructing implicit shape representations [Yariv et al. 2024; Zhang and Wonka 2025; Lin et al. 2025] still only fit networks to *existing* shapes, and do not infer signed distance of imperfect geometry; they also require fitting a new network for each new input shape. The points as tori method completely sidesteps the difficulties of fitting neural fields to the eikonal equation — and the issues of expensive per-shape training, expensive inference, and limited scalability — by formulating point cloud reconstruction as a purely local problem, before applying a simple analytical formula to produce distance (Equation 3.16). Our network shares weights across point neighborhoods, across all shapes — so it can be trained once, and thereafter be applied to other input data without further GPU training. Since the network can be smaller, inference time can also be faster.

5.5.4 Why tori?

Numerous past works reconstruct point clouds as (piecewise) smooth surfaces by fitting geometric proxies such as planes, cylinders, spheres, quadric surfaces, splines, and polynomial patches [Faugeras 1983; Besl and Jain 1988; Cao et al. 1994; Kaveti et al. 1996; Yang and Lee 1999; Cohen-Steiner et al. 2004; Cazals and Pouget 2005; Wu and Kobbelt 2005; Simari and Singh 2005; Yan et al. 2006; Attene et al. 2006; Xia and Ju 2025]. Modern approaches use learning-based methods [Groueix et al. 2018; Erler et al. 2020]. This process is called *shape approximation* or *jet-fitting*, and is often used for shape segmentation, reconstruction, simplification, or derivative estimation.

In our setting, we need geometric proxies that not only are expressive enough to accurately reconstruct a shape, but also admit efficient distance queries. Quadratic surfaces are expressive, but do not in general admit closed-form distance expressions. Some works build efficient approximations of distance to polynomial curves and surfaces [Taubin 1993; Lennerz and Schömer 2002; Lott III 2014] and quadric surfaces [Martínez and Estrada-Sarlabous 2003; Sappa and Rouhani 2009; Lopes et al. 2010], but their procedures remain relatively expensive and unreliable to apply in our context: each evaluation of our SDF requires a distance query to every geometric proxy in the scene. A few works consider fitting tori [Lukács et al. 1998; Liu et al. 2009; Eberly 2020]; we take advantage of the fact that tori have closed-form inexpensive SDFs to compute our global SDF approximation, and learn the parameters used for data-fitting rather than rely on manual or heuristic-based selection.

5.5.5 An appeal for small, judicious use of learning

The points as tori algorithm boils down to a deceptively simple idea of interpolating local estimates, and relying on the structure of a classical formula to obtain accurate signed distance. As described above, the core difficulty arises in deriving the local estimates, for which we decided to use a neural network; this network is injected precisely and only where it is needed.

Recently, trends in machine learning, especially in the "AI for Science" space, have made an appeal for generality, reusability, and scalability of neural models: for example, a PDE solver should be able to generalize to new domains or boundary conditions without hours of re-training. Some have begun calling such networks "foundation models". In our problem of signed distance, we achieved these three properties by relying on an analytical formula for signed distance: the data-driven component is local and hence can be re-used across multiple problems, and the resulting algorithm is much more scalable than past methods. Some recent research has followed a similar local/global problem breakdown [Quackenbush and Atzberger 2025; Choi et al. 2025]; for example, Choi et al. [2025] advocates for replacing traditional polynomial bases in FEM with richer, data-driven bases, while leaving global assembly unchanged. The higher-level takeaway is that one may get mileage out of using data-driven methods to gain representational power, but otherwise rely on classical methods to provide structure, and convergence, and stability.

5.5.6 Future extensions of PAT

Accelerating pointwise signed distance. The points as tori method described in Chapter 5 is not yet "instant", since there is a non-negligible precomputation cost for each new point cloud. Although its implementation already uses, for example, the fairly performant `MultiHeadAttention` function of `Flax NNX`, faster implementations of the attention mechanism have been an active area of research [Dao et al. 2022; Dao 2024]. The method's neural network architecture also has not been extensively engineered, owing to my relative ignorance (at the time) of practical machine learning; I suspect that performance can be dramatically improved with better choice or normalization of input features, fewer attention layers, or perhaps a different architecture entirely. If used for optimization tasks, tori can perhaps be updated

using simple gradient-based updates rather than forward passes of the neural network. It also may still be possible to develop an effective non-neural approach to fitting tori — for example, building on top of existing interpolants like *natural neighbor Duchon* [Xia and Ju 2025] — that bypasses the need for a neural network entirely.

Torus signed distance as a differentiable shape representation. The points as tori method essentially provides a compressed representation of signed distance fields by encoding shapes as a collection of point samples and associated tori. This representation is not only analytical, but also fully differentiable, allowing point clouds to be used as a sparse representation of a global signed distance field used for *e.g.* shape optimization and other inverse design tasks. Compared to general signed implicit representations, signed distance functions have more meaningful gradients, especially far away from the surface. Like forward evaluation, computation of derivatives is local, output-sensitive, and trivially parallelizable; ideally, these properties imply that one can avoid differentiating through global solves or neural networks, bringing large speedups within optimization loop.

The torus-based signed distance representation can also be developed further as a data structure. For example, the current algorithm uses a torus for each point in a point cloud, but one might achieve a more compressed representation by using coarse-to-fine approach to torus-fitting, or by adapting *quadric error simplification* [Garland and Heckbert 1997] to merge tori after fitting. One may also want to explore how to best insert additional points, for example during an optimization task.

Chapter 6

Generalized inside/outside

When discussing generalized signed distance in the previous two chapters, we implicitly assumed that there existed one true signed distance function to approximate. However, on some surfaces there can exist curves — even closed curves — that do not have a well-defined inside or outside, complicating the definition of concepts like signed distance (Section 2.3.2).

In this chapter, we use *winding numbers* as a framework for addressing the question of the inside/outside of curves on surfaces. Ordinary winding number represents the number of times a curve or surface wraps around a given point (Section 2.3.3), and winding numbers are already a basic component of geometric algorithms such as point-in-polygon tests; their generalization to data with noise or topological errors has proven valuable for geometry processing tasks ranging from surface reconstruction to mesh booleans. However, standard definitions on $\mathbb{R}^d$ do not immediately apply on surfaces, where not all curves bound regions. Here, we develop a meaningful generalization, starting with the well-known relationship between winding numbers and harmonic functions. The resulting algorithm, which we call *surface winding numbers (SWN)*, yields

1. a closed, completed version of the input curves;
2. integer labels for regions that are meaningfully bounded by these curves;
3. the complementary curves that do not bound any region.

Because we work with smooth functions defined globally on the domain, our algorithm is much more robust than if we had tried to work directly with sparse, singular curves. The algorithm is guaranteed to work if the input is "perfect" (no noise, gaps, etc.), and otherwise degrades gracefully in the presence of imperfections. The main computational cost is solving a standard Poisson equation, or for surfaces with nontrivial topology, a sparse linear program.

6.1 Surface winding numbers

The key idea of SWN is to turn the difficult problem of determining inside/outside of an unstructured collection of (possibly imperfect) *curves*, into an easier but equivalent problem about

vector fields. We start from the established concept of winding number.

Ordinary winding number is often presented as an integral: for example, in differential geometry, the winding number $w_\Gamma(x)$ of a curve Γ at a point $x \in \mathbb{R}^2$ is usually defined as

$$w_\Gamma(x) = \frac{1}{2\pi} \int_\Gamma \frac{\langle n_\Gamma(y), y - x \rangle}{\|x - y\|^2} \mathrm{d}y, \tag{6.1}$$

or equivalently in complex analysis as the complex contour integral

$$w_\Gamma^{\mathbb{C}}(p) = \frac{1}{2\pi i} \int_\Gamma \frac{1}{z - p} \mathrm{d}z, \quad p \in \mathbb{C}.$$

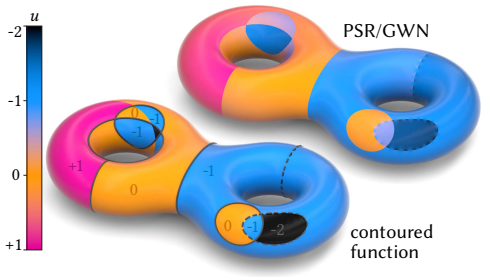


Fig. 6.1: On surfaces, contouring the solution to Equation 6.2, equivalent to generalized winding numbers (GWN) and unregularized Poisson surface reconstruction (PSR), can yield regions that do not follow the input curves, and/or jump across nonbounding curves.

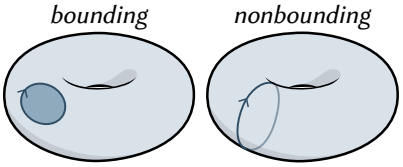
One could, for example, extend these integral definitions to surfaces that admit planar or spherical parameterizations. But in general, these contour integrals can't easily be computed on surfaces. There are several other characterizations of winding numbers (see Feng et al. [2023, supplemental] for an in-depth discussion), but they also fail to generalize on surfaces.

One insight is that the winding number is a special type of *harmonic function*, meaning a function whose Laplacian vanishes. In particular, winding number is a *jump harmonic function* (Section 2.3.3), and can be computed using the jump Laplace equation

$$\begin{aligned} \Delta u &= 0, && \text{on } M \setminus \Gamma, \\ u^+ - u^- &= 1, && \text{on } \Gamma, \\ \partial u^+ / \partial n &= \partial u^- / \partial n, && \text{on } \Gamma. \end{aligned} \tag{6.2}$$

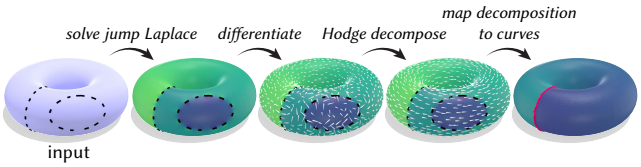
More generally, u can jump by a different integer value across Γ if Γ wraps onto itself multiple times. The jump Laplace equation is the partial differential equation (PDE) version of *solid angle*, a generalization of winding number to open curves Γ. If M is *simply-connected*, then we can simply solve Equation 6.2 for $u(x)$. If Γ is closed, this case is analogous to ordinary winding number in the plane, which yields a piecewise constant, integer-valued function. If Γ is not closed, then $u(x)$ yields a "soft" real-valued (rather than integer-valued) indicator function, analogous to solid angle.

If M is not simply-connected, however, the input curves Γ may have nonbounding components (Section 2.3.2) such that the solution u to Equation 6.2 will not look like a region labeling (Figure 6.1). Formally, (non)bounding loops are loops that are *(non-)nullhomologous*, meaning (non)congruent to zero in the first homology group $H_1(M) = \ker(\partial_1)/\mathrm{im}(\partial_2)$ of the surface M. In words, nonbounding loops are closed curves which don't bound regions, and so do not have a well-defined inside or outside (inset). In principle, to identify nonbounding



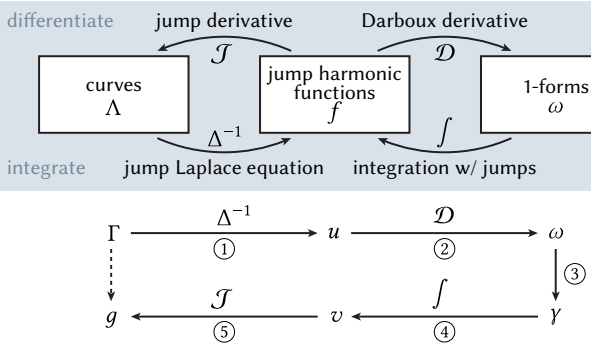
A *separating cycle* is noncontractible but nullhomologous, and hence bounding.

curve components, we just need to determine the homology class of the curve (meaning the nonbounding components of the curve which belong to the first homology group of M). But homology only directly applies to closed curves, so we use *de Rham cohomology* to instead process functions dual to curves.



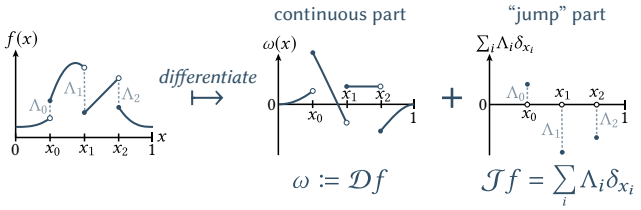
At a high level, we start by taking an appropriate derivative of our jump harmonic function. Once in the "gradient domain", our problem of decomposing a curve into its bounding and nonbounding components — meaning components that can and cannot be explained as region boundaries — becomes the more well-studied problem of decomposing a vector field into components that can and cannot be explained by a potential, which we can solve using the classical tool of Hodge decomposition (Section 2.2). Once we have the decomposition of the vector field, we transform it into a decomposition of the curve.

6.1.1 Differentiating and integrating jump harmonic functions



Solving the jump Laplace equation (Equation 6.2) is Step ①. If M is not simply-connected, we need to take additional steps. The key insight here is that jump harmonic functions form the "bridge" through which we can transform curves into functions, and vice versa. In particular, jump harmonic functions can both encode curves (where the function jumps in value) as well as a curve's homology class (through its derivative).

The SWN algorithm amounts to a round-trip around the inset diagram, where we first translate the input curve Γ into a jump harmonic function (Equation 6.2), then differentiate this function to obtain a differential 1-form. We then translate the resulting 1-form back into a curve and jump harmonic function, corresponding to the curve decomposition and final winding number function, respectively. Performing these translations amounts to using appropriate notions of differentiation and integration.



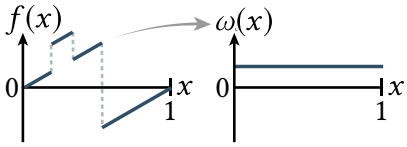
At a high level, these notions of differentiation and integration correspond to their usual notions from ordinary calculus, except we take extra care when dealing with the discontinuities of jump harmonic functions. To demonstrate, we first consider a periodic 1-dimensional function $f(x)$ defined on the unit interval $[0, 1]$. The distributional

derivative of any such function can be expressed as

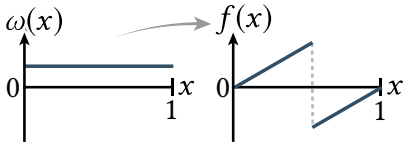
$$f'(x) = \omega(x) + \sum_i \Lambda_i \delta_{x_i}$$

where ω is a periodic piecewise smooth function, and Λ_i is the size of the jump at x_i. The gradient of such a function $f(x)$ with respect to x yields a continuous part $\omega := \mathcal{D}f$, obtained from considering each continuous interval of $f(x)$, as well as a "jump" part $\mathcal{J}f$ encoding the location and magnitude of the jumps in $f(x)$, expressed in the inset figure as a sum of Dirac-deltas.

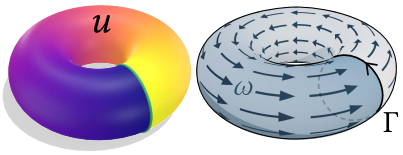
For a jump *harmonic* function in particular, the continuous part $\omega := \mathcal{D}f$, which we call the *Darboux derivative*, "forgets" jumps (inset). The Darboux derivative $\mathcal{D}f$ as "df modulo jumps", where d denotes the standard exterior derivative. Just as the standard exterior derivative df measures the failure of f to be constant, $\mathcal{D}f$ measures the failure of f to be piecewise constant. In essence, even though jump harmonic functions have discontinuities, the condition of harmonicity forces the Darboux derivative of a jump harmonic function to be globally continuous because the derivatives must "match" across the discontinuity.[1]



Because the Darboux derivative of a jump harmonic function "forgets" the jumps, there is no unique inverse to Darboux differentiation. In particular, we can only integrate "up to jumps": the inset shows one possible function f whose derivative explains ω, but this f is different than the original function. The moral of this story is that to map from derivatives back to curves, we can integrate ω and *choose* the jumps. This choice of jumps is what will allow us to decompose curves into bounding vs. nonbounding components.



Our understanding of the 1D case more or less extends directly to the 2D case. Instead of a 1D function on a periodic interval, we can, for example, consider a solution $u(x)$ of the jump Laplace equation on a 2D periodic domain (inset). We can likewise take the derivative of u obtaining a 1-form $\omega = \mathcal{D}u$. Just as $\omega(x)$ "forgets" about the jumps in a 1D piecewise linear function, $\mathcal{D}f$ forgets about jumps across region boundaries on a surface.



Thus Step ② in the SWN algorithm is taking the Darboux derivative of the jump harmonic function obtained in Step ①. The Darboux derivative ω of u is now a 1-form, which can be thought of as a vector field. If $\omega = 0$, then $u(x)$ is piecewise constant, meaning that $u(x)$ is already a valid (piecewise constant) region labeling. Conversely, if $w \neq 0$, then there are nonbounding components of Γ. In particular, nonbounding components of Γ, which are noncongruent to zero in the first homology group $H_1(M)$, are encoded by the harmonic component γ of the 1-form ω, which is noncongruent to zero in the first cohomology group $H^1(M) = \ker(\mathrm{d}_1)/\mathrm{im}(\mathrm{d}_0)$ (Section 2.3.2).

[1] Discontinuous functions $u : M \to \mathbb{R}$ that jump by an integer across curves can, for example, be represented as continuous angle-valued functions $\varphi := e^{2\pi\imath u}$. Angle-valued functions "forget" about any integer jumps, since $e^{2\pi\imath s} = e^{2\pi\imath(s+1)}$.

6.1.2 Derivative and curve decomposition

In Step ② of the SWN algorithm, we use Hodge decomposition (Section 2.2) to extract the harmonic part γ of ω. In this case, only $\delta\beta$ will be nonzero, due to singular behavior near interior endpoints. In the end, we need only solve a single Poisson equation $\Delta_2\beta = d\omega$, then evaluate $\gamma \leftarrow \omega - \delta\beta$.

Next, Step ④ is finding a *residual function* v whose Darboux derivative looks like γ, and hence describes the nonbounding part of our input curves. If we imagine this nonbounding part is a curve Γ', then v must jump across Γ', and should not jump across the complementary bounding component $\Gamma \setminus \Gamma'$. However, the choice of Γ' is in general ambiguous (Figure 6.2). Hence, we look for the *minimal* jumps, in an L^1 sense, needed for v to integrate γ. We formulate the resulting integration problem as a sparse linear program with $|F|$ degrees of freedom, where $|F|$ is the number of triangle faces in the underlying surface mesh.

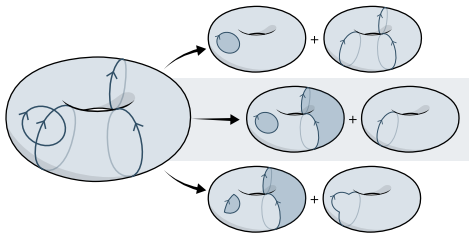


Fig. 6.2: A collection of loops can be decomposed into bounding and nonbounding components in many different ways. We look for the decomposition whose residual is shortest (*middle*).

Ultimately, the residual function v is another jump harmonic function whose jumps encode a completion of the nonbounding components of Γ. Step ⑤ of the algorithm is simply reading off the curve decomposition from the residual function v. The locus of points $\mathcal{J}v$ where v jumps represents a completion G of the nonbounding components of the input curve Γ; the bounding components of Γ are simply the complement of the nonbounding components, taken as a subset of Γ. Finally, we solve for the final winding number function w by solving Equation 6.2 with jumps encoded by $\Gamma \setminus \mathcal{J}v$. The function w can be rounded to yield integer region labels W.

Discretization. In the discrete setting, we assume the surface domain M is specified as a triangle mesh. We represent curves and regions as *chains* on M, meaning as signed integer-value functions on mesh edges and mesh faces, respectively. The jump Laplace equation can be discretized on the mesh as a sparse, positive definite linear system. Performing Hodge decomposition on the gradient of the solution u amounts to solving another Laplace-type sparse linear system. Finally, performing the integration procedure necessary to map the decomposition of our 1-form to a decomposition of our curve amounts to solving sparse linear program. The last two steps are only necessary if M is topologically nontrivial.

6.2 Boundary behavior of surface winding numbers

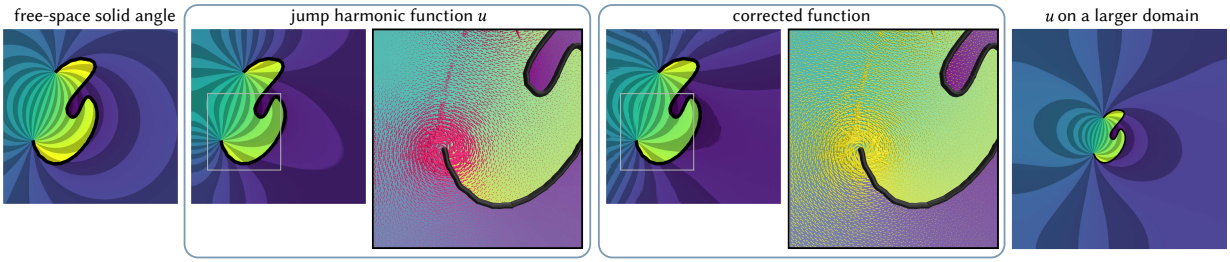


Fig. 6.3: *Left:* Generalized winding number (solid angle) in $\mathbb{R}^2$. *Center, left:* The solution u of Equation 6.2. *Center, right:* Using vector diffusion, the boundary behavior more closely matches the free-space solution. *Right:* Alternatively, one can simply move the curve farther away from the domain boundary, though this is not always easily possible.

The solution u to Equation 6.2 is not strictly equivalent to the free-space winding number (*e.g.*, Equation 6.1) because Equation 6.2 naturally has zero Neumann boundary conditions at the domain boundary ∂M. Concretely, if the domain M is planar, such that ∂M can be seen as a curve embedded in $\mathbb{R}^2$, the solution u to Equation 2.2 has different behavior at $\partial M \subset \mathbb{R}^2$ than the free-space generalized winding number solution (Figure 6.3).

To achieve boundary behavior analogous to that of ordinary winding number in the plane, we can alternatively directly solve for the derivative of u in Step ①. (Like in the signed heat method, diffusing derivative information avoids warping near the domain boundary — see Figure 4.11.) Since u is a jump harmonic function, its Darboux derivative $\omega := \mathcal{D}u$ is a continuous harmonic 1-form away from any endpoints of Γ. We can solve for ω by solving a Laplace equation acting on 1-forms, whose boundary conditions at the curve Γ match the known gradient behavior of solid angle, namely that ω behaves locally like $d\theta$ around endpoints of Γ. It is tempting to solve a Laplace problem using the Hodge Laplacian acting on discrete 1-forms (see Sharp et al. [2019c, §6.1.1]), with boundary conditions set to $\mathcal{D}u$ around the endpoints of the curve Γ. However, solving this equation would simply reproduce the warped behavior near the domain boundary. We thus try the following procedure, which converts ω to a vector field instead:

1. Solve Equation 6.2 for the function u, in order to obtain its Darboux derivative $\omega := \mathcal{D}u$.
2. Convert the discrete 1-form ω into a vector field X using Whitney interpolation.
3. Solve a constrained Laplace problem $\Delta^{\nabla} X' = 0$ with $X' = X$ at edges incident to the curve Γ.
4. Convert X' back to a 1-form, and integrate using the linear program described above for the residual function.

This process is not very elegant, especially the third step (solving a constrained Laplace problem) because it is unclear how many values we should constrain. One can see warping remain especially on the right side of the domain in Figure 6.3, center right. However, even though the resulting function doesn't match the true free-space solution exactly, it does negate some of the warping present in the original function u.

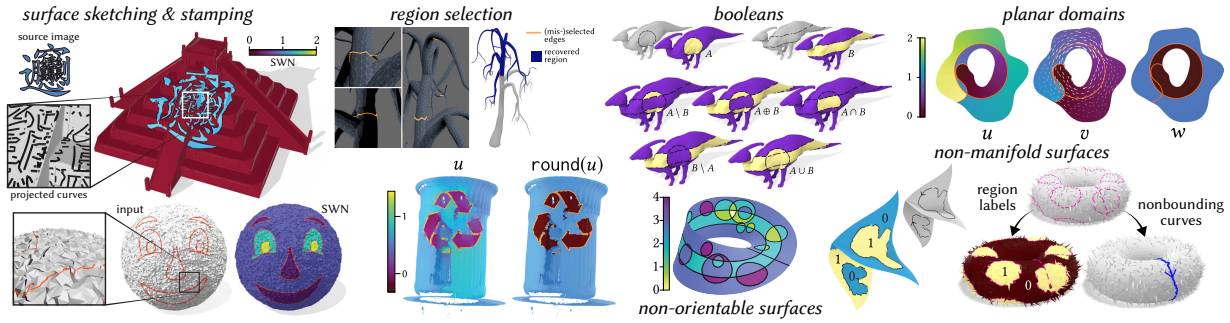


Fig. 6.4: Some examples demonstrating the utility and robustness of SWN.

Examples and applications. SWN can be used to compute inside/outside from incomplete oriented curves on surfaces, which arise in many settings ranging from curves projected onto noisy surfaces, to strokes painted on a noisy mesh, to imperfect user selections Figure 6.4. In general, SWN is robust to defects in the input curves, as well as defects in the surface mesh. Like with the signed heat method, an intrinsic formulation enables the use of *intrinsic retriangulation* if mesh tessellation quality is particularly bad [Sharp et al. 2019b; Gillespie et al. 2021; Sharp et al. 2021].

The magnitude of the jumps in the residual function also gives an idea of "how nonbounding" curves are (Figure 6.6).

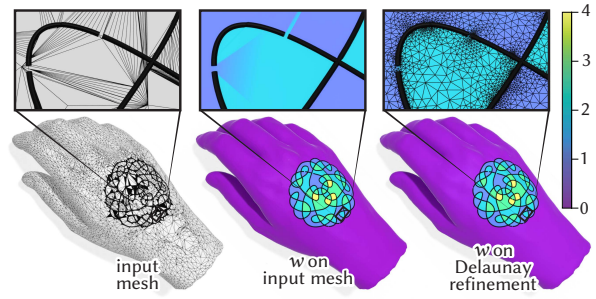


Fig. 6.5: Even on meshes with low element quality, SWN can produce reasonable region labels (*center*). Since our formulation is intrinsic, any remaining artifacts can be eliminated via *intrinsic Delaunay refinement* (*right*).

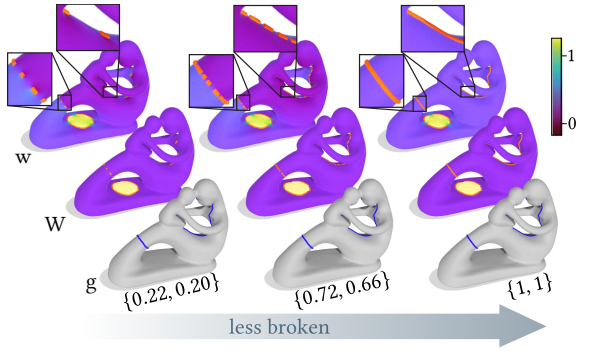


Fig. 6.6: As Γ becomes less broken, w approaches the expected winding number function, and the coefficients on nonbounding loops g approach 1.

6.3 Accuracy and performance of surface winding numbers

We measure the ability of SWN to accurately identify nonbounding curves by applying the algorithm on 934 test cases, 451 of which are defined on nonsimply-connected surfaces (meaning those with nontrivial topology). For each test case, we quantify error as the percentage of mislabeled surface area.

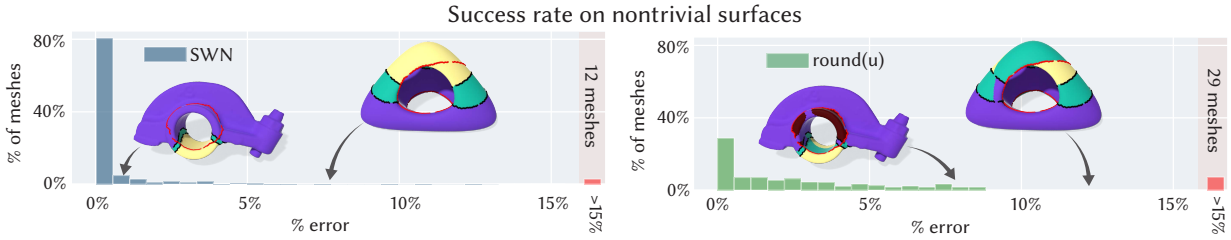


Fig. 6.8: Error rates for SWN (*left*) compared to naive rounding of u (*right*). The two example pairs show how naive rounding can fail to filter out nonbounding loops (in red).

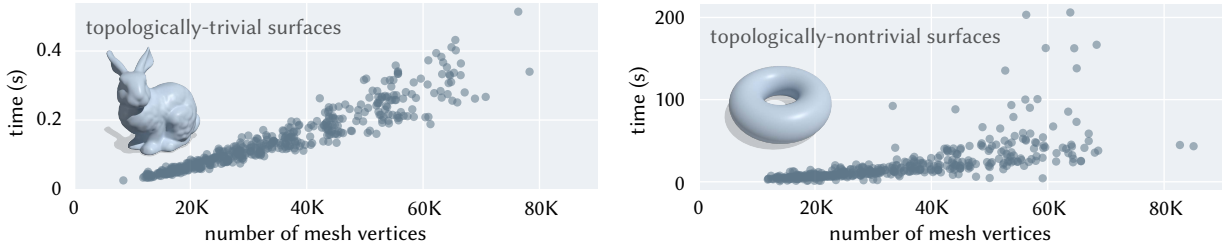


Fig. 6.9: On topologically trivial surfaces, SWN involves only a single sparse linear solve (*left*), and on surfaces with nontrivial topology solves an additional linear program (*right*). See the discussion below for a faster alternative.

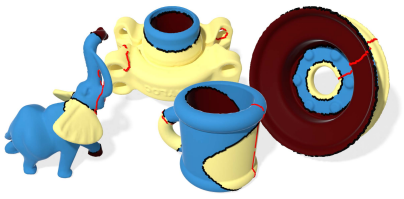


Fig. 6.7: Four of the 934 test cases in our synthetic benchmark. Each model is assigned ground truth region labels (indicated by colors), along with broken boundaries for those regions (black), and additional broken nonbounding loops (red).

On simply-connected surfaces, SWN typically takes less than two seconds (Figure 6.9, *left*), and achieves a mean/max error of only 0.14%/5% (Figure 6.8, *left*). On nonsimply-connected surfaces, SWN must solve an additional linear program and can take up to a few minutes (though see below for an approximate, faster alternative), but achieves errors under 0.5% on 80% of models (Figure 6.8, *right*). Some "errors" actually were not due to the method, but simply due to fundamental ambiguity in the input: Figure 6.10 shows one such example with multiple correct answers.

More importantly, SWN performs much better than naive rounding of the function u (analogous to how generalized winding number is contoured in $\mathbb{R}^d$). Simply rounding u without processing nonbounding curves can create phantom curves (Figure 6.1) which significantly degrade the accuracy of the final labels (Figure 6.8).

Reduced-size linear program. The original linear program (LP) simultaneously optimizes for both the sizes and locations of the residual function's jumps. Instead, we could first separately try to complete the curve Γ using a shortest-path heuristic (for example, Dijkstra),

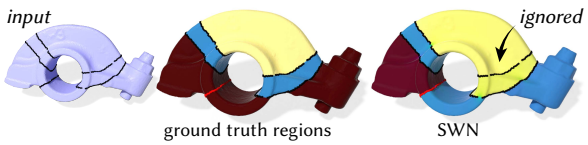


Fig. 6.10: The curves on this model represent five homologous loops; given their orientations, there are actually multiple possible valid curve decompositions. SWN always returns the one with the shortest collection of nonbounding curves (Figure 6.2).

partitioning the domain M into several connected components (Figure 6.11).

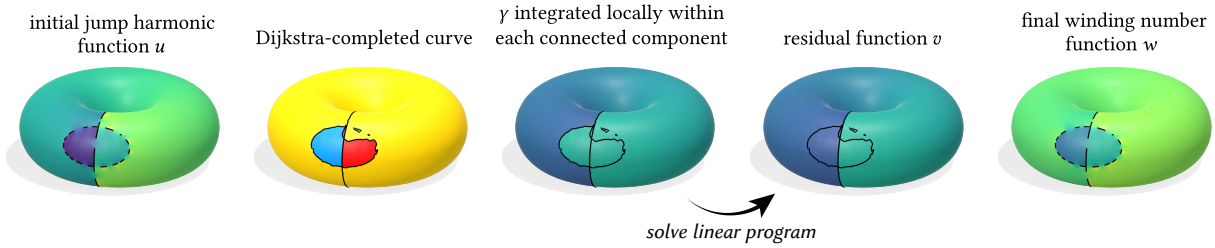


Fig. 6.11: A reduced-size linear program uses a path-finding heuristic to reduce the number of degrees of freedom from $|F|$ to just a few connected components; otherwise one follows the same procedure as the rest of the SWN algorithm.

We can then locally integrate the harmonic 1-form γ within each connected component, similar to how we integrated within each triangle face in the original LP, before using the original LP to minimize the L^1 norm of jumps across connected components. This means the number of DOFs is reduced from the number of faces in the mesh to the number of connected components, which are relatively few. As a result, whereas the original LP took up to a few minutes in the worst cases, this approximate solution takes at most a few seconds, at the cost of a small decrease in accuracy on the benchmark (Figure 6.12). This represents a ~100x speedup.

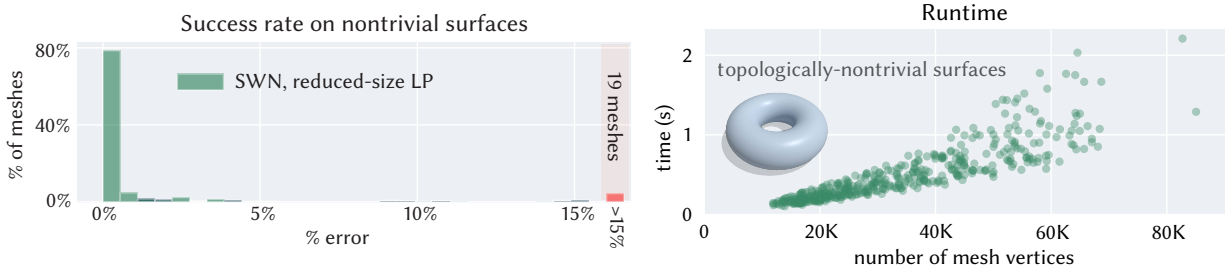
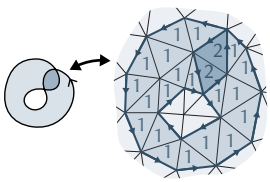


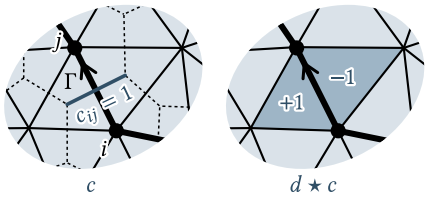
Fig. 6.12: The reduced-size linear program (*right*) is on average about 100 times faster than the original linear program (*left*) for topologically non-trivial surfaces, with comparable accuracy for region classification. (Compare with Figure 6.9).

We could have also plugged in a different shortest-path method for the curve-completion step; for example, we could use weighted Dijkstra where the weights correspond to the magnitude of the gradient of the jump harmonic function u.

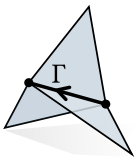
6.4 Extensions to the surface winding numbers algorithm



A dual formulation. A curve can be viewed as the boundary of "weighted regions", where the weights are the winding numbers. Discretely, this means if we view the curve as a 1-chain $\mathcal{C}$ in a triangulation of the domain, then the winding number is a 2-chain $\mathcal{W}$ such that $\partial\mathcal{W} = \mathcal{C}$ (inset).



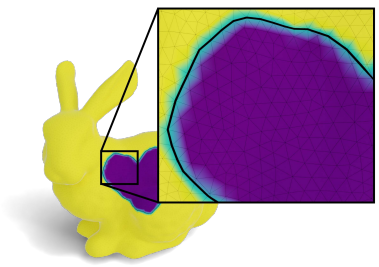
This perspective leads to a "dualized" formulation of the algorithm presented above, where we solve for a dual 0-form (values per face) instead of a primal 0-form. If we encode the curve Γ as a dual 1-chain c, then the relationship "Γ bounds a region b" can be expressed as the linear relation $db = c$ where d is the discrete exterior derivative acting on the dual 0-form b, and the jump condition induced by Γ is interpreted as a dual 1-form c (inset). The least-squares solution is given by another Laplace equation, $d \star db = d \star c$. To identify the nonbounding components of the curve, one can do Hodge decomposition on the derivative, just as before (although the corresponding equations will also be appropriately "dualized").



The dual formulation streamlines certain concepts; for example, the jump derivative $\mathcal{J}$, which represents the locus of points where a function jumps, becomes the usual boundary operator. Encoding the input curve as a dual 1-chain would subsume more difficult situations on non-manifold and non-orientable meshes where primal 1-chains can't unambiguously specify more than two "sides" of a curve segment that runs along a non-manifold edge — in other words, dual 1-chains are a way of addressing ambiguity in a curve's normal orientation, even if it has a well-defined tangent orientation (Section 2.3.1).

The dual algorithm is the same as the primal algorithm in that it solves the same linear systems and linear program, but on dual elements. On the other hand, the primal formulation will perhaps be more familiar to graphics practitioners. For example, in the primal formulation, discretizing the jump Laplace equation yields the ordinary cotan-Laplace matrix on vertices, whereas in the dual formulation one would build an analogous cotan-Laplace matrix acting on 2-forms (though the latter isn't any more difficult to build than the former). The primal formulation also yields better resolution by solving on corners (vertices), rather than on faces. One approach isn't clearly better over the other; we picked the more familiar primal formulation.

Non-conforming curves. The SWN algorithm assumes the input curve Γ is restricted to mesh edges. However, we can relax this assumption, at least in the first step (solving for the initial jump harmonic function u). Instead of a Laplace equation with jump conditions, we may consider an equivalent Poisson equation with a source term encoding the gradient of an indicator function



$$\Delta u = \nabla \cdot N \mu_\Gamma. \tag{6.3}$$

The source term on the right-hand side of Equation 6.3 corresponds to a singular vector field concentrated on Γ, where the vectors align with the normals of the input curve. This is in fact the perspective taken by the classic Poisson surface reconstruction algorithm — which, as noted in Section 3.1, is equivalent to a regularized version of generalized winding number and solid angle. We can discretize this Poisson equation using finite elements, although the solution is inevitably "smeared out" somewhat if we use continuous elements to represent a discontinuous

function (inset). We can still take the derivative of this function in the same sense of the original algorithm, and Hodge-decompose. On the other hand, integration is less straightforward.

6.4.1 Nonbounding loops and discontinuous distance

When solving for (generalized) signed distance on surfaces of nontrivial topology, one may want to first filter out nonbounding components using SWN. Then, one can solve for signed distance without worrying about whether the sign is well-defined.

Feng and Crane [2024] also explore an alternative approach using an expanded notion of what is considered a valid signed distance function. In particular, we can replace the L^2 problem in the third step of the signed heat method (Equation 4.6) with an L^1 problem that ignores neighborhoods where vector field Y_t is highly nonintegrable. On triangle meshes, we quantify nonintegrability via an edge-based curl inspired by Polthier and Preuß [2003, Section 4]. We formulate the L^1 problem as a linear program whose structure is the same as the one used in SWN: we solve for a (possibly discontinuous) function ϕ represented as piecewise linear functions interpolating values at face corners, and minimize a L^1 norm that penalizes discontinuous jumps across edges, while enforcing integration of the vector field Y_t in each triangle face. In summary, we still aim to integrate Y_t as usual, but "try less hard" when the field is not integrable; simultaneously, the L^1 objective tries to minimize the length of any discontinuity. An example is shown in Figure 6.13.

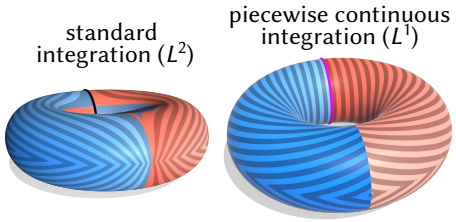


Fig. 6.13: *Left:* The signed heat method (Chapter 4) produces extremely warped isolines when a curve fundamentally has no inside and outside. *Right:* Allowing piecewise continuous functions produces well-behaved, evenly-spaced isolines despite the fundamental sign ambiguity.

Chapter 7

Practical notes

One result of this thesis is that (screened) winding numbers — meaning functions satisfying a jump (screened) Laplace equation — are closely related to signed distance functions. The latter are often more expensive to compute, but often give more robust reconstructions, especially for point-sampled or poorly-sampled data. This chapter contains further exploration of winding numbers- and generalized signed distance-based reconstruction, such as behavior in the presence of imperfections. The purpose of this chapter is to help users navigate the finer points of algorithm behavior.

7.1 Winding numbers vs. generalized signed distance for surface reconstruction

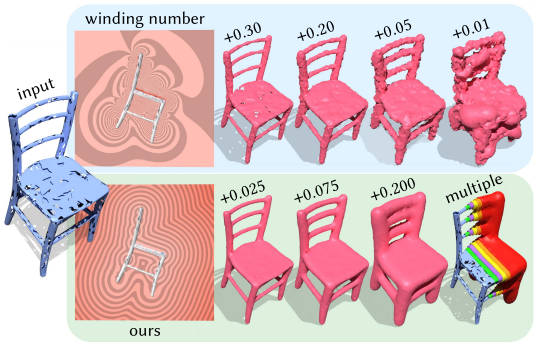


Fig. 7.1: *Top:* Generalized winding number cannot be used for offset surfaces, since it provides only a smooth indicator function — not distance. *Bottom:* Generalized signed distance provides much nicer offsets on the same broken mesh.

In a nutshell, our generalized signed distance methods are better than winding numbers at completing broken geometry. On the other hand, in $\mathbb{R}^3$, evaluating winding number can be much cheaper because it admits a simple closed-form boundary integral formula, which can further be accelerated using Barnes-Hut [Barill et al. 2018]; thus, if the input geometry is very "nice", you may be able to get results comparable to the more expensive generalized signed distance methods (Figures 7.7 and 7.8, see high-density examples in the left column without corruption). However, on curved surface domains, the cost between our surface winding numbers algorithm (Chapter 6) and the signed heat method (Chapter 4) remains comparable, since they both must solve similar Poisson problems.

Offset surfaces. One clear limitation of winding numbers is that they cannot easily provide surface reconstructions beyond the zero level set. Figure 7.1 uses the signed heat method to compute well-behaved offset surfaces of a broken chair mesh.

Inside/outside accuracy. On curved surface domains, one can estimate the inside and outside regions bounded by corrupted curves either using surface winding numbers (SWN, Chapter 6) or the signed heat method (SHM, Chapter 4). To compare the methods' inside/outside accuracy, we create a dataset of 220 examples. As input surface domains, we use genus-zero meshes without boundary and with ~5k vertices from Myles et al. [2014]; 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. (All curves are thus originally bounding, and we do not have to be concerned about the possibility of nonbounding curves.)

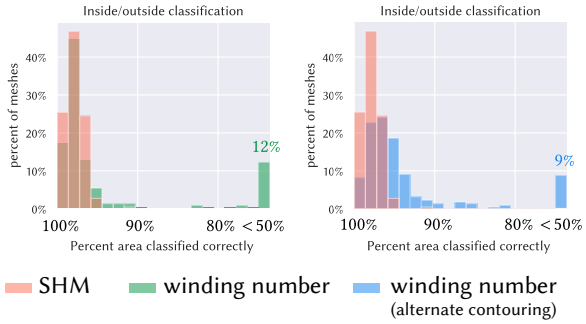
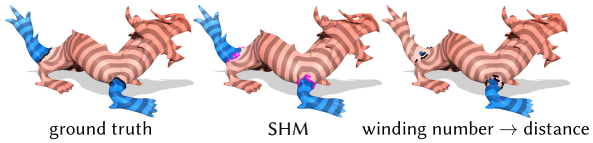


Fig. 7.2: For corrupted geometry, winding numbers are often worse than generalized signed distance (SHM) at classifying inside/outside.

We found that SHM was better than SWN at classifying inside and outside (Figure 7.2). This experiment also highlights the difficulty in contouring winding number. Contouring has been a noted Achilles' heel for winding number since its introduction to graphics; Jacobson et al. [2013, §5] notes that simply contouring in $\mathbb{R}^n$ according to the 0.5-level set is often insufficient and proposes a graph cut optimization, though it is not clear the latter is more effective (the latter method is not implemented in the official `libigl` release). On surfaces, an alternative approach contours according to the average of the winding number function along the input curves. Our SWN algorithm uses a more effective contouring method [Feng et al. 2023, §3.5.1], but it still 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. However, regardless of what contouring strategy we tried, there were a significant number of cases on our benchmark where both contouring methods misclassified more than 50% of the surface area on about 10% of examples. As a result, computing signed distance to curves contoured from SWN yields about 3x lower distance accuracy than SHM on average (0.11 vs. 0.04 L_2 error, *resp.*), even though the SWN→contour→distance pipeline benefits from exact distance computation via MMP [Mitchell et al. 1987] (inset). On surfaces, this pipeline is also about 10x more expensive due to its intermediate steps; solving the sparse Poisson problem in SWN already has cost asymptotically equivalent the SHM, which also hinges on solving sparse Poisson linear systems.



Completion of corrupted surfaces in $\mathbb{R}^3$. We find that using higher-order information about the input surface aids reconstruction of surfaces with large holes: for example, the vector diffusion of SHM yields more faithful surface reconstruction than winding numbers and Poisson surface reconstruction, which essentially use scalar diffusion. The latter methods fill in holes with flat or saddle-like patches, behavior which is endemic to their underlying PDE and thus *independent* of the observed surface. In contrast, vector diffusion interpolates observed normal vectors, so holes are filled with patches that more closely align with the curvature of the surrounding surfaces (Figure 7.3, *left*).

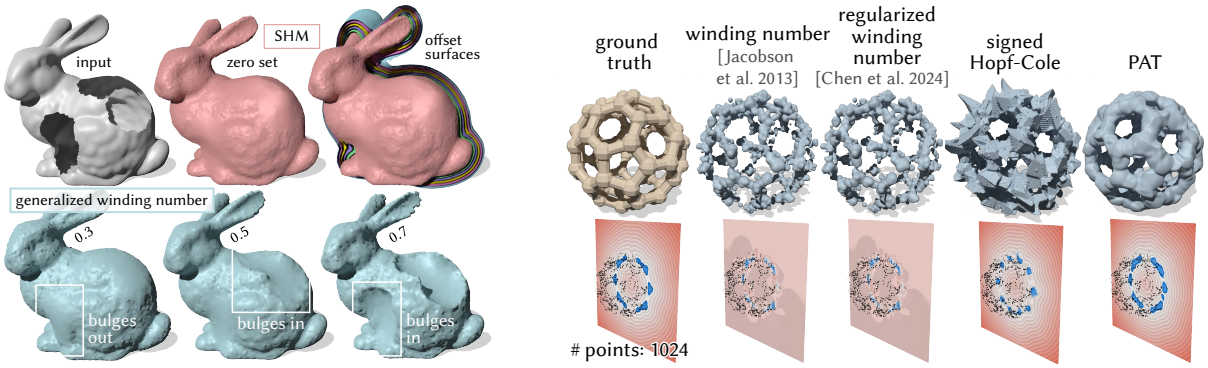
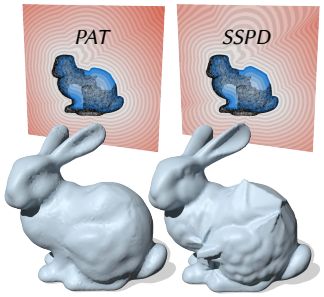


Fig. 7.3: *Left:* Winding number completes surfaces with saddle-shaped harmonic patches that exhibit poor normal continuity with the observed geometry (across many contour values). *Right:* PAT gives better reconstructions than generalized winding number or its regularized variant, especially for sparse data.

Winding number methods can also struggle with sparse or unevenly sampled input data. Figure 5.13 shows a more thorough set of examples comparing the reconstruction results of points as tori (PAT, Chapter 5), SHM, winding number, screened Poisson surface reconstruction [Kazhdan and Hoppe 2013], and NN-VIPSS [Xia and Ju 2025]. The neural network of PAT was not trained on point clouds with large regions of completely missing data, and so does not produce as good surface completions as SHM (inset). PAT does still produce better results than a naive convolutional method (Equation 5.9).



Completion priors. Winding numbers and our generalized signed distance methods have different surface completion priors. In a nutshell, winding numbers tend to close or "cap off" open shapes, while the signed heat method tend to extend the boundaries open shapes, because it favors smooth extrapolation of both position and normal information across gaps (Figure 7.4).

The signed heat method additionally can exhibit "pointy" shape completions; the pointiness tends to be-

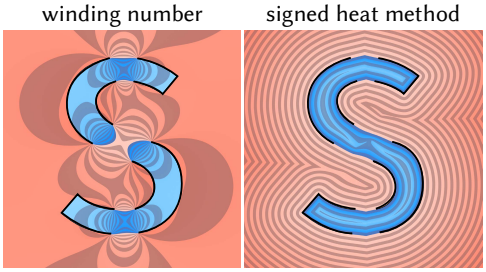


Fig. 7.4: Winding numbers tend to cap off open curves, whereas the signed heat method tends to continue them.

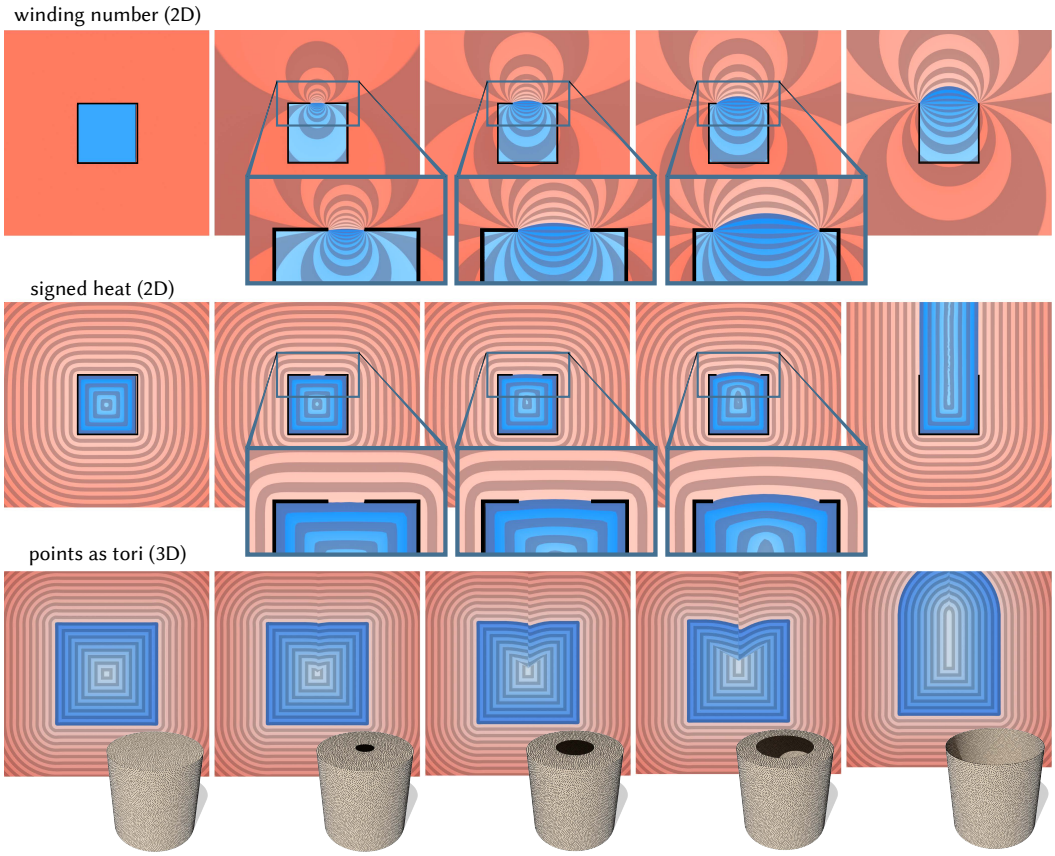


Fig. 7.5: We show the shape completion behavior of the three methods as a flat hole grows in size. *Top:* In 2D, winding numbers always connect curve endpoints with circular arcs (in 3D, more counter-intuitive saddle-shaped surfaces may emerge). *Middle:* The signed heat method considers normal information and hence has flatter completions, though still exhibits some bulging as the ratio of horizontal vs. vertical segments shrinks; the completed shape also exhibits a sudden change between the rightmost and second rightmost examples, when horizontal segments disappear entirely. *Bottom:* PAT works purely locally rather than solving a global reconstruction problem, and hence can exhibit discontinuity where neighboring fitted tori disagree.

come more visible as the underlying meshed domain becomes highly refined (Figure 7.4, *right*, top of "S" shape). The reason for "pointy" shape completions is because as vector diffusion time $t \to 0$, points inherit the normal at the closest point, leading to a cusp where the closest point changes.

The signed heat method always use the default diffusion time $t = h^2$, where h denotes the mean distance between nodes in the computational domain, following past heat methods [Crane et al. 2013b; Sharp et al. 2019c]. This choice of timestep was motivated by the fact that Laplacian Δ is second-order and thus a diffusion time proportional to h^2 makes the algo-

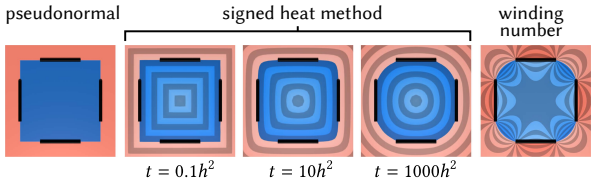


Fig. 7.6: SHM's diffusion time affects shape completion, interpolating between the linear prior used by the pseudonormal test (as $t \to 0$) and the harmonic prior used by generalized winding numbers (as $t \to \infty$).

rithm invariant to scaling and refinement. However, perhaps the timestep should instead be set according to the geometry of the source set, rather than the discretization of the underlying domain; after all, the refinement argument makes little sense in $\mathbb{R}^n$ where discretization is not required for short-time diffusion. Figure 7.6 shows an instance of tuning the timestep to complete sharper or rounder completions of a square, though it remains unclear how to pick a single timestep for a shape with multiple missing regions of different sizes and shapes. An extension of the signed heat method could perhaps diffuse estimated curvature information in addition to normal vectors.

In the other direction, winding numbers do not consider even normal information with respect to shape completion. On the one hand, this means winding numbers can complete large holes poorly (Figure 7.3, *left*). On the other hand, because winding numbers do not consider normal information, shape completion is less sensitive to perturbations that do not introduce significantly different normals (Figure 7.5).

Winding numbers and SHM may be limited compared to data-driven priors that make more use of global and/or semantic information about the shape. Section 8.1 includes a few suggestions on how one might use our methods in combination with modern machine learning methods.

7.2 Empirical conditions on input

Each of the algorithms presented in this thesis (winding numbers, signed heat method, points as tori) tend to degrade gradually as the quality of input degrades: in other words, there tends to be a smooth transition in the resulting inside/outside or signed distance function as the input mesh or point cloud becomes increasingly corrupted. A question that arises is when a defect becomes a feature according to each algorithm, or equivalently, at what level of corruption each algorithm finally fails to ignore the corruption. For instance, the signed heat method essentially ignores small holes in the surface domain (Figure 4.7, *lower left*), but it would be unclear whether a very large hole were an intended feature of the surface, maybe even to a human.

Below are the results of several experiments using a family of cylinders with varying aspect ratio, and spheres with varying radius, showing how each algorithm degrades under increasing corruption. Each cylinder has a fixed height of 1; varying cylinder aspect ratio is achieved by varying cylinder radius, so smaller "cylinder radius" means the cylinder is more "skinny". Each shape is centered at the origin. Each shape is sampled into an oriented point cloud using uniform sampling; each point cloud is optionally corrupted by Gaussian noise, outliers, or flipped normals. Each point cloud has 2048 points, though we also run experiments on varying sampling density. To add Gaussian noise, both point positions and normals are additively perturbed by an amount drawn from a 3D normal distribution with mean zero and a specified standard deviation. Outliers are generated by sampling n_{outliers} additional points from $[-1, 1]^3$ uniformly at random, where n_{outliers} is equal to the specified fraction of points in the original point cloud, rounded down to the nearest integer; normals are sampled uniformly at random from the unit sphere. Flipped normals are generated by starting with ground-truth normals, and flipping the sign of a specified fraction of points, chosen uniformly at random.

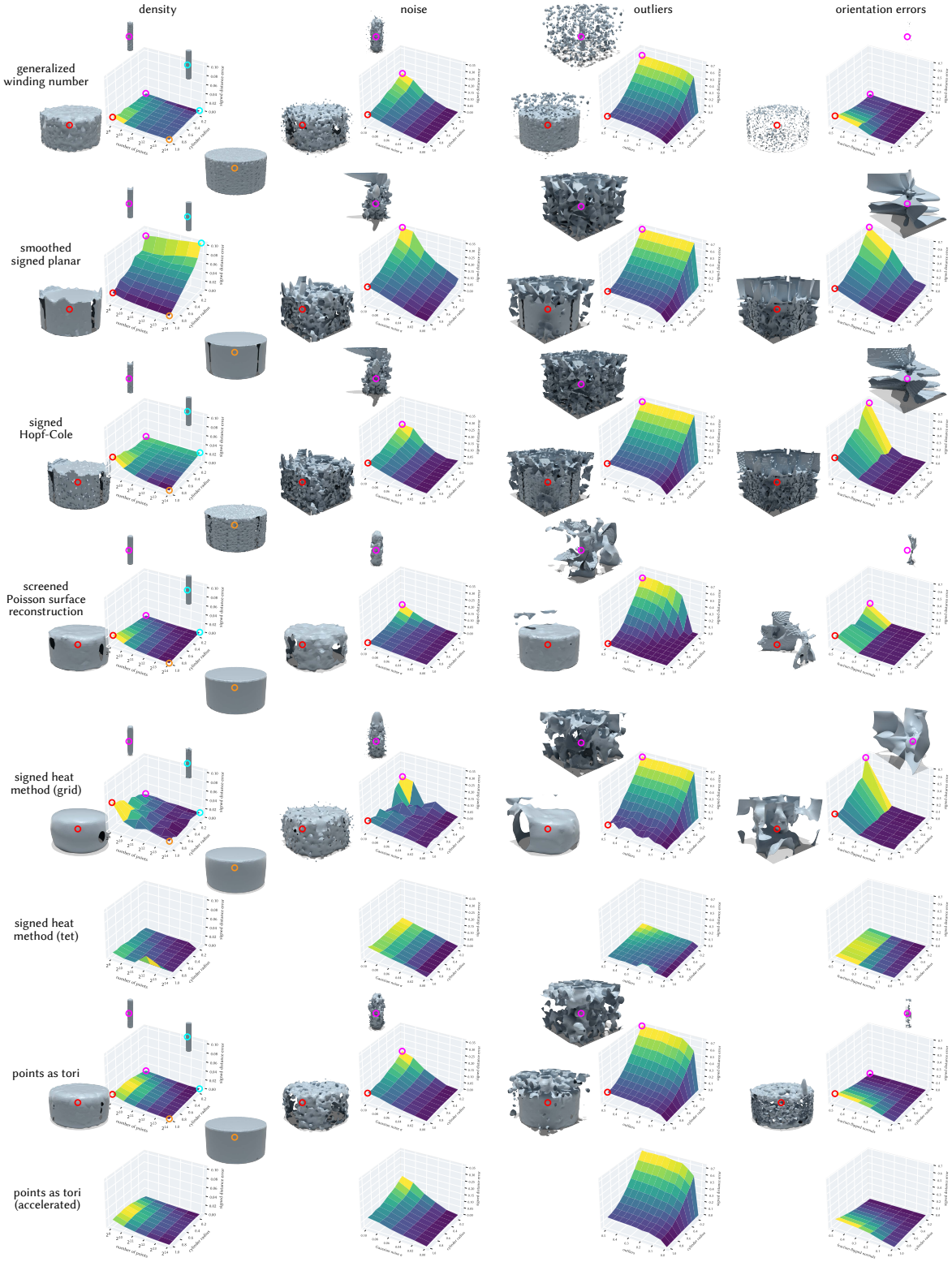


Fig. 7.7: Landscapes of signed distance error as point clouds of a cylinder become progressively worse under four different types of corruption, and varying cylinder aspect ratio. Colored circles match sets of conditions to visualizations of the corresponding contoured shape.

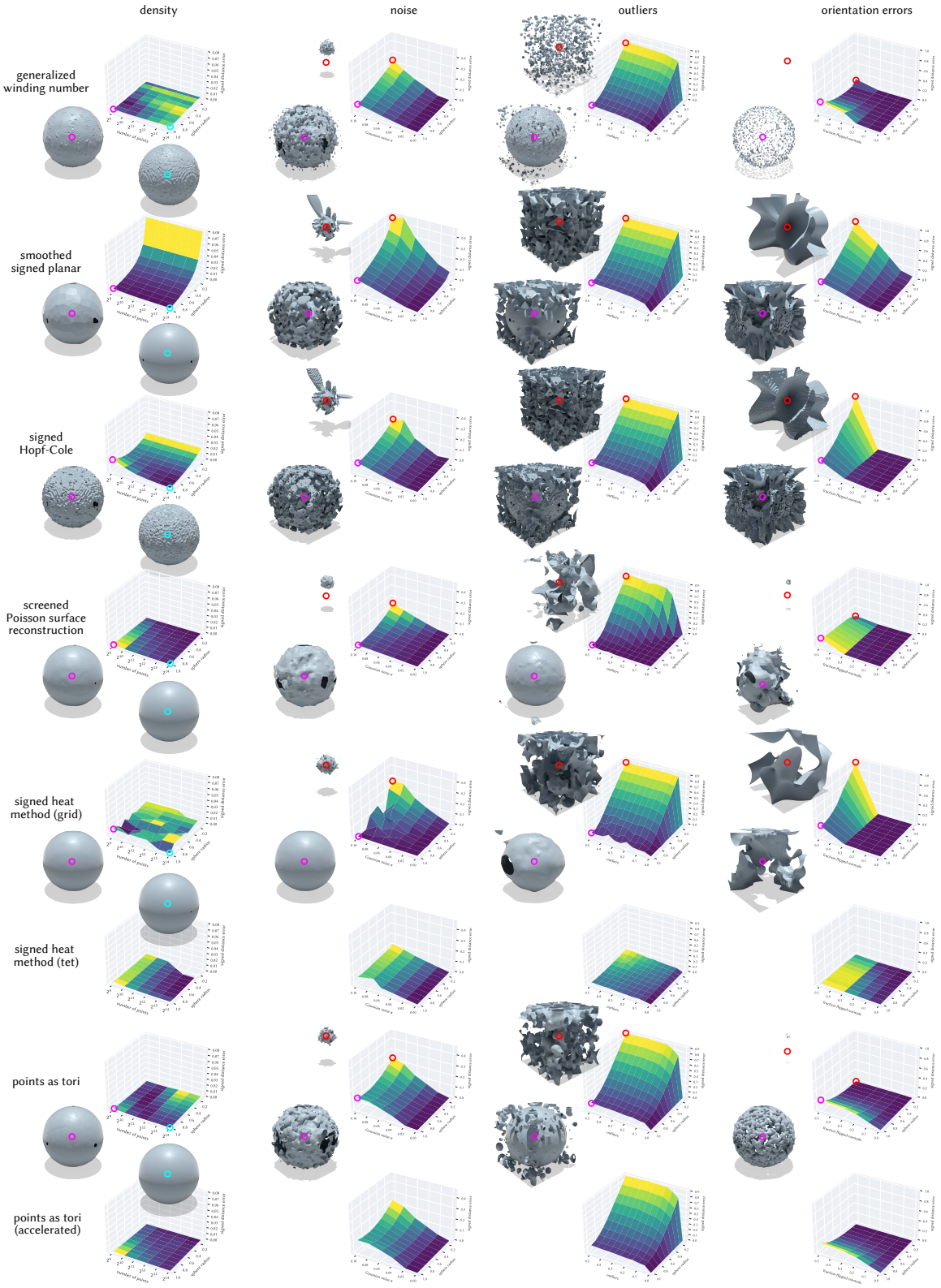


Fig. 7.8: An analogous set of experiments to those in Figure 7.7, using a sphere instead of a cylinder.

The error of each method is defined as error in signed distance, computed the same as in Section 5.3: we evaluate all methods on a regular grid of resolution $G := 64^3$, and compute their mean absolute error over all grid points $\mathbf{q}_i$

$$\epsilon(\phi) := \frac{1}{G} \sum_{i=1}^{G} |\phi(\mathbf{q}_i) - \phi_{\text{true}}(\mathbf{q}_i)|,$$

where ϕ is the signed distance estimate, ϕ_{true} is the ground-truth signed distance.

We show the resulting signed distance errors of generalized winding number (GWN), the signed heat method (SHM), points as tori (PAT), signed Hopf-Cole distance (SHC, Equation 5.8), smoothed signed planar distance (SSPD, Equation 5.9), and screened Poisson surface reconstruction (SPSR). For methods that do not directly compute signed distance (GWN, SPSR), the resulting function is first contoured into a triangle mesh on the 64^3 grid using marching cubes, and signed distance computed from the triangle mesh. (For performance comparisons, see Section 5.3.) Select meshed contours are also visualized; the results of the tet mesh-based version of SHM are not shown because `libigl`'s marching tetrahedra implementation, which was used to contour functions on tet meshes, appeared to produce incorrect results. (Occasionally there are holes in the contours simply because they intersect the boundary of $[-1, 1]^3$.) We observe similar trends when the point clouds are sampled using farthest point sampling, rather than uniform.

Unsurprisingly, all methods tend to produce more accurate results as sampling density increases. Even so, SHC and SSPD remain brittle even at high densities: even when the error is not dramatically elevated, their contoured surfaces can be bumpy.

The error in all methods tends to be exacerbated as cylinders grow thinner (smaller cylinder radius), or spheres grow smaller. This is probably unsurprising: especially in the case of noise or outliers, the relative magnitude of a fixed level of perturbation becomes larger as feature sizes shrink, making it difficult for any algorithm to distinguish the true surface from corruption. Grid-based SHM seems to tolerate Gaussian noise well up to a variance about 10% of cylinder or sphere radius; PAT and winding numbers do not appear to share as much robustness, exhibiting a steady increase in error with noise.

In the presence of noise and outliers, SHM and SPSR tend to produce smoother shape completions than GWN; SHM and SPSR both solve global Poisson problems that fit functions to the observed data. GWN can produce noticeably bumpy contours even at high sampling densities, because the winding number is singular at the source geometry. Thus for shape completion, regularized winding numbers — which additionally involves convolution with a Gaussian kernel — is likely a better choice than ordinary winding numbers [Chen et al. 2024a]. SPSR and SHM also seem to gracefully tolerate up to 30% orientation flips, whereas winding number seems to only tolerate up to 20%.

Among the types of corruption tested, nearly all methods are most sensitive to the presence of outliers: the error landscapes exhibit a sharp discontinuity as soon as even a small fraction of outlier points is introduced. SPSR is the only method that does not have such a sudden sharp increase in error as the amount of outliers becomes nonzero: instead, the error increases more

smoothly before shooting up once the ratio of outliers fraction to cylinder or sphere radius increases beyond about 0.2 to 0.3.

PAT's behavior in the presence of noise and orientation errors is somewhat of a departure from that of the other methods: as the corruption increases, the fitted tori tend to degenerate toward small spheres concentrated around each input point, leading to less extreme signed distance error at the highest levels of corruption — but the resulting contours have a distinctive "grape bunch" appearance. PAT and its accelerated variant (which effectively sums only over a subset of points nearest to the query point) exhibit very similar behavior across all experiments, suggesting the acceleration does not substantially alter the robustness properties of PAT.

Finally, the error landscape of SHM is notably less smooth than those of other methods. I hypothesize that as point clouds become sparse, or as noise increases, relatively small perturbations in the distribution of input normals can lead to outsized changes in the completed surface (see discussion in the "Completion priors" paragraph above, and Figure 7.6). Some fluctuations may also be due to differences in the linear solvers used in the SHM implementation, which switches between an algebraic multigrid method [Demidov 2025] and a direct solver in case the first fails. Additionally, the tet mesh-based variant of SHM often produces lower error than the grid-based variant; I hypothesize that the adaptive nature of the tet meshing, which produces coarser tets where input points are less dense, results in less sensitivity to outliers especially (see Figure 4.17). However, it remains unclear why tet meshing also appears to result in less sensitivity to orientation errors.

7.3 Alternative approaches for perfect geometry

This thesis presents methods for processing geometry robustly in the presence of imperfections like missing data and noise; there is less incentive to use our methods if the input geometry is perfect or close to perfect. Below we list a few simpler and faster alternative approaches that may be applicable if input geometry is close to perfect.

If one has as source geometry a watertight triangle mesh without any defects, then there exist many possible methods for computing the inside/outside of the mesh, or distance in $\mathbb{R}^3$ to the shape, exactly as-is. For example, one can compute unsigned distance via fast, exact closest point queries via a library like FCPW [Sawhney 2021], and sign this distance using the classic winding number ("single-sample" methods for inside/outside computation, like ray shooting and the pseudonormal test, tend to be numerically brittle). Alternatively, one can sample geometry onto a grid and use methods like fast sweeping to compute (signed) distance [Osher et al. 2004], for which there also exist parallel and sparse implementations [Detrixhe et al. 2013; Vicini et al. 2022; Museth 2017].

Likewise, if one has as input simple, closed, bounding curves without defects on a curved surface domain also without defects, then one can compute signed distance using the fast marching method (FMM) [Kimmel and Sethian 1998]. To compute inside/outside of such curves, one can simply run (the first step of) our surface winding numbers algorithm, which takes up to a few seconds on meshes with a few hundred thousand vertices. Alternatively, one might compute

inside/outside using a flood-fill algorithm that starts by assigning one corner an arbitrary starting value, then uses *e.g.* breadth-first search to progressively assign values to all other corners, incrementing or decrementing the value being assigned each time one enters into an adjacent corner by crossing the oriented curve from the right or left, respectively. However, note that wavefront-based methods like fast marching and the proposed flood-fill method will lead to significant artifacts if there are even slight defects in the input (Figure 7.9).

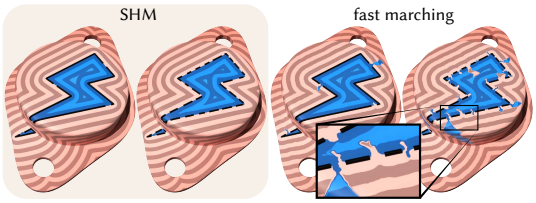


Fig. 7.9: The fast marching method is accurate and efficient when the input curve and surface are perfect, but otherwise can easily propagate sign and distance errors.

If one has as input an evenly and densely sampled point cloud without noise — in particular, if the maximum spacing between points is not greater than the size of features one hopes to reconstruct — then one can obtain a decent signed distance reconstruction from the point cloud by using one of the naive convolutional distance formulas in Equation 3.15 or Equation 3.16 presented in Section 3.2, and setting λ small enough so that points are merged (this approach to reconstruction is analogous to the classic "metablob" approach for implicit modeling). Likewise, one can estimate inside/outside using fast winding numbers for point clouds [Barill et al. 2018]. However, note that naive convolutional distance formulas have no capability of ignoring noise and outliers (surface samples that are false positives), even if the rest of the point cloud may be sampled well.

Chapter 8

Conclusion

Ideally, this thesis has convinced you, the reader, of the need for robust geometry processing, and of the effectiveness of generalized geometric queries tolerant of defective data. In practice, the algorithms developed in this thesis have so far been used in downstream engineering and design tasks such as improving manufacturability [Wang et al. 2025b], boolean operations [Jung et al. 2025], geometric sampling [Suriyababu et al. 2025], surface reconstruction from sparse curves [Yu 2025], and mesh LODs [Ladeuil et al. 2026]. By strengthening operations as fundamental as inside/outside and distance computation, we have strengthened algorithms across fields like computer graphics and vision that all depend on having good infrastructure for processing geometry.

Rather than merely contributing what might feel like the umpteenth reconstruction algorithm, we shed light on why our generalized signed distance methods work well (and how they can be extended to fast, output-sensitive methods) by drawing on theoretical connections between signed distance and inside/outside. We also identify a core difficulty for any inside/outside algorithm: the existence of nonbounding loops. In formalizing and highlighting fundamental difficulties, especially with signed distance in Chapter 3, we also draw connections to other "manifold learning" problems in *e.g.* machine learning that are fundamentally geometric in nature.

One theme that has appeared throughout this thesis is that functions such as winding numbers and signed distance can sometimes be nicer than working directly with defective curves and surfaces, because functions offer a richer computational space, and are tolerant of defects. Functions also offer computational flexibility in the sense that their smooth equations and variational perspectives can often be re-applied to different representations of geometry.

Another theme is that higher-order information about geometry is valuable for high-quality generalization. The signed heat method (Chapter 4), which diffuses gradient information, obtains better shape completions than methods based purely on scalar diffusion (Figure 7.3, Chapter 3), though as a result requires an additional integration step. The points as tori method likewise tackles generalized signed distance by effectively estimating the curvature of the underlying surface using a learned kernel.

Finally, another theme is that diffusion in general is a powerful mathematical and compu-

tational tool — a theme that may be "obvious" to those familiar with geometric methods, but is worth highlighting. In each of the methods developed in this thesis, diffusion (either scalar- or vector-valued) plays an important theoretical or computational component: beyond simply providing a "nice" way to interpolate data, diffusion yields connections to parallel transport, connections betweeen geodesic distance and kernel methods (Varadhan's formulas, Hopf-Cole transformations...), and connections to topology via de Rham cohomology. Diffusion also has nice computational properties that allow us to turn these ideas into concrete, well-behaving algorithms; and there exist mature tools to implement diffusion on different domains and data structures, from $\mathbb{R}^n$ to discretized surfaces. That being said, simply applying diffusion naively is not a guarantee for success: in particular, Chapter 3 explains in depth the limitations of naive convolutional distances, and Section 5.5 describes the challenges we experienced using standard techniques on the way to developing the points as tori method.

Looking forward, I hope that the emphasis on reliable manipulation of geometry provides foundational components for geometric problems at large. This thesis primarily considers problems of geometric *inference* — that is, drawing conclusions about geometric properties of shapes given limited observations — and I believe our methods will provide foundations for geometric *optimization* as well: that is, the active search for a curve or surface that best satisfies a given design objective. For example, the points as tori method contributes a novel point-based representation whose differentiability and output-sensitivity begs applications to inverse problems. It is tempting to think that deep learning renders classic geometry processing obsolete by (supposedly) being able to "just learn" everything, though I believe data-driven methods need robust geometry processing more than the other way around, especially due to the richness of geometric phenomena. Building off Section 5.5.5, below I list a few possible suggestions for future work.

8.1 Using geometric methods to inform data-driven methods, and vice versa

Both the signed heat method and surface winding numbers algorithms, though effective, use relatively simple geometric priors to address ambiguity. However, the likelihood that a predicted shape completion explains the given observation can depend on both what was already observed, and prior information. For example, one would probably expect that the completion of a partially-observed fencepost should be "sharper" than the completion of a smoother object like a bowling pin. Or, perhaps one is only interested in performing geometric queries on bowling pin shapes, and would like to completely ignore the possibility of sharp, fencepost-like completions. The current wisdom of this decade is to incorporate data-driven priors, and/or to condition shape completions on already-observed data (perhaps also in a data-driven manner). Here, the function-based perspective of our methods aids the incorporation of neural network-based methods or other variational approaches. Conversely, the geometric insights developed in this thesis may be used to inform data-driven pipelines.

Learned parameters. Our algorithms also contain parameters that could instead be set by learned priors. For example, the extent to which the signed heat method interpolates the original input data can be controlled by a possibly learned parameter (Section 4.4); similarly, the points as tori's λ parameter could be tuned differently depending on the input data's fidelity (Figure 5.16).

Fine-tuning. The points as tori method (Chapter 5) already uses data-driven priors by training on a large dataset of point cloud neighborhoods to aid prediction of fitted tori. The points as tori method can hence easily be fine-tuned to a particular dataset. Conversely, our methods could also serve as a "warm start" for other, more challenging optimizations: in particular, our methods already return reasonable results given just a single instance of broken geometric data, and these solutions could be then be further optimized to match additional input data.

Conditional generation. The signed heat method and points as tori method are shown in this thesis to give more plausible shape completions than ordinary winding number, especially when the input data suffers from large amounts of missing data, because they use higher-order information (gradients, curvatures) of the observed surface to inform completion (Figures 5.13, 7.3). In this sense, these methods already implement a type of conditional completion or generation, since completions are dependent on the observed surface.

Following in the footsteps of the points as tori method, one could also envision building a data-driven version of the signed heat method: given a point cloud, a neural network could learn to predict a vector field whose directions best represent the gradient of the corresponding signed distance field. This vector field would then be integrated into an SDF as usual. Importantly, the choice of representation (*what* to learn) makes a huge difference in the efficacy and efficiency of learning-based methods. Directly learning a signed distance function is extremely challenging because signed distance functions follow a rigid structure: in particular, a nonlinear eikonal condition $\|\nabla\phi\|^2 = 1$, and precise Dirichlet and Neumann boundary conditions. (Section 5.5.3). In this case, learning a gradient vector field seems more promising, because it is more linear in the sense that it can closely be approximated by a linear process (Chapter 4). The recent work by Mello Rella et al. [2024] in fact proposes learning distance gradient-like vector fields for shape reconstruction, instead of scalar functions.

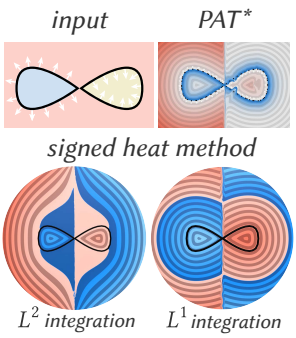
Winding number and signed distance as compressed (neural) representations. Occupancy functions and SDFs are in fact already very popular neural representations for a variety of learning-based geometric tasks, ranging from surface reconstruction to shape optimization [Mescheder et al. 2019; Park et al. 2019; Remelli et al. 2020]. More generally, neural fields (*i.e.*, neural network-parameterized implicit functions) have become popular within the last five years because they inherit the usual benefits of implicit representations — namely, straightforward function-based optimization that allows easy topology changes. However, these methods come at high cost; for example, when used for shape optimization, they must re-fit and differentiate naively through entire neural networks parameterizing global fields, which depend non-locally and non-linearly on field values, at every iteration [Yang et al. 2021; Mehta et al. 2022]. Typical

neural fields also tend to blur across shape boundaries [Belhe et al. 2023].

In contrast, winding numbers and the torus-based signed distance field developed in Chapter 5 generate global, smooth fields extremely cheaply from lightweight, explicit point-based representations using simple, analytical formulas that also admit analytical derivatives. I would suggest that for many applications, parameterizing functions through these analytical formulas rather than end-to-end neural networks is both more efficient, and provides important geometric structure especially if learning-based components are present (see also the discussion in Sections 5.5.5 and 5.5.6).

Winding number and signed distance also have valuable geometric properties that may be useful even in applications not specifically targeted for inside/outside or signed distance computation. For example, Chang et al. [2025] use jump harmonic functions to represent precise discontinuities in neural fields.

8.2 Lingering questions



***n*-ary distance.** All signed distance fields, including ours, can only give binary region classifications corresponding to the inside or outside of a given shape. Hence all signed distance methods fail to give n-ary region classifications. The inset figure shows the result of our signed distance algorithms on an example with three regions. The signed heat method tries and fails to represent a three-region segmentation, producing extremely warped level sets, and its optional L^1 integration produces a discontinuity far away from the true zero set. A preliminary 2D version of our points as tori method actually better interpolates the zero set since the method only looks at local point cloud patches rather than solving a global problem. Winding numbers are an elegant way to capture arbitrary region classifications, though it's not clear how to best design an n-ary distance function. On one hand, acquired data almost always does not represent nested regions or surfaces with inward-facing normals. However, such shapes do occur in modeling. It could be interesting to design an n-ary distance function, ideally without tedious tracking of multiple interfaces and associated SDFs, as is common in level set methods [Losasso et al. 2006].

Closed-form expressions for screened winding numbers. As mentioned in Section 3.2, it remains an open question whether there are closed-form expressions for the integrals of Yukawa potentials or other exponential kernels over (rational) parametric curves or triangles, analogous to closed-form expressions for winding numbers [Jacobson et al. 2013; Liu et al. 2025]. Of course, a screened Laplace equation can be solved on a triangle mesh as opposed to closed-form integration along the source curve (Figure 3.1). Outside of FEM-based methods, Madan and Levin [2022] do provide a quadrature scheme for pointwise LogSumExp that should provide good distance to perfect geometry.

Statistical geometry processing. The field of geometry processing was so named to evoke an analogy to signal processing, and classic signal processing and statistical methods continue to inform the solution of modern-day geometric problems (for example, the connection made between signed distance approximation and kernel density estimators in Chapter 3). Because surface reconstruction can be interpreted as regression, statistics remains a rich field that has yet to be fully mined towards gaining computational advantages in geometric problems. In turn, insights from geometric problems can inform modern statistical methods, such as generative modeling (see below).

For instance, as is always the case with ill-posed problems, we cannot guarantee exactly the true answer, which is fundamentally unknown and only one of many possible solutions that can explain the observed data. Some past work attempts to quantify "how wrong" an algorithm might be, though this usually requires a model for the source of error [Pauly et al. 2004; Hofsetz et al. 2004; Jenke et al. 2006; Pöthkow et al. 2011; Senin et al. 2021; Sellán and Jacobson 2022], and it is unclear how realistic these assumptions are. For winding numbers in particular, it is possible to do uncertainty analysis if you assume the winding number function arises from a Gaussian process [Sellán and Jacobson 2022; Chen et al. 2024a]. However, this analysis can be done because (regularized) winding number is purely linear. It is unclear how, or even if, to obtain covariances and probabilities for LogSumExp or Hopf-Cole-type distances, where a nonlinear logarithm is applied to the kernel sum, or for the signed heat method (which applies a nonlinear normalization step), or the points as tori method (which goes through a neural network, and then applies a formula with a nonlinear normalization).

Numerical approaches like randomized linear algebra — while seemingly timely in the age of big data, and tempting to explore in *e.g.* a one-pass or streaming setting — are probably less applicable for applied geometry problems, where one works almost exclusively in low dimensions (2D and 3D) rather than in higher-dimensional settings where randomized methods excel.

Generalization to other data-fitting tasks. Section 3.4 describes how the reason why naive convolutional distance formulas all fail to generalize from point clouds is the same reason why naive flow matching models fail to generalize from their discrete training data. The current state of the art is to train very large and expensive neural networks whose approximation error indirectly induces generalization, and I hope that future work will use geometric insights to improve the generalization of statistical and machine learning models without the extreme expense. Some authors have begun to explore geometry-based ideas [Bamberger et al. 2025; Wang et al. 2026], and I hope that future work can sidestep time-dependent ODEs entirely, and better address data-scarce applications (analogous to our improved generalization of signed distance from sparse point clouds).

Other generalizations of distance. SDFs are often the field of choice in field-based methods, such as level set methods (simulation, shape optimization) [Osher et al. 2004], and closest-point methods (for solving PDEs) [King et al. 2024]. One reason why SDFs are popular for

hybrid explicit-implicit methods is that they provide an intuitively meaningful scalar field that represents geometry through closest-point mapping. It would be interesting to explore the geometric properties of other types of distance fields, or even other implicit fields. For example, one project I explored but did not finish involved *maximum* distance fields, whose level sets generalize curves of constant width.

Chapter 9

Bibliography

Rémi Abgrall. 2022. Evaluating a distance function. arXiv:2211.02319 [math.NA] https://arxiv.org/abs/2211.02319 3.4

Anders Adamson and Marc Alexa. 2006. Anisotropic point set surfaces. In *Proceedings of the 4th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa* (Cape Town, South Africa) *(AFRIGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 7–13. https://doi.org/10.1145/1108590.1108592 3.4

M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C.T. Silva. 2001. Point set surfaces. In *Proceedings Visualization, 2001. VIS '01.* 21–29, 537. https://doi.org/10.1109/VISUAL.2001.964489 3.3

Nina Amenta and Yong Joo Kil. 2004. Defining point-set surfaces. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 264–270. https://doi.org/10.1145/1015706.1015713 3.4

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 214–223. 3.4

Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. 2006. Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput.* 22, 3 (March 2006), 181–193. https://doi.org/10.1007/s00371-006-0375-x 5.5.4

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 5.5.3

Jakob Andreas Bærentzen and Henrik 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 3.3

Jakob Andreas Bærentzen. 2005. Robust generation of signed distance fields from triangle meshes. In *Fourth International Workshop on Volume Graphics, 2005.* IEEE, 167–239. 2.4

Jacob Bamberger, Iolo Jones, Dennis Duncan, Michael M. Bronstein, Pierre Vandergheynst, and Adam Gosztolai. 2025. Carré du champ flow matching: better quality-generalisation tradeoff

in generative models. arXiv:2510.05930 [cs.LG] https://arxiv.org/abs/2510.05930 3.4, 8.2

Gavin Barill, Neil G. Dickson, Ryan Schmidt, David I. W. Levin, and Alec Jacobson. 2018. Fast winding numbers for soups and clouds. *ACM Trans. Graph.* 37, 4, Article 43 (July 2018), 12 pages. https://doi.org/10.1145/3197517.3201337 5.3, 7.1, 7.3

Yash Belhe, Michaël Gharbi, Matthew Fisher, Iliyan Georgiev, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Discontinuity-aware 2D neural fields. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 41, 6 (2023). https://doi.org/10.1145/3550454.3555484 8.1

Alexander Belyaev and Pierre-Alain Fayolle. 2020. An ADMM-based scheme for distance function approximation. *Numerical Algorithms* 84 (2020), 14 pages. https://doi.org/10.1007/s11075-019-00789-5 4.4, 4.5

Alexander Belyaev and Pierre-Alain Fayolle. 2024. Accuracy Improvements for Convolutional and Differential Distance Function Approximations. arXiv:2412.09200 [math.NA] https://arxiv.org/abs/2412.09200 3.2, 3.2

Alexander G. Belyaev and Pierre-Alain Fayolle. 2015. On Variational and PDE-Based Distance Function Approximations. *Computer Graphics Forum* 34, 8 (2015), 104–118. https://doi.org/10.1111/cgf.12611 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12611 3.1, 3.4

T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl. 1996. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering* 139, 1 (1996), 3–47. https://doi.org/10.1016/S0045-7825(96)01078-X 3.4

Yizhak Ben-Shabat and Stephen Gould. 2020. DeepFit: 3D Surface Fitting via Neural Network Weighted Least Squares. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I* (Glasgow, United Kingdom). Springer-Verlag, Berlin, Heidelberg, 20–34. https://doi.org/10.1007/978-3-030-58452-8_2 5.5.1

Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. 2022. DiGS : Divergence guided shape implicit neural representation for unoriented point clouds . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 19301–19310. https://doi.org/10.1109/CVPR52688.2022.01872 5.5.3

Carl M. Bender and Steven A. Orszag. 1999. *Advanced Mathematical Methods for Scientists and Engineers I: Asymptotic Methods and Perturbation Theory*. Springer New York, NY. https://doi.org/ttps://doi.org/10.1007/978-1-4757-3069-2 3.2

Matthew Berger, Andrea Tagliasacchi, Lee Seversky, Pierre Alliez, Joshua Levine, Andrei Sharf, and Claudio Silva. 2014. State of the Art in Surface Reconstruction from Point Clouds. In *EUROGRAPHICS star report (EUROGRAPHICS star report, Vol. 1)*. Strasbourg, France, 161–185. https://doi.org/10.2312/egst.20141040 2.3

Nicole Berline, Ezra Getzler, and Michèle Vergne. 1992. *Heat Kernels and Dirac Operators* (1 ed.). Springer Berlin, Heidelberg. 4.1, 4.2

Paul J Besl and Ramesh C Jain. 1988. Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10, 2 (1988), 167–192. https://doi.org/10.1109/34.3881 5.5.4

Jack Binysh and Gareth P Alexander. 2018. Maxwell's theory of solid angle and the construction of knotted fields. *Journal of Physics A: Mathematical and Theoretical* 51, 38 (aug 2018), 385202. https://doi.org/10.1088/1751-8121/aad8c6 2.3.3

Giulio Biroli, Tony Bonnaire, Valentin de Bortoli, and Marc Mézard. 2024. Dynamical Regimes of Diffusion Models. *Nature Communications* 15 (2024). Issue 1. https://doi.org/10.1038/s41467-024-54281-3 3.4

Arpan Biswas and Vadim Shapiro. 2004. Approximate distance fields with non-vanishing gradients. *Graphical Models* 66, 3 (2004), 133–159. https://doi.org/10.1016/j.gmod.2004.01.003 3.4

Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. 2020. Accurately computing the log-sum-exp and softmax functions. *IMA J. Numer. Anal.* 41, 4 (08 2020), 2311–2330. https://doi.org/10.1093/imanum/draa038 arXiv:https://academic.oup.com/imajna/article-pdf/41/4/2311/40758053/draa038.pdf 5.1.3

James F. Blinn. 1982. A Generalization of Algebraic Surface Drawing. *ACM Trans. Graph.* 1, 3 (July 1982), 235–256. https://doi.org/10.1145/357306.357310 3.4

Jules Bloomenthal. 1997. Bulge Elimination in Convolution Surfaces. *Computer Graphics Forum* 16, 1 (1997), 31–41. https://doi.org/10.1111/1467-8659.114 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.114 3.4

Jules Bloomenthal and Ken Shoemake. 1991. Convolution surfaces. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. Association for Computing Machinery, New York, NY, USA, 251–256. https://doi.org/10.1145/122718.122757 3.4

Jean-Daniel Boissonnat and Frédéric Cazals. 2002. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Computational Geometry* 22, 1 (2002), 185–203. https://doi.org/10.1016/S0925-7721(01)00048-7 16th ACM Symposium on Computational Geometry. 3.3, 3.4

David Bommes and Leif Kobbelt. 2007. Accurate Computation of Geodesic Distance Fields for Polygonal Curves on Triangle Meshes. *VMV*, 151–160. 2.4, 4.5

C.A. Brebbia, J.C.F. Telles, and L. Wrobel. 1984. *Boundary Element Techniques: Theory and Applications in Engineering*. Springer Berlin Heidelberg. 5

Alan Brunton and Lubna Abu Rmaileh. 2021. Displaced Signed Distance Fields for Additive Manufacturing. *ACM Trans. Graph.* 40, 4, Article 179 (jul 2021), 13 pages. https://doi.org/10.1145/3450626.3459827 2.4

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 4.3

Fatih Calakli and Gabriel Taubin. 2011. SSD: Smooth Signed Distance Surface Reconstruction. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, The Eurographics Association and Blackwell Publishing Ltd., 1993–2002. https://doi.org/10.1111/j.1467-8659.2011.02058.x 2.4, 4.4, 5.5.3

Xingping Cao, Neelima Shrikhande, and Gongzhu Hu. 1994. Approximate orthogonal distance regression method for fitting quadric surfaces to range data. *Pattern Recognition Letters* 15, 8 (1994), 781–796. https://doi.org/10.1016/0167-8655(94)90006-X 5.5.4

Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramèr, Borja Balle, Daphne Ippolito, and Eric Wallace. 2023. Extracting training data from diffusion models. In *Proceedings of the 32nd USENIX Conference on Security Symposium* (Anaheim, CA, USA) *(SEC '23)*. USENIX Association, USA, Article 294, 18 pages. 3.4

Jonathan C. Carr, Rick K. Beatson, J. B. Cherrie, T. J. Mitchell, W. Richard Fright, Bruce C. McCallum, and T. R. Evans. 2001. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 67–76. https://doi.org/10.1145/383259.383266 3.4

Frédéric Cazals and Marc Pouget. 2005. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design* 22, 2 (2005), 121–146. https://doi.org/10.1016/j.cagd.2004.09.004 5.5.4

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 4.4

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D Data in Indoor Environments. In *2017 International Conference on 3D Vision (3DV)*. 667–676. https://doi.org/10.1109/3DV.2017.00081 2.3.3

Yue Chang, Mengfei Liu, Zhecheng Wang, Peter Yichen Chen, and Eitan Grinspun. 2025. Lifting the Winding Number: Precise Discontinuities in Neural Fields for Physics Simulation. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 25, 11 pages. https://doi.org/10.1145/3721238.3730597 8.1

Hanyu Chen, Bailey Miller, and Ioannis Gkioulekas. 2024a. 3D Reconstruction with Fast Dipole Sums. *ACM Trans. Graph.* 43, 6, Article 192 (Nov. 2024), 19 pages. https://doi.org/10.1145/3687914 (document), 3.1, 3.3, 3.3, 3.4, 7.2, 8.2

Jiong Chen, Florian Schaefer, and Mathieu Desbrun. 2024b. Lightning-fast Method of Fundamental Solutions. *ACM Trans. Graph.* 43, 4, Article 77 (July 2024), 16 pages. https://doi.org/10.1145/3658199 5.4.1

Vladimir Chernov and Yuli B. Rudyak. 2009. On generalized winding numbers. *St. Petersburg Mathematical Journal* 20, 5 (2009), 837–849. 2.3.3

Cheng Chi and Shuran Song. 2021. GarmentNets: Category-Level Pose Estimation for Garments via Canonical Space Shape Completion. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 3304–3313. https://doi.org/10.1109/ICCV48922.2021.00331 2.3.3

Julian Chibane, Aymen Mir, and Gerard Pons-Moll. 2020. Neural unsigned distance fields for implicit function learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1816, 15 pages. 5.5.3

David RJ Chillingworth. 1972. Winding numbers on surfaces, I. *Math. Ann.* 196, 3 (1972), 218–249. 2.3.3

Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. 2016. A Large Dataset of Object Scans. *arXiv:1602.02481* (2016). (document), 4.7, 4.9

Youngsoo Choi, Siu Wun Cheung, Youngkyu Kim, Ping-Hsuan Tsai, Alejandro N. Diaz, Ivan Zanardi, Seung Whan Chung, Dylan Matthew Copeland, Coleman Kendrick, William Anderson, Traian Iliescu, and Matthias Heinkenschloss. 2025. Defining Foundation Models for Computational Science: A Call for Clarity and Rigor. arXiv:2505.22904 [cs.LG] https://arxiv.org/abs/2505.22904 5.5.5

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. 4.3

David Coeurjolly and Jérémy Levallois. 2015. VolGallery. https://github.com/dcoeurjo/VolGallery. (document), 4.5

David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2004. Variational shape approximation. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 905–914. https://doi.org/10.1145/1015706.1015817 5.5.4

Guillaume Coiffier and Louis Béthune. 2024. 1-Lipschitz Neural Distance Fields. *Computer Graphics Forum* 43, 5 (2024), e15128. https://doi.org/10.1111/cgf.15128 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.15128 5.5.3

Ronald R. Coifman and Stéphane Lafon. 2006. Diffusion maps. *Applied and Computational Harmonic Analysis* 21, 1 (2006), 5–30. https://doi.org/10.1016/j.acha.2006.04.006 Special Issue: Diffusion Maps and Wavelets. 5.5.1

Julian D Cole. 1951. On a quasi-linear parabolic equation occurring in aerodynamics. *Quarterly of applied mathematics* 9, 3 (1951), 225–236. 3.1

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality streamable free-viewpoint video. *ACM Trans. Graph.* 34, 4, Article 69 (July 2015), 13 pages. https://doi.org/10.1145/

2766945 2.3.3

Michael G Crandall and Pierre-Louis Lions. 1983. Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American mathematical society* 277, 1 (1983), 1–42. 3.1

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013a. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses* (Anaheim, California) *(SIGGRAPH '13)*. Association for Computing Machinery, New York, NY, USA, Article 7, 126 pages. https://doi.org/10.1145/2504435.2504442 2.2, 2.2

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013b. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Trans. Graph.* 32, 5, Article 152 (Oct. 2013), 11 pages. https://doi.org/10.1145/2516971.2516977 (document), 3.1, 4.2, 4.3, 4.11, 4.4, 4.5, 4.5, 7.1

George Dantzig. 1963. *Linear programming and extensions*. Princeton university press. 4.4

Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *International Conference on Learning Representations (ICLR)*. 5.5.6

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5.5.6

Denis Demidov. 2025. AMGCL. github.com/ddemidov/amgcl. 7.2

Mathieu Desbrun, Eva Kanso, and Yiying Tong. 2005. Discrete differential forms for computational modeling. In *ACM SIGGRAPH 2005 Courses* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 7–es. https://doi.org/10.1145/1198555.1198666 2.2

Miles Detrixhe, Frédéric Gibou, and Chohong Min. 2013. A parallel fast sweeping method for the Eikonal equation. *J. Comput. Phys.* 237 (2013), 46–55. https://doi.org/10.1016/j.jcp.2012.11.042 7.3

Tamal K. Dey and Jian Sun. 2005. An adaptive MLS surface for reconstruction with guarantees. In *Proceedings of the Third Eurographics Symposium on Geometry Processing* (Vienna, Austria) *(SGP '05)*. Eurographics Association, Goslar, DEU, 43–es. 3.4

Alexandre Djerbetian. 2016. *Tangent Vector Fields on Triangulated Surfaces - An Edge-Based Approach*. Master's thesis. Computer Science Department, Technion. 4.3

Manfredo P. do Carmo. 1992. *Riemannian Geometry* (1 ed.). 2.2

Manfredo P. Do Carmo. 2016. *Differential Geometry of Curves & Surfaces* (second ed.). Dover, New York. 2.3.3

Tevian Dray. 2014. *Differential Forms and the Geometry of General Relativity* (1 ed.). Taylor & Francis Group. 2.2

Jan Dvořák, Zuzana Káčereková, Petr Vaněček, Lukáš Hruda, and Libor Váša. 2022. As-rigid-as-possible volume tracking for time-varying surfaces. *Computers & Graphics* 102 (2022),

329–338. https://doi.org/10.1016/j.cag.2021.10.015 2.3.3

David Eberly. 2020. Fitting 3D Data with a Torus. https://www.geometrictools.com/Documentation/TorusFitting.pdf. 5.5.4

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 4.4, 4.5

Jeff Erickson. 2019. *Algorithms*. Jeff Erickson. http://jeffe.cs.illinois.edu/teaching/algorithms/ 4.4

Jeff Erickson and Kim Whittlesey. 2005. Greedy optimal homotopy and homology generators. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Vancouver, British Columbia) *(SODA '05)*. Society for Industrial and Applied Mathematics, USA, 1038–1046. 2.3.2

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J. Mitra, and Michael Wimmer. 2020. Points2Surf Learning Implicit Surfaces from Point Clouds. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V* (Glasgow, United Kingdom). Springer-Verlag, Berlin, Heidelberg, 108–124. https://doi.org/10.1007/978-3-030-58558-7_7 5.5.4

Leonhard Euler. 1781. De mensura angulorum solidorum. *Acta Academiae Scientiarum Imperialis Petropolitanae* (1781), 31–54. 2.3.3

Lawrence C. Evans. 1998. *Partial Differential Equations*. American Mathematical Society. 3.1, 3.2

Olivier D Faugeras. 1983. Segmentation of Range Data into Planar and Quadric Patches. *Proceedings of Third Computer Vision and Pattern Recognition Conference* (1983). 5.5.4

Nicole Feng. 2025. signed-heat-python. github.com/nzfeng/signed-heat-python. 5.3

Nicole Feng and Keenan Crane. 2024. A Heat Method for Generalized Signed Distance. *ACM Trans. Graph.* 43, 4, Article 92 (jul 2024), 16 pages. https://doi.org/10.1145/3658220 4.3, 4.3, 6.4.1

Nicole Feng, Mark Gillespie, and Keenan Crane. 2023. Winding Numbers on Discrete Surfaces. *ACM Trans. Graph.* 42, 4, Article 36 (July 2023), 17 pages. https://doi.org/10.1145/3592401 2.3.3, 6.1, 7.1

Nicole Feng, Ioannis Gkioulekas, and Keenan Crane. 2026. Points as Tori: Fast Pointwise Signed Distance for Point Clouds. *ACM Trans. Graph.* 45, 4, Article 53 (jul 2026), 24 pages. https://doi.org/10.1145/3811385 3.1, 3.1, 3.1, 5.1, 5.1.1

Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. 2005. Robust moving least-squares fitting with sharp features. In *ACM SIGGRAPH 2005 Papers* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 544–552. https://doi.org/10.1145/1186822.1073227 5

Simon Fuhrmann and Michael Goesele. 2014. Floating scale surface reconstruction. *ACM Trans. Graph.* 33, 4, Article 46 (July 2014), 11 pages. https://doi.org/10.1145/2601097.2601163 3.4

Weiguo Gao and Ming Li. 2024. How Do Flow Matching Models Memorize and Generalize in Sample Data Subspaces? *CoRR* abs/2410.23594 (2024). https://doi.org/10.48550/arXiv.2410.23594 3.4

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 5.5.1, 5.5.6

Carl Friedrich Gauss. 1838. General Theory of Terrestrial Magnetism. (1838). https://hgss.copernicus.org/articles/5/11/2014/hgss-5-11-2014.pdf 2.3.3

Izrail Moiseevitch Gelfand, Richard A Silverman, et al. 2000. *Calculus of variations*. Courier Corporation. 4.4

Mark Gillespie, Nicholas Sharp, and Keenan Crane. 2021. Integer coordinates for intrinsic geometry processing. *ACM Trans. Graph.* 40, 6, Article 252 (Dec. 2021), 13 pages. https://doi.org/10.1145/3478513.3480522 (document), 4.3, 4.6, 6.2

Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. 1984. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.* 18, 3 (Jan. 1984), 213–222. https://doi.org/10.1145/964965.808601 2.3.3

Craig Gotsman and Daniel Keren. 1998. Tight Fitting of Convex Polyhedral Shapes. *International Journal of Shape Modeling* 04, 03n04 (1998), 111–126. https://doi.org/10.1142/S021865439800009X arXiv:https://doi.org/10.1142/S021865439800009X 5.4.4

Olivier Gourmel, Loic Barthe, Marie-Paule Cani, Brian Wyvill, Adrien Bernhardt, Mathias Paulin, and Herbert Grasberger. 2013. A gradient-based implicit blend. *ACM Trans. Graph.* 32, 2, Article 12 (April 2013), 12 pages. https://doi.org/10.1145/2451236.2451238 3.4

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 5.5.3

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 2018. A Papier-Mache Approach to Learning 3D Surface Generation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 216–224. https://doi.org/10.1109/CVPR.2018.00030 5.5.4

Xiangming Gu, Chao Du, Tianyu Pang, Chongxuan Li, Min Lin, and Ye Wang. 2025. On Memorization in Diffusion Models. arXiv:2310.02664 [cs.LG] https://arxiv.org/abs/2310.02664 3.4

Xianfeng Gu and Shing-Tung Yau. 2003. Global conformal surface parameterization. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Aachen,

Germany) *(SGP '03)*. Eurographics Association, Goslar, DEU, 127–137. 2.3.3

Karthik S. Gurumoorthy and Anand Rangarajan. 2009. A Schrödinger Equation for the Fast Computation of Approximate Euclidean Distance Functions. In *Scale Space and Variational Methods in Computer Vision*, Xue-Cheng Tai, Knut Mørken, Marius Lysaker, and Knut-Andreas Lie (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 100–111. 3.2, 3.3, 3.4

Eric Haines. 1994. *Point in polygon strategies.* Academic Press Professional, Inc., USA, 24–46. 2.3.3, 2.4, 4.5

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. 5.2

Christian Hofsetz, Kim Ng, George Chen, Peter McGuinness, Nelson Max, and Yang Liu. 2004. Image-based rendering of range data with estimated depth uncertainty. *IEEE Computer Graphics and Applications* 24, 4 (2004), 34–41. https://doi.org/10.1109/MCG.2004.8 8.2

Eberhard Hopf. 1950. The partial differential equation $u_t + uu_x = mu_{xx}$. *Communications on Pure and Applied mathematics* 3, 3 (1950), 201–230. 3.1

Yixin Hu, Qingnan Zhou, Xifeng Gao, Alec Jacobson, Denis Zorin, and Daniele Panozzo. 2018. Tetrahedral Meshing in the Wild. *ACM Trans. Graph.* 37, 4, Article 60 (July 2018), 14 pages. https://doi.org/10.1145/3197517.3201353 2.3.3

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 32, 11 pages. https://doi.org/10.1145/3641519.3657428 5.4.1

Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. 2023. Neural Kernel Surface Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4369–4379. 3.4, 5.4.1

Saar Huberman, Amit Bracha, and Ron Kimmel. 2023. Deep Accurate Solver for the Geodesic Problem. In *Scale Space and Variational Methods in Computer Vision: 9th International Conference, SSVM 2023, Santa Margherita Di Pula, Italy, May 21–25, 2023, Proceedings* (Santa Margherita di Pula, Italy). Springer-Verlag, Berlin, Heidelberg, 288–300. https://doi.org/10.1007/978-3-031-31975-4_22 2.4

Pierre Hubert-Brierre, Eric Guérin, Adrien Peytavie, and Eric Galin. 2025. Accelerating Signed Distance Functions. *Computer Graphics Forum* 44, 7 (2025), e70258. https://doi.org/10.1111/cgf.70258 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70258 2.4

Stephen P. Humphries and Dennis Johnson. 1989. A Generalization of Winding Number Functions on Surfaces. *Proceedings of the London Mathematical Society* s3-58, 2 (1989), 366–386. https://doi.org/10.1112/plms/s3-58.2.366 arXiv:https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s3-58.2.366 2.3.3

Alec Jacobson, Ladislav Kavan, and Olga Sorkine-Hornung. 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph.* 32, 4, Article 33 (July 2013),

12 pages. https://doi.org/10.1145/2461912.2461916 2.3.3, 2.3.3, 7.1, 8.2

Anubhav Jain, Yuya Kobayashi, Takashi Shibuya, Yuhta Takida, Nasir Memon, Julian Togelius, and Yuki Mitsufuji. 2025. Classifier-Free Guidance inside the Attraction Basin May Cause Memorization. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12871–12879. https://doi.org/10.1109/CVPR52734.2025.01201 3.4

Philipp Jenke, Michael Wand, Martin Bokeloh, Andreas Schilling, and Wolfgang Straßer. 2006. Bayesian Point Cloud Reconstruction. *Computer Graphics Forum* 25, 3 (2006), 379–388. https://doi.org/10.1111/j.1467-8659.2006.00957.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2006.00957.x 8.2

Yiwen Ju, Xingyi Du, Qingnan Zhou, Nathan Carr, and Tao Ju. 2024. Adaptive grid generation for discretizing implicit complexes. *ACM Trans. Graph.* 43, 4, Article 82 (July 2024), 17 pages. https://doi.org/10.1145/3658215 5.3

Hyun-Seok Jung, Seong-Hyeon Kweon, and Seung-Hyun Yoon. 2025. GeoText: Geodesic-Based 3D Text Generation on Triangular Meshes. *Symmetry* 17, 10 (2025). https://doi.org/10.3390/sym17101727 8

Mason Kamb and Surya Ganguli. 2025. An analytic theory of creativity in convolutional diffusion models. arXiv:2412.20292 [cs.LG] https://arxiv.org/abs/2412.20292 3.4

Christina Karam, Kenjiro Sugimoto, and Keigo Hirakawa. 2019. Fast Convolutional Distance Transform. *IEEE Signal Processing Letters* 26, 6 (2019), 853–857. https://doi.org/10.1109/LSP.2019.2910466 3.3, 3.4

Satish Kaveti, Eam Khwang Teoh, and Han Wang. 1996. Second-order implicit polynomials for segmentation of range images. *Pattern Recognition* 29, 6 (1996), 937–949. https://doi.org/10.1016/0031-3203(95)00137-9 5.5.4

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson Surface Reconstruction. In *Symposium on Geometry Processing (SGP '06)*. Eurographics Association, 61–70. 2.3.3, 3.1, 4.4, 5.4.1

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 (document), 1, 5.11, 5.3, 7.1

Michael Kazhdan, Allison Klein, Ketan Dalal, and Hugues Hoppe. 2007. Unconstrained isosurface extraction on arbitrary octrees. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Barcelona, Spain) *(SGP '07)*. Eurographics Association, Goslar, DEU, 125–133. 5.3

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4, Article 139 (July 2023), 14 pages. https://doi.org/10.1145/3592433 5.2

Daniel Keren and Craig Gotsman. 1999. Fitting curves and surfaces with constrained implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21, 1 (1999), 31–41. https://doi.org/10.1109/34.745731 5.4.4

Ron Kimmel and James A. Sethian. 1998. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences* 95, 15 (1998), 8431–8435. https://doi.org/10.1073/pnas.95.15.8431 arXiv:https://www.pnas.org/doi/pdf/10.1073/pnas.95.15.8431 2.4, 4.4, 7.3

Nathan King, Haozhe Su, Mridul Aanjaneya, Steven Ruuth, and Christopher Batty. 2024. A Closest Point Method for PDEs on Manifolds with Interior Boundary Conditions for Geometry Processing. *ACM Trans. Graph.* 43, 5, Article 159 (Aug. 2024), 26 pages. https://doi.org/10.1145/3673652 8.2

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A Big CAD Model Dataset For Geometric Deep Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5.1.2

Ravikrishna Kolluri. 2008. Provably good moving least squares. *ACM Trans. Algorithms* 4, 2, Article 18 (May 2008), 25 pages. https://doi.org/10.1145/1361192.1361195 3.3, 3.4

G. Kreisselmeier and R. Steinhauser. 1980. Systematic Control Design By Optimizing A Vector Performance Index. In *Computer Aided Design of Control Systems*, M. A. Cuenod (Ed.). Pergamon, 113–117. https://doi.org/10.1016/B978-0-08-024488-4.50022-X 3.2, 3.4

Pavel A Krutitskii. 2001. The jump problem for the Laplace equation. *Applied Mathematics Letters* 14, 3 (2001), 353–358. 2.3.3

Mathieu Ladeuil, Marc Trabucato, Alexis Vaisse, and Noura Faraj. 2026. Construction of clustered HLOD with As-Simplified-As-Possible boundaries. *Computer Graphics Forum* n/a, n/a (2026), e70380. https://doi.org/10.1111/cgf.70380 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70380 8

Joseph Louis Lagrange. 1798. *Solutions de quelques problèmes relatifs aux triangles sphériques, avec une analyse complète de ces triangles.* 2.3.3

Peter Lancaster and Kes Salkauskas. 1981. Surfaces generated by moving least squares methods. *Mathematics of computation* 37, 155 (1981), 141–158. 3.4

John M. Lee. 2012. *Introduction to Smooth Manifolds* (2 ed.). Springer New York, NY. 2.2

Eric Lengyel. 2017. GPU-Centered Font Rendering Directly from Glyph Outlines. *Journal of Computer Graphics Techniques (JCGT)* 6, 2 (14 June 2017), 31–47. http://jcgt.org/published/0006/02/02/ 2.3.3

Christian Lennerz and Elmar Schömer. 2002. Efficient distance computation for quadratic curves and surfaces. In *Geometric Modeling and Processing. Theory and Applications. GMP 2002. Proceedings.* 60–69. https://doi.org/10.1109/GMAP.2002.1027497 5.5.4

David Levin. 1998. The Approximation Power of Moving Least-Squares. *Math. Comp.* 67, 224 (1998), 1517–1531. http://www.jstor.org/stable/2584860 3.4

Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. 2000. The digital Michelangelo project: 3D scanning of large statues. In *Proceedings of the 27th*

Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00). ACM Press/Addison-Wesley Publishing Co., USA, 131–144. https://doi.org/10.1145/344779.344849 5.3

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. 2.4

Guying Lin, Lei Yang, Congyi Zhang, Hao Pan, Yuhan Ping, Guodong Wei, Taku Komura, John Keyser, and Wenping Wang. 2025. Patch-Grid: An Efficient and Feature-Preserving Neural Implicit Surface Representation. *ACM Trans. Graph.* 44, 2, Article 16 (April 2025), 21 pages. https://doi.org/10.1145/3727142 5.5.3

Yaron Lipman. 2021. Phase Transitions, Distance Functions, and Implicit Neural Representations. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 6702–6712. https://proceedings.mlr.press/v139/lipman21a.html 3.1, 3.1, 3.4, 5.5.3

Shibo Liu, Ligang Liu, and Xiao-Ming Fu. 2025. Closed-form Generalized Winding Numbers of Rational Parametric Curves for Robust Containment Queries. *ACM Trans. Graph.* 44, 4, Article 75 (July 2025), 9 pages. https://doi.org/10.1145/3730886 8.2

Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. 2021. Deep Implicit Moving Least-Squares Functions for 3D Reconstruction. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1788–1797. https://doi.org/10.1109/CVPR46437.2021.00183 3.4

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2022. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. arXiv:2209.03003 [cs.LG] https://arxiv.org/abs/2209.03003 3.4

Xiao-Ming Liu, Lei Yang, Jun-Hai Yong, He-Jin Gu, and Jia-Guang Sun. 2009. A torus patch approximation approach for point projection on surfaces. *Computer Aided Geometric Design* 26, 5 (2009), 593–598. https://doi.org/10.1016/j.cagd.2009.01.004 5.5.4

Daniel Simões Lopes, Miguel Tavares da Silva, Jorge Alberto Cadete Ambrósio, and João Paulo Flores Fernandes. 2010. A mathematical framework for rigid contact detection between quadric and superquadric surfaces. 24, 3 (2010), 255–280. https://doi.org/10.1007/s11044-010-9220-0 5.5.4

Frank Losasso, Tamar Shinar, Andrew Selle, and Ronald Fedkiw. 2006. Multiple interacting liquids. *ACM Trans. Graph.* 25, 3 (July 2006), 812–819. https://doi.org/10.1145/1141911.1141960 8.2

Gus K Lott III. 2014. Direct Orthogonal Distance to Quadratic Surfaces in 3D. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 9 (2014), 1888–1892. https://doi.org/10.1109/TPAMI.2014.2302451 5.5.4

Gabor Lukács, Ralph Martin, and Dave Marshall. 1998. Faithful Least-Squares Fitting of Spheres, Cylinders, Cones and Tori for Reliable Segmentation. In *Proceedings of the 5th European Con-*

ference on Computer Vision-Volume I - Volume I (ECCV '98). Springer-Verlag, Berlin, Heidelberg, 671–686. 5.5.4

Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. 2021. Neural-Pull: Learning Signed Distance Function from Point clouds by Learning to Pull Space onto Surface. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 7246–7257. https://proceedings.mlr.press/v139/ma21b.html 5.5.3

Richard Henri MacNeal. 1949. *The Solution of Partial Differential Equations by Means of Electrical Networks*. Ph. D. Dissertation. California Institute of Technology. https://thesis.caltech.edu/1529/ 2.2

Abhishek Madan and David I. W. Levin. 2022. Fast evaluation of smooth distance constraints on co-dimensional geometry. *ACM Trans. Graph.* 41, 4, Article 68 (July 2022), 17 pages. https://doi.org/10.1145/3528223.3530093 (document), 3.2, 3.3, 3.4, 5.2, 8.2

Cedric Martens and Mikhail Bessmeltsev. 2025. One-Shot Method for Computing Generalized Winding Numbers. *Computer Graphics Forum* 44, 5 (2025), e70194. https://doi.org/10.1111/cgf.70194 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70194 2.3.3

Dimas Martínez and Jorge Estrada-Sarlabous. 2003. ON THE DISTANCE FROM A POINT TO A QUADRIC SURFACE. *Revista Investigación Operacional* 24 (01 2003). 5.5.4

James Clerk Maxwell. 1881. *A Treatise on Electricity and Magnetism*. Vol. II. Oxford University Press. 2.3.3

Margaret McIntyre and Grant Cairns. 1993. A new formula for winding number. *Geometriae Dedicata* 46, 2 (1993), 149–159. 2.3.3

Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. 2022. A Level Set Theory for Neural Implicit Evolution under Explicit Flows. In *European Conference on Computer Vision*. Springer, 711–729. 8.1

Edoardo Mello Rella, Ajad Chhatkuli, Ender Konukoglu, and Luc Van Gool. 2024. Neural Vector Fields for Implicit Surface Representation and Inference. *Int. J. Comput. Vision* 133, 4 (Oct. 2024), 1855–1878. https://doi.org/10.1007/s11263-024-02251-z 8.1

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4455–4465. https://doi.org/10.1109/CVPR.2019.00459 8.1

Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. 1987. The Discrete Geodesic Problem. *SIAM J. Comput.* 16, 4 (1987), 647–668. https://doi.org/10.1137/0216045 arXiv:https://doi.org/10.1137/0216045 4.4, 4.5, 7.1

Bryan S. Morse, Terry S. Yoo, Penny Rheingans, David T. Chen, and K. R. Subramanian. 2005. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. In *ACM SIGGRAPH 2005 Courses* (Los Angeles, California) *(SIGGRAPH '05)*.

Association for Computing Machinery, New York, NY, USA, 78–es. https://doi.org/10.1145/1198555.1198645 3.4

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 2.4

James R Munkres. 1984. *Elements of Algebraic Topology* (1st. ed.). CRC press. 2.3.2, 2.3.2

Shigeru Muraki. 1991. Volumetric shape description of range data using "Blobby Model". In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '91)*. Association for Computing Machinery, New York, NY, USA, 227–235. https://doi.org/10.1145/122718.122743 3.4

Ken Museth. 2017. Novel algorithm for sparse and parallel fast sweeping: efficient computation of sparse signed distance fields. In *ACM SIGGRAPH 2017 Talks* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 74, 2 pages. https://doi.org/10.1145/3084363.3085093 7.3

Ken Museth, David E. Breen, Ross T. Whitaker, and Alan H. Barr. 2002. Level set surface editing operators. *ACM Trans. Graph.* 21, 3 (July 2002), 330–338. https://doi.org/10.1145/566654.566585 2.4

Ashish Myles, Nico Pietroni, and Denis Zorin. 2014. Robust field-aligned global parametrization. *ACM Trans. Graph.* 33, 4, Article 135 (July 2014), 14 pages. https://doi.org/10.1145/2601097.2601154 7.1

Lea Müller, Ahmed A. A. Osman, Siyu Tang, Chun-Hao P. Huang, and Michael J. Black. 2021. On Self-Contact and Human Pose. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 9985–9994. https://doi.org/10.1109/CVPR46437.2021.00986 2.3.3

Vinh Phu Nguyen, Timon Rabczuk, Stéphane Bordas, and Marc Duflot. 2008. Meshless methods: A review and computer implementation aspects. *Mathematics and Computers in Simulation* 79, 3 (2008), 763–813. https://doi.org/10.1016/j.matcom.2008.01.003 3.4

Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. 2005. Multi-level partition of unity implicits. In *ACM SIGGRAPH 2005 Courses* (Los Angeles, California) *(SIGGRAPH '05)*. Association for Computing Machinery, New York, NY, USA, 173–es. https://doi.org/10.1145/1198555.1198649 3.4

Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. 2003. A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. In *2003 Shape Modeling International.* 153–161. https://doi.org/10.1109/SMI.2003.1199611 3.4

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. 2.4

Stanley Osher, Ronald Fedkiw, and K Piechor. 2004. Level set methods and dynamic implicit surfaces. *Appl. Mech. Rev.* 57, 3 (2004), B15–B15. 2.4, 7.3, 8.2

A. Cengiz Öztireli, Gaël Guennebaud, and Markus Gross. 2009. Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression. *Computer Graphics Forum* 28, 2 (2009), 493–501. https://doi.org/10.1111/j.1467-8659.2009.01388.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2009.01388.x 3.3, 3.4, 5

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)*. 8.1

Yesom Park, Taekyung Lee, Jooyoung Hahn, and Myungjoo Kang. 2023. p-Poisson surface reconstruction in curl-free flow from point clouds. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 2626, 22 pages. 5.5.3

Mark Pauly, Niloy J. Mitra, and Leonidas J. Guibas. 2004. Uncertainty and variability in point cloud surface data. In *Proceedings of the First Eurographics Conference on Point-Based Graphics* (Switzerland) *(SPBG'04)*. Eurographics Association, Goslar, DEU, 77–84. 8.2

Jakiw Pidstrigach. 2022. Score-based generative models detect manifolds. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 2598, 14 pages. 3.4

Konstantin Poelke and Konrad Polthier. 2016. Boundary-aware hodge decompositions for piecewise constant vector fields. *Comput. Aided Des.* 78, C (Sept. 2016), 126–136. https://doi.org/10.1016/j.cad.2016.05.004 2.3.2

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. 6.4.1

Albert Pumarola, Artsiom Sanakoyeu, Lior Yariv, Ali Thabet, and Yaron Lipman. 2022. VisCo grids: surface reconstruction with viscosity and coarea grids. In *Proceedings of the 36th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, Article 1313, 12 pages. 5.5.3

Kai Pöthkow, Britta Weber, and Hans-Christian Hege. 2011. Probabilistic Marching Cubes. *Computer Graphics Forum* 30, 3 (2011), 931–940. https://doi.org/10.1111/j.1467-8659.2011.01942.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2011.01942.x 8.2

Blaine Quackenbush and Paul J. Atzberger. 2025. Transferable foundation models for geometric tasks on point cloud representations: geometric neural operators. *Machine Learning: Science and Technology* 6, 4 (nov 2025), 045045. https://doi.org/10.1088/2632-2153/ae1bf8 5.5.5

Inigo Quilez. 2008. Raymarching Signed Distance Fields. https://iquilezles.org/

articles/raymarchingdf/. 2.4

Inigo Quilez. 2025. 3D distance and gradient functions - 2025. https://iquilezles.org/articles/distgradfunctions3d/. 5.1.1

Bruce L Reinhart. 1960. The winding number on two manifolds. In *Annales de l'institut Fourier*, Vol. 10. 271–283. 2.3.3

Bruce L Reinhart. 1963. Further remarks on the winding number. In *Annales de l'institut Fourier*, Vol. 13. 155–160. 2.3.3

Edoardo Remelli, Artem Lukoianov, Stephan R. Richter, Benoît Guillard, Timur Bagautdinov, Pierre Baque, and Pascal Fua. 2020. MeshSDF: differentiable iso-surface extraction. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1884, 11 pages. 8.1

Marzia Riso, Giacomo Nazzaro, Enrico Puppo, Alec Jacobson, Qingnan Zhou, and Fabio Pellacini. 2022. BoolSurf: Boolean Operations on Surfaces. *ACM Trans. Graph.* 41, 6, Article 247 (Nov. 2022), 13 pages. https://doi.org/10.1145/3550454.3555466 2.3.2, 2.3.3

Angel D. Sappa and Mohammad Rouhani. 2009. Efficient distance estimation for fitting implicit quadric surfaces. In *Proceedings of the 16th IEEE International Conference on Image Processing* (Cairo, Egypt) *(ICIP'09)*. IEEE Press, 3485–3488. 5.5.4

Rohan Sawhney. 2021. *FCPW: Fastest Closest Points in the West.* 2.4, 4.5, 5.3, 7.3

Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *ACM Trans. Graph.* 37, 1, Article 5 (dec 2017), 14 pages. https://doi.org/10.1145/3132705 2.3.3

Christopher Scarvelis, Haitz Sáez de Ocáriz Borde, and Justin Solomon. 2023. Closed-Form Diffusion Models. arXiv:2310.12395 [cs.LG] https://arxiv.org/abs/2310.12395 3.4

Günter Schwarz. 2006. *Hodge Decomposition-A method for solving boundary value problems.* Springer. 2.2

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 8.2

Nicola Senin, Sofia Catalucci, Michele Moretti, and Richard K. Leach. 2021. Statistical point cloud model to investigate measurement uncertainty in coordinate metrology. *Precision Engineering* 70 (2021), 44–62. https://doi.org/10.1016/j.precisioneng.2021.01.008 8.2

Manu Sethi, Anand Rangarajan, and Karthik Gurumoorthy. 2012. The Schrödinger distance transform (SDT) for point-sets and curves. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 198–205. 3.3, 3.4

Nicholas Sharp and Keenan Crane. 2020. A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum* 39, 5 (2020), 69–80. https://doi.org/10.1111/cgf.14069 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14069 4.3

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). 2.2

Nicholas Sharp, Mark Gillespie, and Keenan Crane. 2021. Geometry processing with intrinsic triangulations. In *ACM SIGGRAPH 2021 Courses* (Virtual Event, USA) *(SIGGRAPH '21)*. Association for Computing Machinery, New York, NY, USA, Article 6, 1 pages. https://doi.org/10.1145/3450508.3464592 6.2

Nicholas Sharp and Alec Jacobson. 2022. Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis. *ACM Trans. Graph.* 41, 4, Article 107 (jul 2022), 16 pages. https://doi.org/10.1145/3528223.3530155 (document), 5.10

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019b. Navigating intrinsic triangulations. *ACM Trans. Graph.* 38, 4, Article 55 (July 2019), 16 pages. https://doi.org/10.1145/3306346.3322979 6.2

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019c. The Vector Heat Method. *ACM Trans. Graph.* 38, 3, Article 24 (June 2019), 19 pages. https://doi.org/10.1145/3243651 3.4, 4.1, 4.3, 4.3, 4.3, 6.2, 7.1

Chen Shen, James F. O'Brien, and Jonathan R. Shewchuk. 2004. Interpolating and approximating implicit surfaces from polygon soup. In *ACM SIGGRAPH 2004 Papers* (Los Angeles, California) *(SIGGRAPH '04)*. Association for Computing Machinery, New York, NY, USA, 896–904. https://doi.org/10.1145/1186562.1015816 3.4

M. Shimrat. 1962. Algorithm 112: Position of point relative to polygon. *Commun. ACM* 5, 8 (Aug. 1962), 434. https://doi.org/10.1145/368637.368653 2.3.3

Patricio D. Simari and Karan Singh. 2005. Extraction and remeshing of ellipsoidal representations from mesh data. In *Proceedings of Graphics Interface 2005* (Victoria, British Columbia) *(GI '05)*. Canadian Human-Computer Communications Society, Waterloo, CAN, 161–168. 5.5.4

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit neural representations with periodic activation functions. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 626, 12 pages. 5.5.3

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Diffusion Art or Digital Forgery? Investigating Data Replication in Diffusion Models . In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 6048–6058. https://doi.org/10.1109/CVPR52729.2023.00586 3.4

Jacob Spainhour, David Gunderman, and Kenneth Weiss. 2024. Robust Containment Queries over Collections of Rational Parametric Curves via Generalized Winding Numbers. *ACM Trans. Graph.* 43, 4, Article 38 (July 2024), 14 pages. https://doi.org/10.1145/3658228 2.3.3

Jacob Spainhour and Kenneth Weiss. 2026. Robust Containment Queries over Collections of Trimmed NURBS Surfaces via Generalized Winding Numbers. *ACM Trans. Graph.* (Feb. 2026). https://doi.org/10.1145/3797957 Just Accepted. 2.3.3

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 4.3

Vijai Kumar Suriyababu, Cornelis Vuik, and Matthias Möller. 2025. Resampling Point Clouds Using Series of Local Triangulations. *Journal of Imaging* 11, 2 (2025). https://doi.org/10.3390/jimaging11020049 8

Gabriel Taubin. 1993. An improved algorithm for algebraic curve and surface fitting. In *1993 (4th) International Conference on Computer Vision*. 658–665. https://doi.org/10.1109/ICCV.1993.378149 5.5.4

Ryan J. Tibshirani, Samy Wu Fung, Howard Heaton, and Stanley Osher. 2024. Laplace Meets Moreau: Smooth Approximation to Infimal Convolutions Using Laplace's Method. arXiv:2406.02003 [math.OC] https://arxiv.org/abs/2406.02003 3.2, 3.4

Mark Tigges, Sheelagh Carpendale, and Brian Wyvill. 1999. Alternate Distance Metrics for Implicit Surface Modeling. (05 1999). 3.4

Yiying Tong, Pierre Alliez, David Cohen-Steiner, and Mathieu Desbrun. 2006. Designing quadrangulations with discrete harmonic forms. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing* (Cagliari, Sardinia, Italy) *(SGP '06)*. Eurographics Association, Goslar, DEU, 201–210. 2.3.3

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. 2.4, 4.5

Greg Turk and James F O'Brien. 1999. Variational implicit surfaces. (1999). 3.4

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. 3.1, 3.2, 3.4

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf 3.4, 5.1.2

Eric Veach and Leonidas J. Guibas. 1995. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 419–428. https://doi.org/10.1145/218380.218498 2.3.3

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable signed distance function rendering. *ACM Trans. Graph.* 41, 4, Article 125 (July 2022), 18 pages. https://doi.org/10.1145/3528223.3530139 2.4, 7.3

Hao Wang, Carlos E. Scheidegger, and Claudio T. Silva. 2008. Optimal bandwidth selection for MLS surfaces. In *2008 IEEE International Conference on Shape Modeling and Applications*. 111–120. https://doi.org/10.1109/SMI.2008.4547957 3.4

Jianlei Wang, Zeyang Wu, Yongqi Tian, and Caigui Jiang. 2025b. Modular Shape Modeling with Controllable Discrete Equivalence Class Distribution. *Computer-Aided Design* 189 (2025), 103935. https://doi.org/10.1016/j.cad.2025.103935 8

Zimo Wang, Ishit Mehta, Haolin Lu, Chung-En Sun, Ge Yan, Tsui-Wei Weng, and Tzu-Mao Li. 2026. Distance Marching for Generative Modeling. arXiv:2602.02928 [cs.LG] https://arxiv.org/abs/2602.02928 8.2

Zimo Wang, Cheng Wang, Taiki Yoshino, Sirui Tao, Ziyang Fu, and Tzu-Mao Li. 2025a. HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1276–1286. 3.4, 5.5.3

Zixiong Wang, Yunxiao Zhang, Rui Xu, Fan Zhang, Peng-Shuai Wang, Shuangmin Chen, Shiqing Xin, Wenping Wang, and Changhe Tu. 2023. Neural-Singular-Hessian: Implicit Neural Representation of Unoriented Point Clouds by Enforcing Singular Hessian. *ACM Trans. Graph.* 42, 6, Article 274 (Dec. 2023), 14 pages. https://doi.org/10.1145/3618311 5.5.3

Jiayi Wei, Jiong Chen, Damien Rohmer, Pooran Memari, and Mathieu Desbrun. 2023. Robust Pointset Denoising of Piecewise-Smooth Surfaces through Line Processes. *Computer Graphics Forum* 42, 2 (2023), 175–189. https://doi.org/10.1111/cgf.14752 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14752 5

Samuel Weidemaier, Florine Hartwig, Josua Sassen, Sergio Conti, Mirela Ben-Chen, and Martin Rumpf. [n. d.]. SDFs from Unoriented Point Clouds using Neural Variational Heat Distances. *Computer Graphics Forum* n/a, n/a ([n. d.]), e70296. https://doi.org/10.1111/cgf.70296 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.70296 5.5.3

Francis Williams, Zan Gojcic, Sameh Khamis, Denis Zorin, Joan Bruna, Sanja Fidler, and Or Litany. 2022. Neural Fields as Learnable Kernels for 3D Reconstruction. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18479–18489. https://doi.org/10.1109/CVPR52688.2022.01795 3.4

Jianhua Wu and Leif Kobbelt. 2005. Structure Recovery via Hybrid Variational Surface Approximation. *Computer Graphics Forum* 24, 3 (2005), 277–284. https://doi.org/10.1111/j.1467-8659.2005.00852.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2005.00852.x 5.5.4

Jianjun Xia and Tao Ju. 2025. Variational Surface Reconstruction Using Natural Neighbors. *ACM Trans. Graph.* 44, 4, Article 85 (July 2025), 19 pages. https://doi.org/10.1145/3731191 5.3, 5.5.4, 5.5.6, 7.1

Hongyi Xu and Jernej Barbič. 2014. Signed Distance Fields for Polygon Soup Meshes. In *Proceedings of Graphics Interface 2014* (Montreal, Quebec, Canada) *(GI '14)*. Canadian Information Processing Society, CAN, 35–41. 2.4

Dong-Ming Yan, Yang Liu, and Wenping Wang. 2006. Quadric Surface Extraction by Variational Shape Approximation. In *Geometric Modeling and Processing - GMP 2006*, Myung-Soo Kim and Kenji Shimada (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 73–86. 5.5.4

Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. 2021. Geometry Processing with Neural Fields. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 22483–22497. https://proceedings.neurips.cc/paper_files/paper/2021/file/bd686fd640be98efaae0091fa301e613-Paper.pdf 8.1

Huizong Yang, Yuxin Sun, Ganesh Sundaramoorthi, and Anthony Yezzi. 2023. StEik: stabilizing the optimization of neural signed distance functions and finer shape representation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 617, 12 pages. 5.5.3

Kaizhi Yang, Liu Dai, Isabella Liu, Xiaoshuai Zhang, Xiaoyan Sun, Xuejin Chen, Zexiang Xu, and Hao Su. 2025. IMLS-Splatting: Efficient Mesh Reconstruction from Multi-view Images via Point Representation. *ACM Trans. Graph.* 44, 4, Article 83 (July 2025), 11 pages. https://doi.org/10.1145/3731210 3.3, 3.4

Minyang Yang and Eungki Lee. 1999. Segmentation of measured point data using a parametric quadric surface approximation. *Computer-Aided Design* 31, 7 (1999), 449–457. https://doi.org/10.1016/S0010-4485(99)00042-1 5.5.4

Lior Yariv, Omri Puny, Oran Gafni, and Yaron Lipman. 2024. Mosaic-SDF for 3D Generative Models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4630–4639. https://doi.org/10.1109/CVPR52733.2024.00443 2.4, 5.5.3

TaeHo Yoon, Joo Young Choi, Sehyun Kwon, and Ernest K. Ryu. 2023. Diffusion Probabilistic Models Generalize when They Fail to Memorize. In *ICML 2023 Workshop on Structured Probabilistic Inference & Generative Modeling*. https://openreview.net/forum?id=shciCbSk9h 3.4

Xue Yu. 2025. *Product Design Sketching in 3D: From Scaffolds to Surfaces*. Ph. D. Dissertation. George Mason University. https://www.proquest.com/dissertations-theses/product-design-sketching-3d-scaffolds-surfaces/docview/3313385565/se-2 8

Lyubomir G. Zagorchev and Arthur Ardeshir Goshtasby. 2012. A Curvature-Adaptive Implicit Surface Reconstruction for Irregularly Spaced Points. *IEEE Transactions on Visualization and Computer Graphics* 18, 9 (2012), 1460–1473. https://doi.org/10.1109/TVCG.2011.276 3.4

Biao Zhang and Peter Wonka. 2025. efunc: An Efficient Function Representation without Neural Networks. arXiv:2505.21319 [cs.GR] https://arxiv.org/abs/2505.21319 5.5.3

Jingyang Zhang, Yao Yao, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, and Long

Quan. 2022. Critical Regularizations for Neural Surface Reconstruction in the Wild. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 6260–6269. https://doi.org/10.1109/CVPR52688.2022.00617 5.5.3

De zhi Ning, Bin Teng, Hai tao Zhao, and Chun ling Hao. 2010. A comparison of two methods for calculating solid angle coefficients in a BIEM numerical wave tank. *Engineering Analysis with Boundary Elements* 34, 1 (2010), 92–96. https://doi.org/10.1016/j.enganabound.2009.06.009 2.3.3

Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh Arrangements for Solid Geometry. *ACM Trans. Graph.* 35, 4, Article 39 (July 2016), 15 pages. https://doi.org/10.1145/2897824.2925901 2.3.3

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016). 5.3

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018). 5.3