Points as Tori: Fast Pointwise Signed Distance for Point Clouds

NICOLE FENG, Carnegie Mellon University, USA
IOANNIS GKIOULEKAS[†], Carnegie Mellon University, USA
KEENAN CRANE[†], Carnegie Mellon University, USA and Roblox, USA

We describe a method for computing signed distance to point clouds that allows fast pointwise evaluation at arbitrary spatial resolution. As input, our method takes a point cloud with normals; as output, it provides an analytical parameterization that allows queries of signed distance to the approximate underlying surface at arbitrary points — simultaneously providing reconstruction and distance. Our key idea is to reconstruct shapes by locally fitting point clouds with tori, which have closed-form signed distance functions. Tori are fitted in a feed-forward manner, using a pre-trained network to output per-point curvature and shift parameters. Importantly, our method does not require costly global optimization or spatial discretization, and is easily parallelizable. Underlying our method is a new theory that unifies signed distance with the classic reconstruction methods of winding numbers and Poisson surface reconstruction. We use our method to compute signed distance to point clouds arising from photogrammetry, meshes, 3D Gaussians, and neural implicits. Our method allows point clouds to be used directly in applications, without explicit surface reconstruction: as examples, we take offsets of point clouds, apply morphological and Boolean operations, and directly visualize offset surfaces using sphere tracing.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; **Point-based models**.

ACM Reference Format:
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 (July 2026), 24 pages. https://doi.org/10.1145/3811385

1 Introduction

For a given shape, its *signed distance function (SDF)* gives the minimum distance between a point $\mathbf{x}$ and the shape boundary Ω, using sign to indicate whether $\mathbf{x}$ is inside or outside Ω. SDFs encode both geometric information (how far $\mathbf{x}$ is from Ω) and topological information (on which side of Ω lies $\mathbf{x}$) into a single scalar function, and have found broad use in geometric modeling, physical simulation, rendering, path planning, geometric learning, and computer vision.

For watertight geometry in $\mathbb{R}^n$, defining signed distance is straightforward. But for point clouds, the inside and outside of the shape — and hence signed distance — are ill-defined. Most existing methods for reconstruction or distance computation generalize signed distance to point clouds at high added cost, typically requiring global solves, and thus precluding applications involving large-scale data, real-time interaction, or compute-constrained systems.

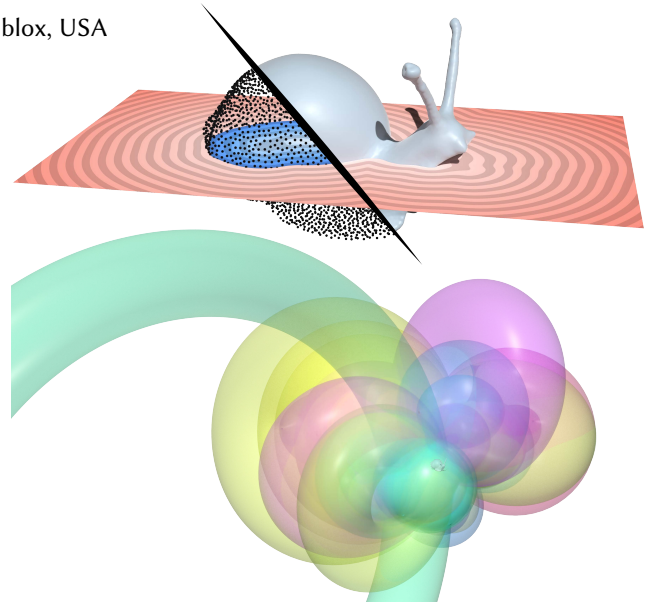


Figure 1. Given a point cloud, our method directly approximates signed distance to the true underlying surface (*top*). The key to our method is an analytical parameterization of the SDF using tori (*bottom*), which allows pointwise queries at arbitrary resolution without spatial discretization or global solves. Here, evaluating the SDF at a single point (to a point cloud with 4096 points) takes only 10^{-4} seconds; multiple evaluations are parallelizable.

In this paper, we address signed distance computation 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. Namely, we give an approximation of signed distance to the true, unknown geometry without needing to explicitly reconstruct it. Significantly, our method is orders of magnitude faster than past signed distance reconstruction methods.

We begin with the observation that there are two possible single-pass formulas for computing distance, called *convolutional distance approximations*: these formulas approximate the minimum distance to a given shape through a summation of kernel functions concentrated on the shape's boundary. Moreover, these formulas in fact unify signed distance with the popular reconstruction methods of *winding numbers* [Jacobson et al. 2013] and *Poisson surface reconstruction* [Kazhdan et al. 2006]. But importantly, we show that only one formula is viable for signed distance reconstruction: in the end, we evaluate signed distance $\phi(\mathbf{x})$ using the *kernel density estimator*

$$\phi(\mathbf{x}) = \frac{\sum_{i=1}^{|P|} g_i(\mathbf{x}) \exp\left(-\lambda_{\mathbf{x}} \|\mathbf{x} - \mathbf{p}_i\|\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda_{\mathbf{x}} \|\mathbf{x} - \mathbf{p}_i\|\right)}, \tag{1}$$

[†]indicates equal contribution.

Authors' Contact Information: Nicole Feng, Carnegie Mellon University, Pittsburgh, PA, USA; Ioannis Gkioulekas, Carnegie Mellon University, Pittsburgh, PA, USA; Keenan Crane, Carnegie Mellon University, Pittsburgh, PA, USA and Roblox, San Mateo, CA, USA.

ACM 1557-7368/2026/7-ART53
https://doi.org/10.1145/3811385

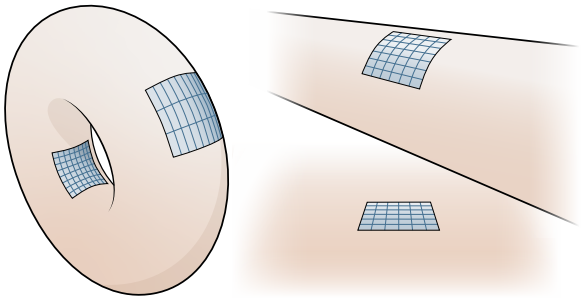


Figure 2. In $\mathbb{R}^3$, SDFs can be approximated by blending the SDFs of oriented tori. Tori have simple closed-form SDF expressions, and can capture locally spherical, ellipsoidal, and saddle-shaped surfaces, as well as cylindrical and planar surfaces as limiting behavior.

Algorithm 1 Points as tori

Input: Point positions and normals $P = \{(\mathbf{p}_i, \mathbf{n}_i)\}_{i=1}^{|P|}$, and a query point $\mathbf{x} \in \mathbb{R}^3$.

Output: The generalized signed distance $\phi(\mathbf{x})$ to P.

Precomputation: Precompute tori $\{\mathbb{T}_i\}_{i=1}^{|P|}$ fitted to each point of P using a pre-trained neural network (Section 4.3).

Inference: Evaluate $\phi(\mathbf{x}) = \frac{\sum_{i=1}^{|P|} g_i(\mathbf{x}) \exp(-\lambda_{\mathbf{x}} \|\mathbf{x}-\mathbf{p}_i\|)}{\sum_{i=1}^{|P|} \exp(-\lambda_{\mathbf{x}} \|\mathbf{x}-\mathbf{p}_i\|)}$ using the signed distance functions $\{g_i\}_{i=1}^{|P|}$ of the fitted tori, and a large $\lambda_{\mathbf{x}}$ (Section 4.2).

giving an exponentially-weighted average of per-point functions g_i associated with a point cloud $P \coloneqq \{\mathbf{p}_i\}_{i=1}^{|P|}$. We define g_i as SDFs to tori, because such SDFs have efficient closed-form expressions, and tori can locally approximate surfaces up to second order (Figures 1, 2). The parameter $\lambda_{\mathbf{x}}$ is automatically chosen with a simple expression. Variants of Equation 1 have been used as heuristics in other point set methods, but were underexplored for signed distance; in Section 2, we show why Equation 1 converges to the true SDF, and why other generalized distance formulas fail.

The main challenge in our 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.

In summary, our method relies on a small pre-trained neural network that takes as input a size-k neighborhood of point $\mathbf{p}_i$, and returns parameters yielding a "best-fit" torus intended to provide a good signed distance reconstruction of the point cloud (Section 4.3).

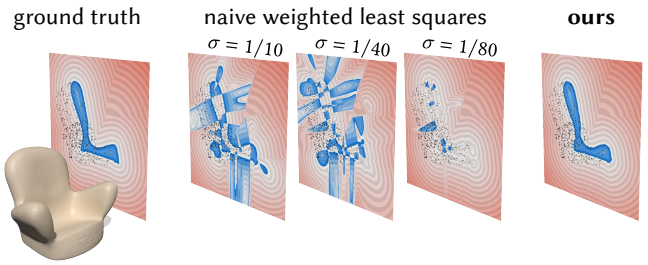


Figure 3. Classic point set methods often use a weighted least squares approach that solves for a best-fit quadratic polynomial around each point $\mathbf{p}_i$, using Gaussian weights $\exp\left(-\|\mathbf{p}_i - \mathbf{p}_j\|^2/\sigma^2\right)$ for each neighbor $\mathbf{p}_j$. In this example, we additionally fit tori to the fitted polynomials to get signed distance (Sections 4.1, 4.2). Without further tuning, this process is incredibly brittle, with no values of σ yielding accurate results. Instead of relying on hand-tuned parameters, we directly learn best-fit surface parameters.

Once this network is trained, one can infer signed distance from *any* point cloud P in two steps (Algorithm 1). Our algorithm, which we call *points as tori (PAT)*,

- infers signed distance to a well-reconstructed surface underlying imperfect observations;
- takes between 10^{-4} and 10^{-3} seconds for a single query to point clouds with millions of points;
- is *output-sensitive*, allowing fast SDF evaluation at specific query points, at arbitrary resolution, without requiring expensive global solves or spatial discretization.

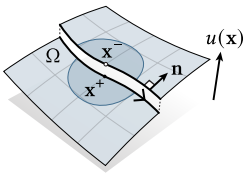
In contrast to other learning-based approaches that try to directly learn SDFs from entire point clouds, we intentionally invoke the minimum amount of learning needed — relying on classical results for convergence to true signed distance, and using learning only for the one component with no easy analytical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

As a signed distance method, our method is not specialized to reconstruction, and its reconstruction quality can likely be outperformed by mature methods specialized to bad geometry. However, our method is competitive with common reconstruction algorithms (Figures 8, 17) and shows tolerance to noise, outliers, imperfect sampling, and inconsistent orientations (Figures 16, 21, 22, 25, 30); at the same time, it extracts richer information that can be used for evenly-spaced offset surfaces, Booleans, morphological operations, and sphere tracing (Figures 22, 23, 24, 12). More broadly, our method is fully differentiable and output-sensitive, providing a promising representation for future work in *e.g.* inverse rendering and generative modeling. Currently, precomputation can take minutes for point clouds with tens of millions of points, but we highlight opportunities for future work and improvement in Section 6.

2 Preliminaries

Convolutional approximations for unsigned distance have been re-discovered several times throughout signal processing, image processing, computer vision, and computer graphics. Here we derive the two main types of such approximations, extend them to signed distance, and show that only one can be used for point clouds.

One-sided limits. Throughout, we discuss signed functions whose value depends on the side from which we approach a given curve or surface Ω. On $\mathbb{R}^d$, we use $\mathbf{x}^{\pm} := \lim_{s\to 0} \mathbf{x} \pm s\mathbf{n}(\mathbf{x})$ to denote a point $\mathbf{x} \in \Omega$ as approached from the positive or negative side of Ω, where $\mathbf{n}(\mathbf{x})$ is the outward-pointing normal of $\Omega \subset \mathbb{R}^d$ at $\mathbf{x}$ that points from the negative to positive side of Ω (see inset). We then denote the corresponding one-sided limits of a function $u(\mathbf{x})$ as $u^{\pm}(\mathbf{x}) := u\left(\mathbf{x}^{\pm}\right) := \lim_{s\to 0} u\left(\mathbf{x} \pm s\mathbf{n}(\mathbf{x})\right)$.

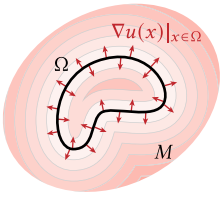


2.1 A unification of signed distance and occupancy

The eikonal equation with two-sided boundary conditions

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

describes an unsigned distance function $u(\mathbf{x})$ to Ω in domain M (inset). Distance functions satisfy Equation 2 only in a "viscosity sense" [Crandall and Lions 1983]: a true distance function is not everywhere differentiable, so one introduces a small amount of scalar diffusion,



$$\begin{aligned} \|\nabla u\|^2 - 1 &= \tfrac{1}{\lambda}\Delta u(\mathbf{x}) & \mathbf{x} &\notin \Omega \\ u(\mathbf{x}) &= 0 & \mathbf{x} &\in \Omega \\ \frac{\partial u^{\pm}}{\partial \mathbf{n}}(\mathbf{x}) &= \pm 1 & \mathbf{x} &\in \Omega. \end{aligned} \tag{3}$$

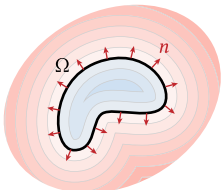
Equation 3 is an example of a time-independent, viscous *Burgers' equation*, for which there is a well-known change of variables called the *Hopf-Cole transformation* (sometimes called *Cole-Hopf transformation*) [Hopf 1950; Cole 1951]; Evans [1998, Section 4.4] gives a derivation. In our context, this transformation becomes

$$w(\mathbf{x}) = \exp(-\lambda u(\mathbf{x})), \tag{4}$$

and turns Equation 3 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]:

$$\begin{aligned} \Delta w(\mathbf{x}) - \lambda^2 w(\mathbf{x}) &= 0 & \mathbf{x} &\notin \Omega \\ w(\mathbf{x}) &= 1 & \mathbf{x} &\in \Omega \\ \frac{\partial w^{+}}{\partial \mathbf{n}}(\mathbf{x}) &= -\frac{\partial w^{-}}{\partial \mathbf{n}}(\mathbf{x}) & \mathbf{x} &\in \Omega. \end{aligned} \tag{5}$$

Hopf-Cole transformations, as applied to the heat equation and the screened Laplace equation in Equation 5, correspond to the two formulas for geodesic distance presented by Varadhan [1967].



We now consider an eikonal equation with boundary conditions continuous across Ω that solves for signed rather than unsigned distance (inset),

$$\begin{aligned} \|\nabla u(\mathbf{x})\|^2 &= 1 & \mathbf{x} &\notin \Omega \\ u(\mathbf{x}) &= 0 & \mathbf{x} &\in \Omega \\ \frac{\partial u}{\partial \mathbf{n}}(\mathbf{x}) &= 1 & \mathbf{x} &\in \Omega. \end{aligned} \tag{6}$$

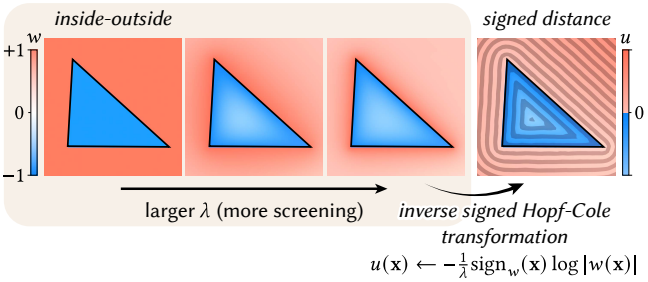


Figure 4. As screening λ goes to 0, the solution w to Equation 9 converges to a jump harmonic function describing inside-outside (*left*). As λ increases, w converges to a signed distance function via a log transformation (*right*). However, this formula works only for closed geometry and fails for sampled geometry like point clouds (Section 2.3). We instead use Equation 17.

We likewise consider a signed eikonal equation with viscosity,

$$\begin{aligned} \operatorname{sign}_{\Omega}(\mathbf{x})\left(\|\nabla u(\mathbf{x})\|^2 - 1\right) &= \tfrac{1}{\lambda}\Delta u(\mathbf{x}) & \mathbf{x} &\notin \Omega \\ u(\mathbf{x}) &= 0 & \mathbf{x} &\in \Omega \\ \frac{\partial u}{\partial \mathbf{n}}(\mathbf{x}) &= 1 & \mathbf{x} &\in \Omega, \end{aligned} \tag{7}$$

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

$$w(\mathbf{x}) = \operatorname{sign}_w(\mathbf{x}) \exp\left(-\lambda \operatorname{sign}_w(\mathbf{x}) u(\mathbf{x})\right). \tag{8}$$

Applying Equation 8 to Equation 7 yields a *jump* screened Laplace equation, whose solution jumps across the source geometry Ω:

$$\begin{aligned} \Delta w(\mathbf{x}) - \lambda^2 w(\mathbf{x}) &= 0 & \mathbf{x} &\notin \Omega \\ w^{\pm}(\mathbf{x}) &= \pm 1 & \mathbf{x} &\in \Omega \\ \frac{\partial w^{+}}{\partial \mathbf{n}}(\mathbf{x}) &= \frac{\partial w^{-}}{\partial \mathbf{n}}(\mathbf{x}) & \mathbf{x} &\in \Omega. \end{aligned} \tag{9}$$

We give a derivation of the signed Hopf-Cole transformation, valid for both closed and open Ω, in Appendix A. Intuitively, applying the inverse signed Hopf-Cole transformation

$$u(\mathbf{x}) = -\tfrac{1}{\lambda} \operatorname{sign}_w(\mathbf{x}) \log |w(\mathbf{x})| \tag{10}$$

reverses the exponential decay of Equation 9, giving an approximation of distance as $\lambda \to \infty$ (Figure 4, *right*). We show in Appendix B that Equation 9 can be rewritten as the screened Poisson equation

$$\Delta w(\mathbf{x}) - \lambda^2 w(\mathbf{x}) = -2\left(\nabla \cdot \mathbf{n}(\mathbf{x})\right) \mu_{\Omega}(\mathbf{x}) \tag{11}$$

without boundary, where $\mathbf{n}(\mathbf{x})$ denotes the outward-pointing unit normals to Ω, and μ_{Ω} is a measure concentrated on Ω.

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

$$G^{\lambda}(\mathbf{x}, \mathbf{y}) := \frac{\exp\left(-\lambda \|\mathbf{x} - \mathbf{y}\|\right)}{4\pi \|\mathbf{x} - \mathbf{y}\|} \tag{12}$$

yields the normal derivative (or *Yukawa double-layer potential*)

$$\nabla G^{\lambda}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{n} := \frac{(\lambda \|\mathbf{x} - \mathbf{y}\| + 1)\left\langle \mathbf{x} - \mathbf{y}, \mathbf{n} \right\rangle \exp\left(-\lambda \|\mathbf{x} - \mathbf{y}\|\right)}{2\pi \|\mathbf{x} - \mathbf{y}\|^3} \tag{13}$$

which can be used to express the solution $w(\mathbf{x})$ of Equation 9 at all $\mathbf{x} \notin \Omega$ as the boundary integral

$$w(\mathbf{x}) = \int_{\Omega} \frac{(\lambda \|\mathbf{x} - \mathbf{z}\| + 1)\left\langle \mathbf{x} - \mathbf{z}, \mathbf{n}(\mathbf{z}) \right\rangle}{2\pi \|\mathbf{x} - \mathbf{z}\|^3} \exp\left(-\lambda \|\mathbf{x} - \mathbf{z}\|\right) \, \mathrm{d}A(\mathbf{z}), \tag{14}$$

where A is the area measure. Unlike the distance function u for which $|u(\mathbf{x})| \to \infty$ as $\|\mathbf{x}\| \to \infty$, the solution w of Equation 9 exhibits exponential decay at infinity, and so can be represented by the boundary integral in Equation 14.

Fascinatingly, these derivations establish a close relationship between classic occupancy methods and signed distance. As the screening parameter $\lambda \to 0$, Equation 9 becomes a jump Laplace equation whose solutions — so-called *jump harmonic functions* [Feng et al. 2023] — describe the generalized winding number (Figure 4, *left*).[1] In turn, the generalized winding number is a special case of *Poisson surface reconstruction* [Kazhdan et al. 2006]: Poisson surface reconstruction is equivalent to a regularized version of winding numbers, which corresponds to convolving the right-hand side of the Poisson equation in Equation 11 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.

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

2.2 Convolutional distance approximations

For perfect geometry, signed distance can be obtained by convolving an exponential kernel over the source geometry Ω (Equation 14), then applying an appropriate log transformation (Equation 10). The kernel used for convolution, however, does not have to be the Yukawa double-layer potential in Equation 13. 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

$$\begin{aligned} &\int_\Omega h(\mathbf{z}) \exp\left(-\lambda\varphi(\mathbf{z})\right)\, \mathrm{d}A(\mathbf{z}) \overset{\lambda\to+\infty}{\sim} \\ &\quad (2\pi/\lambda)^{d/2} \det\left(\nabla^2\varphi(\mathbf{x}^*)\right)^{-1/2} h(\mathbf{x}^*) \exp\left(-\lambda\varphi(\mathbf{x}^*)\right), \end{aligned} \tag{15}$$

where $\mathbf{x}^* := \operatorname{argmin}_{\mathbf{z}\in\Omega} \varphi(\mathbf{z})$ is the minimizer of φ, assumed unique [Belyaev and Fayolle 2024], [Tibshirani et al. 2024, Equation 11]. The observation in Equation 15 is an example application of *Laplace's method*, a classic method in asymptotic analysis [Evans 1998, §4.5], [Bender and Orszag 1999, §6.4].

Intuitively, the integral on the left-hand side of Equation 15 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 15, asymptotically we obtain a direct estimate of the minimum of φ:

$$\begin{aligned} &-\tfrac{1}{\lambda}\log\left(\int_\Omega h(\mathbf{z}) \exp\left(-\lambda\varphi(\mathbf{z})\right)\, \mathrm{d}A(\mathbf{z})\right) \sim \\ &\quad \varphi(\mathbf{x}^*) + \tfrac{d}{2}\tfrac{\log\lambda}{\lambda} - \tfrac{\log h(\mathbf{x}^*)}{\lambda} + O(\lambda^{-1}), \qquad \lambda \to +\infty. \end{aligned}$$

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

[1] The screening in Equations 5, 9, and 11 is volumetric and distinct from the surface-based screening in screened Poisson surface reconstruction [Kazhdan and Hoppe 2013].

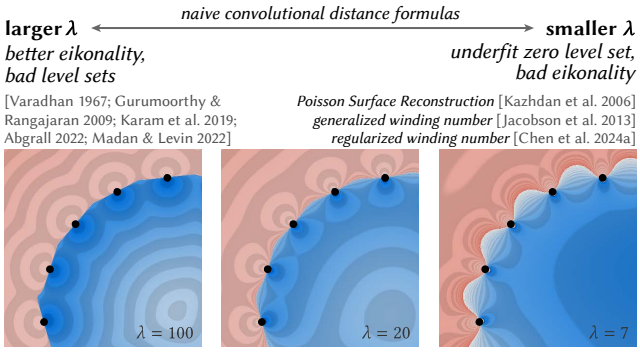


Figure 5. In naive convolutional distance formulas (Equation 16), the parameter λ is coupled to both regression quality and distance accuracy, in opposite directions: using a large λ improves eikonality, but overfits distance to the point set (*left*); using smaller λ at least interpolates the data points, but deviates from a distance function (*center, right*). In these figures, a small amount of regularization [Chen et al. 2024a] was needed for stability.

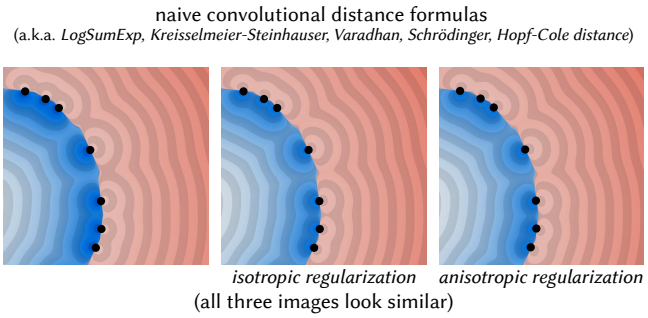


Figure 6. Regularizing the kernel in convolutional distance formulas (Equation 16) is equivalent to choosing the function h, which has no effect asymptotically. Here, we use an exponential kernel with no regularization (*left*), isotropic Gaussian regularization (*center*), and anisotropic Gaussian regularization (*right*), which all produce non-robust distance for $\lambda = 100$.

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

$$\widetilde{d^\lambda}(\mathbf{x}) = -\frac{1}{\lambda}\log\left(\int_\Omega h(\mathbf{x},\mathbf{z}) \exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\, \mathrm{d}A(\mathbf{z})\right) \tag{16}$$

as the general form of a *convolutional distance formula* that approximates the minimum distance from $\mathbf{x}$ to Ω. A special case of Equation 16 is Equation 14: there, $h(\mathbf{x},\mathbf{z}) = \frac{(\lambda\|\mathbf{x}-\mathbf{z}\|+1)\langle\mathbf{x}-\mathbf{z},\mathbf{n}(\mathbf{z})\rangle}{2\pi\|\mathbf{x}-\mathbf{z}\|^3}$. Equation 16 also subsumes unsigned distance approximations such as the *LogSumExp function* [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.

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

$$\widehat{d^\lambda}(\mathbf{x}) = \frac{\int_\Omega g(\mathbf{x},\mathbf{z}) \exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\, \mathrm{d}A(\mathbf{z})}{\int_\Omega \exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\, \mathrm{d}A(\mathbf{z})} \tag{17}$$

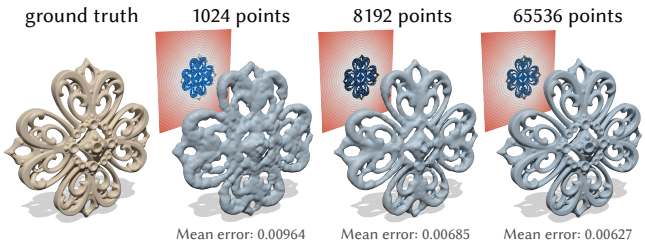


Figure 7. The accuracy of our method improves with sampling quality. Mean error in signed distance is measured over a 128^3 grid of $[-1, 1]^3$.

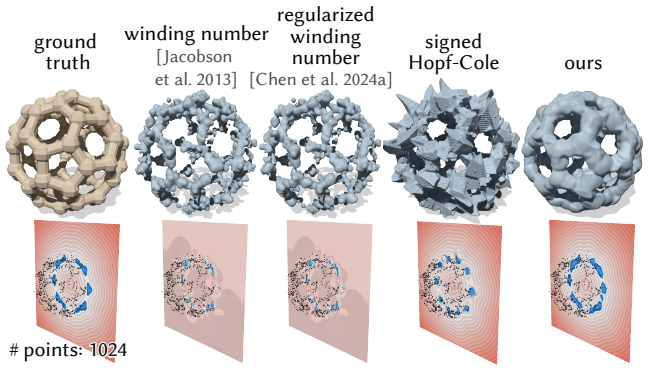


Figure 8. Our method gives better reconstructions than generalized winding number or its regularized variant, especially for sparse data — and provides signed distance to boot, while being almost as fast as either method. Here we use regularization parameter $\varepsilon = 0.02$ for the regularized winding number. Model is object 41140 from Thingi10K [Zhou and Jacobson 2016].

that gives a smooth approximation of $g(\mathbf{x}, \mathbf{x}^*)$ as $\lambda \to \infty$, where $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{z}\in\Omega} \varphi(\mathbf{z})$ is again the minimizer of $\varphi(\mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|$. Equation 17 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 16 with respect to λ.

Equation 17 is an example of a *kernel density estimator*, and offers greater flexibility than Equation 16 because it instead interpolates the function g, which need not be a distance function or even scalar-valued. Equation 17 can also be seen as a partition-of-unity method that computes an expected value of $g(\mathbf{x}, \mathbf{z})$, where the local estimate at $\mathbf{z} = \mathbf{z}'$ has probability $\exp(-\lambda\|\mathbf{x}-\mathbf{z}'\|)/\int_\Omega \exp(-\lambda\|\mathbf{x}-\mathbf{z}\|)\,\mathrm{d}A(\mathbf{z})$.

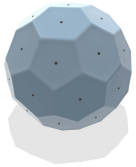
2.3 Fundamental limitations of convolutional distance

As $\lambda \to \infty$, the convolutional distance formulas in both Equations 16 and 17 are completely determined by the behavior of φ or g (resp.) around the global minimizer $\mathbf{x}^*$ 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 $\mathbf{x}^*$ 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 16 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 5). 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 winding numbers, equivalent to adopting a stochastic model of the point cloud geometry. However, the choice of regularizing kernel is equivalent to choice of h, which has no effect asymptotically as $\lambda \to \infty$ (Figure 6).

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 $1/\|\mathbf{x}-\mathbf{z}\|^\lambda$ 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 $\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|^2\right)$, suffer immensely from numerical instability, and anisotropic kernels alter the metric with which distance is measured. Spatially varying λ in Equation 16 compromises eikonality.

Instead, the self-normalized variant of convolutional distance approximations (Equation 17) holds more promise because we can control the manifold's landscape at the closest point via the functions $g(\mathbf{x}, \mathbf{z})$. 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 $g(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} - \mathbf{z}, \mathbf{n}(\mathbf{z})\rangle$ (a.k.a. tangent plane approximation, plane test, or *pseudonormal distance* [Bærentzen and Aanæs 2005]) yielding simple linear continuation of surfaces (inset). We instead fit second-order surfaces: we use tori, which, unlike quadratic polynomial patches or quadric surfaces, admit inexpensive, closed-form signed distance queries. In addition, spatially varying λ does not compromise eikonality as it does in Equation 16, a fact we use in Section 4.2.



3 Related work

We discuss prior work on distance computation, reconstruction, and kernel methods. Our theory in Section 2 allows us to examine the methods in this prior work under a unifying lens, as well as take inspiration from them to develop the first method that simultaneously addresses reconstruction, signed distance, and efficiency.

3.1 Signed distance reconstruction

A few works address signed distance reconstruction, though they struggle with robustness or efficiency. Xu and Barbič [2014] reconstruct point clouds through morphological operations on unsigned distance, which erodes small details. Feng and Crane [2024] use a *generalized signed distance* formulation that, like convolutional distance, relies on the asymptotics of diffusion. Their method uses an intermediate normalization step that decouples regression from distance computation, but necessitates spatial discretization and an expensive global solve. Weidemaier et al. [2025] extend the heat

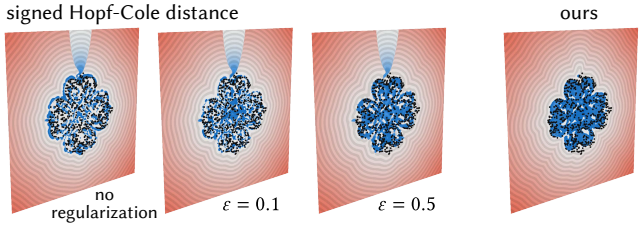


Figure 9. The asymptotic equivalence of signed distance and winding numbers suggests that signed Hopf-Cole distance (Equations 10, 14) can be regularized similarly to the method of Chen et al. [2024a]. However, sign estimation in signed Hopf-Cole distance amounts to the pseudonormal test, which is brittle for point clouds, and singularity regularization does not help.

method for geodesic distance [Crane et al. 2013] using neural networks, which also requires expensive global per-shape optimization.

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 is nonlinear and has many local minima, and because the implicit function values depend nonlinearly on 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 introduce 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. 2023; Yang et al. 2023; Wang et al. 2023; Coiffier and Béthune 2024]. Some methods use the signed Hopf-Cole transformation of Section 2 [Lipman 2021; Wang et al. 2025], but cannot provide signed distance to point clouds. Our 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 17).

Furthermore, these neural field representations must train a new neural network for every shape. Our network shares weights across point neighborhoods and shapes — so it can be trained once, and thereafter applied to other input data without further training.

3.2 Kernel methods

Equation 16 and Equation 17 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 solving PDEs (*e.g.* smoothed particle hydrodynamics). Sharp et al. [2019] also use a version of Equation 17 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 lie on a low-dimensional

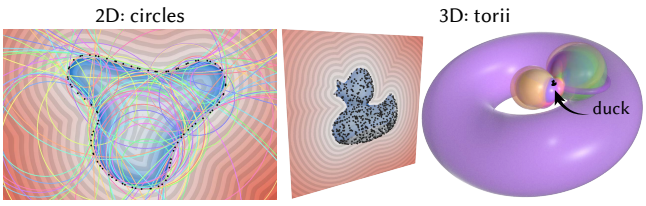


Figure 10. In 2D, our method would use circles to approximate a 1D curve (*left*); in 3D, we use tori to approximate 2D surfaces, since tori generalize circles to two directions of curvature (*right*). Note that our SDF approximation is *not* equivalent to *e.g.* taking the union or intersection of spheres or tori, but instead uses a distance-weighted average.

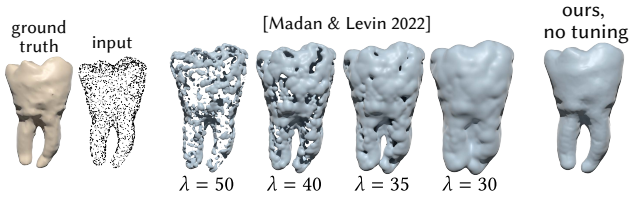


Figure 11. Madan and Levin [2022] estimate unsigned distance to a point cloud P using $d(\mathbf{x}) = -1/\lambda \log\left(\sum_{i=1}^{|P|} w_i(\mathbf{x}) \exp\left(-\lambda \|\mathbf{x} - \mathbf{p}_i\|\right)\right)$ with $w_i(\mathbf{x}) = 1$, an instance of Equation 16. However, it is difficult to choose a single λ that yields a good reconstruction, and easy to accidentally merge distinct features. We experimented with using area weights, but could not tune this method effectively. Spatially varying λ compromises eikonality in LogSumExp-type methods.

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. Though kernel methods can be efficient, they have 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; Guennebaud and Gross 2007; 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.

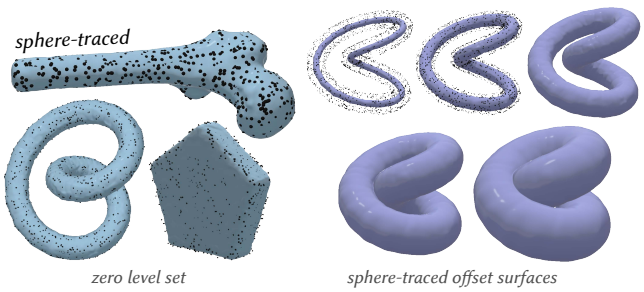


Figure 12. Our method allows directly visualizing surfaces underlying point clouds through sphere tracing. Here we sphere-trace size-2048 point clouds in a shader, on a MacBook Pro laptop without a dedicated GPU.

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.

Convolutional distance methods. Instances of the two convolutional formulas in Equation 16 and Equation 17 appear widely across mathematics, computer science, and engineering. For instance, Varadhan [1967] uses similar asymptotic analysis as in Section 2.2 to derive two formulas for geodesic distance with the same exponential change-of-variables as Hopf-Cole. Many authors have used Equation 16 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]. Like our work, Madan and Levin [2022] use a kernel method to approximate a distance field to meshes; unfortunately, using Equation 16 it is difficult to balance reconstruction and distance for point clouds (Figure 11). More recently, the relationship between eikonal and screened Laplace equations has been used in neural reconstruction methods [Lipman 2021; Wang et al. 2025]. Other types of asymptotic distance approximations include *p-Poisson* or *L_p distances* [Belyaev and Fayolle 2015]. However, none of these methods can directly provide signed distance to point clouds, as discussed in Section 2.3.

More generally, the exponential asymptotics that underlie convolutional distance formulas also underlie common logistic regression techniques used for clustering and classification. Equation 16, 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 17. 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.

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 2.3). Huang et al. [2023]; Williams et al. [2022] use learned neural kernels for surface reconstruction, but do not provide signed distance.

Generative modeling. Modern generative models such as *diffusion models* and *flow matching* amount to simple kernel regression formulas based on Equation 17 that push samples towards the closest datapoint seen in training [Scarvelis et al. 2023; Gao and Li 2024; Kamb and Ganguli 2025]. They can hence 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 2.3). This observation about generative models has been made by many authors [Liu et al. 2022; Somepalli et al. 2022; Pidstrigach 2022; Yoon et al. 2023; Carlini et al. 2023; Jain et al. 2024; 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.

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.

3.3 Fitting geometric proxies to point clouds

Numerous 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.

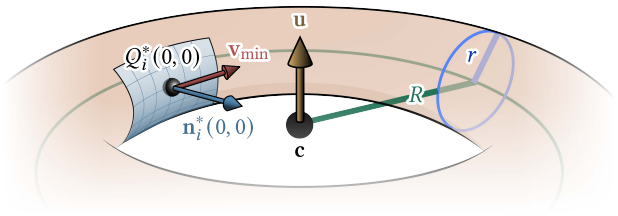


Figure 13. We fit the ith torus so that its equator passes through the point $Q_i^*(0,0)$ given by the polynomial height function at point $\mathbf{p}_i$. The torus's major and minor radii are aligned with the principal curvatures and directions of the polynomial surface (Section 4.1).

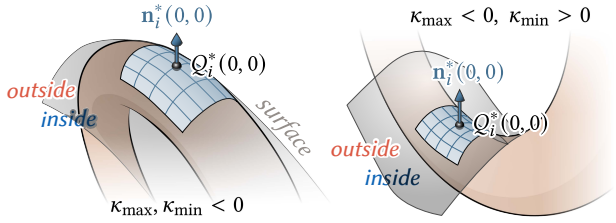


Figure 14. The sign of each torus is determined by whether the torus osculates the polynomial from the shape's interior or exterior. Here we show the two sign combinations of $\kappa_{max}, \kappa_{min}$ corresponding to a positive SDF.

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.

3.4 Signed distance interpolation

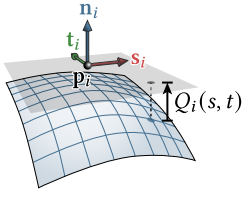
The idea of constructing global SDFs by interpolating local ones is classic in graphics: early examples include accelerating SDF evaluation via texture lookup, or adaptive grid-based sampling [Frisken et al. 2000]. More recently, neural methods have adopted similar interpolation-based, local-to-global approaches for constructing implicit shape representations [Yariv et al. 2023; Zhang and Wonka 2025; Lin et al. 2025]. However, these methods fit networks to *existing* shapes, and do not infer signed distance of imperfect geometry; they require fitting a new network for each new input shape. Other methods interpolate occupancy information [Boulch and Marlet 2022], and do not provide distance information.

4 Algorithm

We are given a point cloud P with normals, and use Equation 1 to estimate distance ϕ. In turn, the zero level set of ϕ produces a surface reconstruction of Ω. In this section, we describe the per-point functions g we use. These correspond to fitting a torus to the neighborhood of each point: for each point $\mathbf{p}_i$ in a given point cloud, we use the coefficients predicted by the neural network described in Section 4.3 to determine the SDF g_i of a single torus fitted at $\mathbf{p}_i$. This neural network directly predicts the entries of the first and second fundamental forms of the surface around each point $\mathbf{p}_i$, which yield principal curvatures and directions, as well as a shift coefficient (Section 4.1). At inference time, we need only evaluate this network per point (Section 4.2) to fit the tori used for the final SDF. Fitting tori is a precomputation step that needs to be done only once per point cloud, and can be parallelized over all points.

4.1 Torus fitting

Let each point $\mathbf{p}_i \in P$ be associated with a supporting plane defined by the normal $\mathbf{n}_i$ at $\mathbf{p}_i$, passing through $\mathbf{p}_i$ (inset). Let $\{a_{n,m}\}_{n,m=0}^{2}$ be the coefficients of a polynomial describing the surface around $\mathbf{p}_i$,



expressed in its local coordinate frame:

$$Q_i^*(s,t) = \mathbf{p}_i + s \cdot \mathbf{s}_i + t \cdot \mathbf{t}_i + Q_i(s,t) \cdot \mathbf{n}_i,$$
$$\text{where } Q_i(s,t) = \sum_{n=0}^{2}\sum_{m=0}^{2} a_{n,m} s^n t^m. \tag{18}$$

A standard torus $\mathbb{T}$ is defined by a major radius $R \in \mathbb{R}$, minor radius $r \in \mathbb{R}$, center $\mathbf{c} \in \mathbb{R}^3$, and axis of revolution $\mathbf{u} \in \mathbb{R}^3$. We fit a torus $\mathbb{T}_i$ to each point $\mathbf{p}_i$ such that its equator passes through the *shifted* point $Q_i^*(0,0) := \mathbf{p}_i + a_{0,0}\mathbf{n}_i$, and its principal curvatures and directions align to those of the polynomial surface $Q_i^*(s,t)$ (Figure 13). In the local coordinate frame, the first and second fundamental forms of $Q_i^*(s,t)$ equal (see Appendix C for derivations):

$$\mathrm{II} = \frac{1}{A}\begin{bmatrix} 2a_{2,0} & a_{1,1} \\ a_{1,1} & 2a_{0,2} \end{bmatrix}, \quad \mathrm{I} = \begin{bmatrix} 1+a_{1,0}^2 & a_{1,0}a_{0,1} \\ a_{1,0}a_{0,1} & 1+a_{0,1}^2 \end{bmatrix},$$

where $A := \sqrt{1 + a_{0,1}^2 + a_{1,0}^2}$. Thus, 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 18 for each point $\mathbf{p}_i$; we describe how we predict them in Section 4.3.

The two principal curvatures equal $\kappa_\pm = H \pm \sqrt{H^2 - K}$, where

$$H = \frac{a_{0,2}(1+a_{1,0}^2) + a_{2,0}(1+a_{0,1}^2) - a_{1,1}a_{1,0}a_{0,1}}{A^3},$$
$$K = \frac{4a_{0,2}a_{2,0} - a_{1,1}^2}{A^4}. \tag{19}$$

The principal directions are parallel to

$$\mathbf{v}_\pm = [\mathbf{w}_\pm]_x \cdot (\mathbf{s}_i + a_{1,0}\mathbf{n}_i) + [\mathbf{w}_\pm]_y \cdot (\mathbf{t}_i + a_{0,1}\mathbf{n}_i),$$
$$\text{where } \mathbf{w}_\pm = \begin{bmatrix} \kappa_\pm a_{1,0}a_{0,1}A - a_{1,1} & 2a_{2,0} - \kappa_\pm\left(1 + a_{1,0}^2\right)A \end{bmatrix}. \tag{20}$$

The unit normal $\mathbf{n}_i^*(0,0)$ of the polynomial at $(s,t) = (0,0)$ equals

$$\mathbf{n}_i^*(0,0) = \frac{\mathbf{n}_i - a_{1,0}\mathbf{s}_i - a_{0,1}\mathbf{t}_i}{\sqrt{1 + a_{1,0}^2 + a_{0,1}^2}}. \tag{21}$$

We take the major radius R to always be greater than the minor radius r, and define

$$\kappa_{min} := \mathrm{argmin}_{\kappa \in \{\kappa_+,\, \kappa_-\}} |\kappa|, \qquad \kappa_{max} := \mathrm{argmax}_{\kappa \in \{\kappa_+,\, \kappa_-\}} |\kappa|,$$
$$\widehat{\mathbf{v}}_{min} := \begin{cases} \frac{\mathbf{v}_+}{\|\mathbf{v}_+\|} & \text{if } |\kappa_+| < |\kappa_-|, \\ \frac{\mathbf{v}_-}{\|\mathbf{v}_-\|} & \text{otherwise,} \end{cases}$$

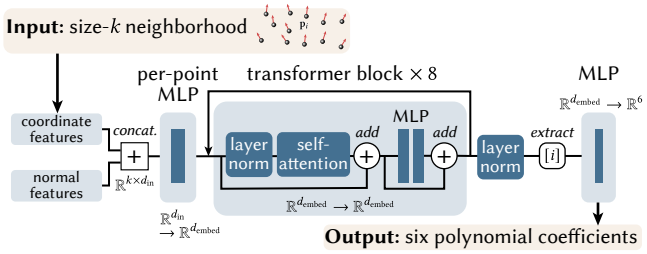


Figure 15. A neural network is applied to each size-k neighborhood in the point cloud. At its core is a series of transformer blocks that learn which points in each neighborhood are important for locally fitting a torus.

and analogously for $\widehat{\mathbf{v}}_{\max}$. Then we have

$$r = |\kappa_{\max}|^{-1}, \quad R = \frac{1}{|\kappa_{\min}|} - \operatorname{sign}(\kappa_+\kappa_-)\, r,$$

where the latter comes from $(R + \operatorname{sign}(\kappa_+\kappa_-)\, r)^{-1} = |\kappa_{\min}|$.

Next, we determine the sign of the torus's SDF as (Figure 14):

$$\operatorname{sign}(\mathbb{T}_i) = \begin{cases} +1 & \text{if } \kappa_{\max}, \kappa_{\min} < 0, \text{ or } \kappa_{\max} < 0 \text{ and } \kappa_{\min} > 0, \\ -1 & \text{otherwise.} \end{cases} \tag{22}$$

The center $\mathbf{c}$ of the torus is then

$$\mathbf{c} = Q_i^*(0,0) - \frac{\operatorname{sign}(\mathbb{T}_i)}{|\kappa_{\min}|}\mathbf{n}_i^*(0,0),$$

and the axis of revolution $\mathbf{u}$ is

$$\mathbf{u} = \mathbf{n}_i^*(0,0) \times \widehat{\mathbf{v}}_{\min}.$$

With these parameters, we can evaluate the SDF of a torus as

$$\phi_{\mathbb{T}}(\mathbf{x}) = \|\mathbf{d}\| - r, \quad \mathbf{d} \coloneqq \left(\|(\mathbf{x}-\mathbf{c}) \times \mathbf{u}\| - R, \langle \mathbf{x}-\mathbf{c}, \mathbf{u}\rangle\right), \tag{23}$$

a formula obtained by applying the SDF to a circle, twice [Quilez 2025]. The final SDF of torus $\mathbb{T}_i$, to be blended using Equation 1, is

$$g_i(\mathbf{x}) = \operatorname{sign}(\mathbb{T}_i)\, \phi_{\mathbb{T}_i}(\mathbf{x}). \tag{24}$$

4.2 Evaluating signed distance

Once tori have been fitted, the global SDF $\phi(\mathbf{x})$ can be computed with Equation 1 using the fitted g_i, at arbitrary query points $\mathbf{x}$.

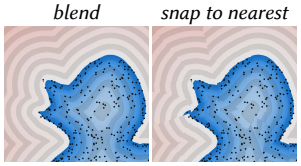
Setting λ. With a good local reconstruction in place via the fitted tori, we simply need to set a large λ in Equation 1 to get good distance. How large we can make λ depends on machine precision, and a simple shifting of the exponents buys precision without changing the result [Blanchard et al. 2020]. Empirically, we find good behavior using the following shifted formula instead of Equation 1:

$$\phi(\mathbf{x}) = \frac{\sum_{i=1}^{|P|} g_i(\mathbf{x}) \exp\left(-\lambda_{\mathbf{x}}\left(\|\mathbf{x}-\mathbf{p}_i\| - \sigma_{\mathbf{x}}\right)\right)}{\sum_{i=1}^{|P|} \exp\left(-\lambda_{\mathbf{x}}\left(\|\mathbf{x}-\mathbf{p}_i\| - \sigma_{\mathbf{x}}\right)\right)}. \tag{25}$$

We set the exponential shift and screening for each $\mathbf{x}$ as

$$\begin{aligned} \sigma_{\mathbf{x}} &\coloneqq \frac{1}{2}\max\left(\{\|\mathbf{x}-\mathbf{p}_i\| \mid \mathbf{p}_i \in P\}\right), \\ \lambda_{\mathbf{x}} &\coloneqq \frac{C}{\max\left(\{\|\mathbf{x}-\mathbf{p}_i\| - \sigma_{\mathbf{x}} \mid \mathbf{p}_i \in P\}\right)} = \frac{C}{\sigma_{\mathbf{x}}}, \end{aligned} \tag{26}$$

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 $C = 64$.



Blending torus SDFs via Equation 25 is important, as simply using the torus of the nearest point results in discontinuities (inset). Equation 25 technically requires area weights, as a discretization of the surface integral in Equation 17, but we omit them for efficiency. We hypothesize that we do not observe many ill effects because we fit surfaces with area, rather than singular kernels.

Acceleration and adaptive λ. Equation 25 naively requires summing over all points in the input point cloud. For large point clouds, this is both slow and potentially inaccurate since summing over a large radius limits how large we can take λ. We instead sum over only points within a fixed radius R_{eval} of each evaluation point $\mathbf{x}$. If no points lie within R_{eval}, we fall back to summing over the 32 nearest points. We select R_{eval} by keeping λ constant for a given point cloud and adjust R_{eval} according to machine precision (Equation 26). We use the heuristic $\lambda = 10^3/D$ where D is the mean distance between a point and its 64 nearest neighbors, averaged over all points in P; then $R_{\text{eval}} = 2C/\lambda$. 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 do not explore these avenues.

4.3 Learning per-neighborhood coefficients

We train a neural network that learns to predict the six polynomial coefficients used to determine the constant shift and the first and second fundamental forms of the surface at each point $\mathbf{p}_i$. As input, the network takes in a point set $\mathcal{N}_k(\mathbf{p}_i)$ comprising $\mathbf{p}_i$ and its k nearest neighbors. As output, the network returns the coefficients $a_{0,0}, a_{0,1}, a_{1,0}, a_{1,1}, a_{0,2}, a_{2,0}$. The network consists primarily of transformer blocks using attention [Vaswani et al. 2017], another form of kernel regression, to learn which points in the neighborhood are most important for fitting a local surface.

Architecture. At a high level, the input points and normals are embedded into a higher dimension, where a series of transformer blocks are applied, before being projected down to scalar values via a learned projection (Figure 15). First, we uniformly scale the input neighborhood by dividing by the median distance σ_i between $\mathbf{p}_i$ and its neighbors. We construct an orthonormal basis using the normal $\mathbf{n}_i$ of the central point $\mathbf{p}_i$, and the position and normal of each point are encoded relative to this local basis to achieve invariance to global rigid transformations of the point cloud. We use a fixed orthonormal basis for each point, so the network can learn more effectively from a fixed set of training data. These features are concatenated into a $\mathbb{R}^6$ feature vector per point, and each feature vector is input to a single-layer multi-layer perceptron (MLP) with $d_{\text{embed}} = 128$ neurons; weights are shared across points. We then use an eight-layer transformer with hidden dimension d_{embed}, eight attention heads, and MLP dimension 512. Finally, a single-layer MLP with six neurons is applied to the feature vector corresponding to point

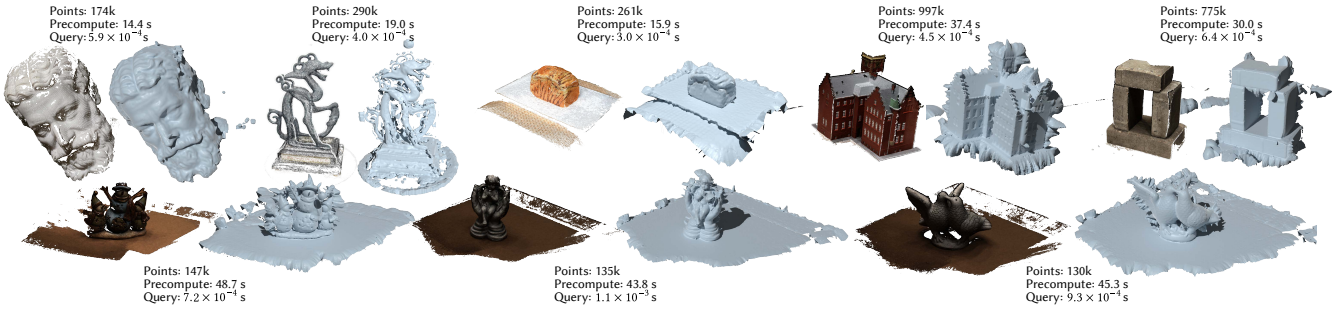


Figure 16. We visualize some reconstructions produced by our method on dense point cloud outputs from COLMAP. Our method yields reasonable reconstructions despite not being trained on any data with noise or outliers.

$\mathbf{p}_i$, to obtain six scalar values $a'_{0,0}, a'_{0,1}, a'_{1,0}, a'_{1,1}, a'_{0,2}, a'_{2,0}$. We scale these coefficients back to their original coordinate system,

$$\begin{aligned} &a_{0,0} \leftarrow \sigma_i \cdot a'_{0,0} \\ &a_{0,1} \leftarrow a'_{0,1}, \qquad a_{1,0} \leftarrow a'_{1,0} \quad \text{(no change)} \\ &a_{1,1} \leftarrow \sigma_i^{-1} \cdot a'_{1,1}, \quad a_{0,2} \leftarrow \sigma_i^{-1} \cdot a'_{0,2}, \qquad a_{2,0} \leftarrow \sigma_i^{-1} \cdot a'_{2,0}. \end{aligned}$$

Loss function. We train our network using a set of point clouds and ground-truth distance data that we create as we describe below. For each point cloud in the training set, the loss function comprises the L^1 norm of distance error — penalizing differences between the SDF ϕ predicted using Equation 1 and ground-truth — and eikonal error — penalizing deviations of ϕ from Equation 6. To evaluate the loss, for each neighborhood in the point cloud, we sample $Q = 120$ query points: 1/3 uniformly from the intersection of the ground-truth surface with the neighborhood's axis-aligned bounding box; 1/3 from a narrow band of width 0.2; and 1/3 from a cube with sidelength equal to the largest extent of the neighborhood's bounding box. We compute a per-neighborhood loss as

$$\begin{aligned} \mathcal{L} &\coloneqq \mathcal{L}_{\text{distance}} + \mathcal{L}_{\text{eikonal}}, \\ \text{where } \mathcal{L}_{\text{distance}} &= \frac{1}{Q} \sum_{j=1}^{Q} \left| \phi\left(\mathbf{q}_j\right) - \phi_{\text{true}}\left(\mathbf{q}_j\right) \right|, \\ \mathcal{L}_{\text{eikonal}} &= \frac{1}{Q} \sum_{j=1}^{Q} \left| 1 - \left\| \nabla \phi\left(\mathbf{q}_j\right) \right\| \right|, \end{aligned} \tag{27}$$

then aggregate this loss across all neighborhoods. The eikonal loss implicitly encourages neighborhoods to blend together smoothly.

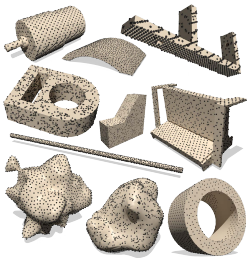
Training data. We create training data by using triangle mesh models to both sample point clouds and compute ground-truth distance values. We use models from the ABC dataset [Koch et al. 2019], which contains one million CAD models, partitioned into 100 chunks; we randomly select chunks 2, 17, 55, and 58.

To complement the geometries of CAD models, we also use procedurally-generated "blob" meshes using a procedure inspired by user484 [2015]. Each blob is defined as the implicit surface

$$F(x, y, z) = x^2 + y^2 + z^2 - (1 + \alpha \cdot \text{PerlinNoise}(x, y, z))^2 .$$

Meshes are created by sampling Perlin noise values on a regular unit grid of resolution 64^3, and contouring the zero level set of F using marching cubes [Lorensen and Cline 1998]. We use the `noise` library [Duncan 2018] for 3D Perlin noise using persistence 0.5 and lacunarity 2; for each blob, we randomly sample a scale value between 0.5 and 2.0, an octave between 1 and 4, and amplitude $\alpha \in [0.1, 0.7]$. We keep only watertight blobs.

To accelerate training, we use only meshes with 2048 or fewer faces, and sample point clouds with 2048 points. For each mesh, we randomly choose with equal probability whether to use uniform or farthest point sampling; the latter entails first uniformly sampling $2048 \cdot 8$ points, then subsampling. We also sample point clouds with gaps and uneven sampling patterns, using a simulated scanning procedure: We first scale and center the mesh so it lies within the $[-1, 1]^3$ cube and its centroid is at the origin. We place four cameras on a bounding sphere of radius $\sqrt{3}$ using Fibonacci sampling, with a random global rotation applied to all cameras. Each camera emits rays arranged in a 2D square grid of sidelength $\sqrt{3}$ with grid spacing randomly chosen between 0.01 and 0.1, and we record the positions and normals at the first intersection of the rays with the mesh using the `FCPW` library [Sawhney et al. 2020].



We use 10000 models from ABC and 2000 procedurally-generated shapes to create two datasets, each containing about 20 million training examples: one with point clouds sampled using either uniform or farthest point sampling, and another with point clouds sampled using the simulated scanning procedure. The inset shows examples.

4.4 Discussion

We tried many alternative approaches that did not 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.

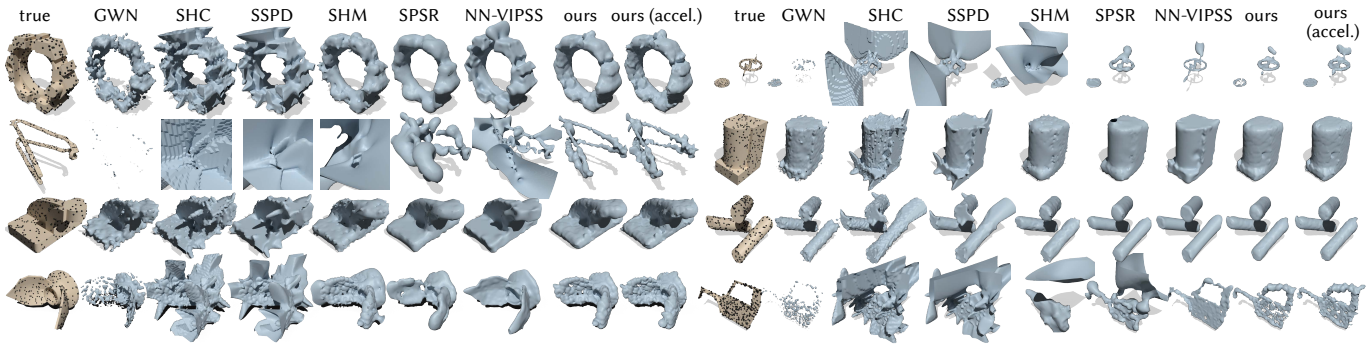


Figure 17. We visualize reconstructions of several shapes used in our evaluation (Section 5.2). Our method tends to give better results on sparse point clouds, probably because our network was trained on relatively sparse point clouds (2048 points); here we use point clouds with 512 points. On some shapes, we observed better results using smaller λ than provided by the simple heuristic presented in Section 4.2, 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 19).

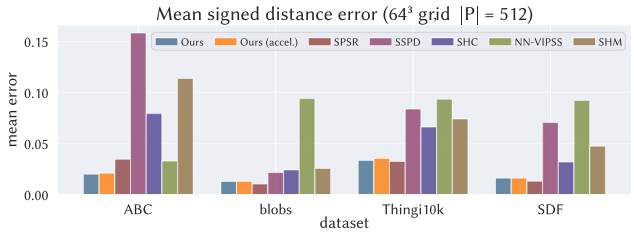


Figure 18. On average, our method has better signed distance accuracy than other convolutional formulas, and even a global method (SHM) for sparse point cloud data. It is also less prone to catastrophic error (Figure 17).

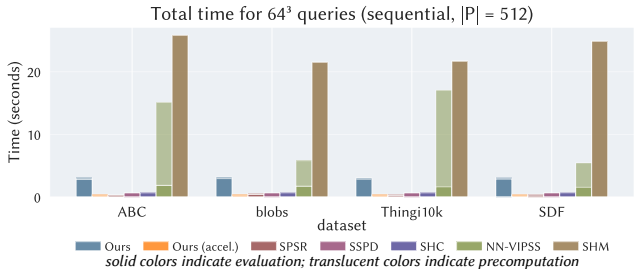


Figure 19. Our method is almost as fast as naive convolutional methods, and orders of magnitude faster than the signed heat method (SHM) while providing good signed distance quality on our datasets to sparse point clouds. For SPSR and NN-VIPSS, "precomputation" refers to reconstruction and meshing, while "evaluation" refers to distance evaluation to the meshed surface. Times correspond to sequential queries.

We also tried approaches that directly optimized for torus parameters, but found they were ill-conditioned. Then, similar to our proposed method, we tried a neural network that for each point learned per-neighbor weights with which to solve a weighted least-squares minimization for the best-fit quadratic surface in the neighborhood. However, under this model it was difficult to provide a good initialization of the 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 least squares problem in the forward pass (for example, as done by Ben-Shabat and Gould [2020]). We additionally experimented with learning 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 from point clouds with any noise, which we hypothesize is because curvature estimation is a relatively ill-conditioned problem.

5 Evaluation

We evaluate our method's accuracy and performance in Section 5.2, and show applications in Section 5.3.

5.1 Implementation

We implemented the neural network in JAX [Bradbury et al. 2018], and the inference-time components of our method (torus-fitting and distance evaluation) in C++, using Python bindings implemented with `nanobind` [Jakob 2022]. We used `nanoflann` [Blanco and Rai 2014] to implement the kd-tree used for k-nearest neighbor and radius queries during inference.

We use the Adam optimizer [Kingma and Ba 2015] for training, and follow a curriculum learning approach, cumulatively increasing the amount of training data according to difficulty. We first train using only the directly-sampled point clouds: we use only the 2000 procedurally-generated shapes for 24 epochs, then incorporate 2000 shapes from the ABC dataset for 8 epochs, then incorporate a further 3500 shapes from ABC for another 8 epochs, then use all directly-sampled shapes for 5 epochs. Finally, we train on all shapes, using all point clouds, for 2 epochs.

For each training phase, we use a learning rate of 1×10^{-6}, except for the first which uses 1×10^{-5}, and the last which uses 1×10^{-7}; and use a linear warmup schedule for the first 1000 training steps, followed by cosine decay. We use a batch size of 1024 point-cloud neighborhoods throughout. We initialize the weights of the network such that the initial output is $a_{0,0} = a_{0,1} = a_{1,0} = a_{1,1} = 0$, $a_{0,2} = a_{2,0} = -0.5$, which corresponds to fitting a sphere tangent to each point. Training took about 45 hours on a single NVIDIA RTX 3090.

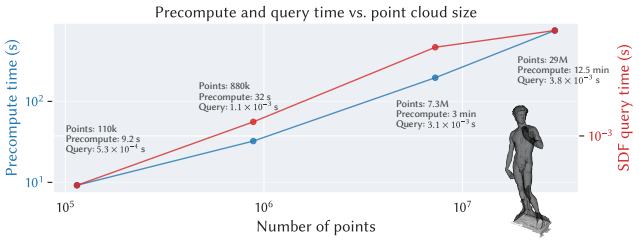


Figure 20. We show precompute and query times for point clouds of size 110k, 880k, 7.3M, and 29M. Even for point clouds reaching tens of millions, each query of our SDF remains on the order of milliseconds.

5.2 Accuracy and performance

We evaluate accuracy and performance on four datasets: 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 $[-1,1]^3$. From Thingi10K, we use only the 45 surface meshes that are closed, manifold, oriented, and non-self-intersecting, so they have well-defined SDFs. To create analytically-defined SDFs, we randomly extrude or revolve fourteen 2D SDFs from Quilez [2019]. We sampled different instances of each shape by randomly selecting the parameters of the 2D SDF with valid ranges (see Appendix E for details), and either revolving or extruding the SDF according to Quilez [[n. d.]] using randomly selected offset and extrusion parameters between 0.1 and 0.3. Ground-truth distances to triangle meshes were computed using `FCPW` [Sawhney et al. 2020], and signed using generalized winding number [Jacobson et al. 2013]. We sample point clouds of 512 points using uniform sampling. We conduct our evaluations on a MacBook Pro laptop (M3 Max chip, 16-core CPU, 64 GB of RAM).

Signed distance accuracy. We compare with two baselines: signed Hopf-Cole (a.k.a. signed LogSumExp, or screened winding number)

$$\phi_{\text{SHC}}(\mathbf{x}) = \text{sign}_w(\mathbf{x}) \left(-\frac{1}{\lambda} \log |w(\mathbf{x})| + \sigma_{\mathbf{x}} \right),$$

$$w(\mathbf{x}) \coloneqq \sum_{i=1}^{|P|} A_i \frac{(\lambda \|\mathbf{x}-\mathbf{p}_i\| + 1)\, \langle \mathbf{x}-\mathbf{p}_i, \mathbf{n}_i \rangle}{2\pi \|\mathbf{x}-\mathbf{p}_i\|^3} \exp\left(-\lambda \left(\|\mathbf{x}-\mathbf{p}_i\| - \sigma_{\mathbf{x}}\right)\right) \tag{28}$$

as an instance of Equation 16; and the smoothed signed planar distance

$$\phi_{\text{SSPD}}(\mathbf{x}) = \frac{\sum_{i=1}^{|P|} A_i \langle \mathbf{x}-\mathbf{p}_i, \mathbf{n}_i \rangle \exp\left(-\lambda \left(\|\mathbf{x}-\mathbf{p}_i\| - \sigma_{\mathbf{x}}\right)\right)}{\sum_{i=1}^{|P|} A_i \exp\left(-\lambda \left(\|\mathbf{x}-\mathbf{p}_i\| - \sigma_{\mathbf{x}}\right)\right)} \tag{29}$$

as a baseline instance of Equation 17. The point area weights A_i are computed using the geodesic Voronoi area approximation described by Barill et al. [2018], using the implementation in `libigl` [Jacobson et al. 2018]. We also compare with the signed heat method (SHM) of Feng and Crane [2024], which can provide signed distance to point clouds though is a grid-based method.

Finally, we 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 et al. 2020] 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]. On occasion, the latter method outputs a mesh with flipped normals; we verify the orientation of the contoured surface mesh by flipping normals if the winding number averaged over a regular grid of points is less than 0, and manual inspection. We limit the maximum depth of SPSR's octree to correspond to the resolution of the grid used for contouring other methods (see below). We use single-threaded SPSR because the multi-threaded version kept crashing, and similarly preserve the bounding box calculations of NN-VIPSS (rather than use the standard $[-1,1]^3$ domain) because we encountered errors otherwise.

We evaluate all methods on a regular grid of resolution 64^3, and compute their mean absolute error over all grid points $\mathbf{q}_i$,

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

where: ϕ is the signed distance estimate, ϕ_{true} is the ground-truth signed distance, and $G = 64^3$. Compared to the other point-based approaches and SHM, our method has consistently lower error across the four datasets (Figure 18). 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 their tet mesh discretization with exact zero-set constraints at input points.

In Figure 17, 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 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 4.2. 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.

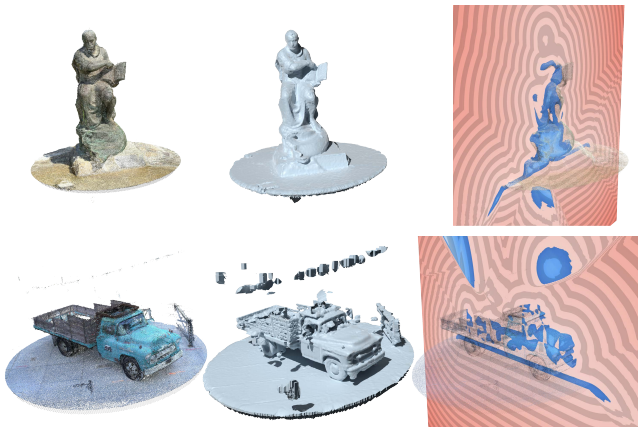


Figure 21. We show reconstructions and slices of the SDF resulting from the noisy point cloud output of GLOMAP [Pan et al. 2024], applied to the Tanks and Temples dataset [Knapitsch et al. 2017].

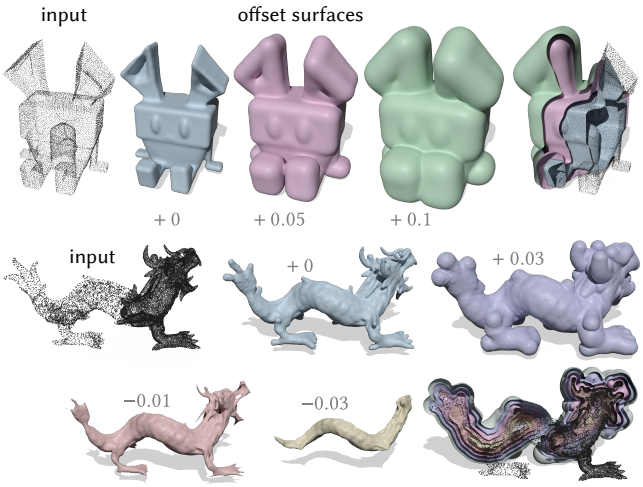


Figure 22. We take offset surfaces to a point cloud sampled from a non-manifold, self-intersecting mesh (object 40923 from Thingi10K) (*top*) as well as to an unevenly sampled point cloud (*bottom*).

Performance. During signed distance evaluation, we measure average sequential query times of our SDF, and the precompute time for each point cloud. The latter includes kd-tree construction, neural network evaluation, and derivation of torus parameters from the inferred coefficients. The vast majority of precompute is spent on neural network evaluations for surface coefficient inference. Figure 19 shows timings, which were measured on our MacBook Pro laptop. For comparison, we also 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 our method averages 7.4×10^{-5} seconds per query on our 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 our MacBook Pro laptop.

In Figure 20, we show our method'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 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. Compared to SPSR's reconstruction and meshing code, our precompute time is about twice as long on our datasets. For point clouds with tens of millions of points, performing reconstruction using SPSR will be much faster (seconds vs. minutes of precompute), though we are optimistic that the performance of our method can be improved, and about the point-based, differentiable nature of our method (Section 6).

5.3 Examples and applications

Our method plugs directly into existing algorithms that rely on pointwise signed distance queries, allowing point clouds to be used directly in geometry routines.

Offset surfaces and Booleans. Our method extends classic SDF operations to point cloud data. In Figure 22, we contour offset surfaces to an originally non-manifold and self-intersecting mesh. We also apply Boolean operations to sparse point clouds (Figure 23).

Morphological operations. Using our SDF, we can directly apply morphological operations to point cloud surfaces. Figure 24 shows the result of repeatedly applying erosion and dilation to several shapes. Given an offset value $c = 0.1$, we first sample the $-c$-level set of our SDF ϕ, and compute normals at the updated points using the gradient $\nabla\phi$. We then re-fit tori to the updated point cloud, and perform dilation by sampling the c-level set of the new SDF.

Direct visualization of point cloud surfaces. As our method produces SDFs, point cloud surfaces and their offsets can be visualized directly using sphere tracing [Hart 1996]. Figure 12 shows a few examples visualizing different offset surfaces in a shader. While *Harnack tracing* could similarly be used for direct visualization of harmonic functions like winding numbers [Gillespie et al. 2024], winding number surfaces can be worse quality (Figures 8, 17), and are not evenly spaced. Our reconstructions in other figures are visualized using marching cubes since our shader implementation has not been optimized for efficient sphere tracing.

Signed distance to 3D Gaussians. Our method can be used to estimate signed distance to objects represented by 3D Gaussians. We take the means of the Gaussians as points, and compute signed normals using the FaCE algorithm of Scrivener et al. [2025]. If camera data were available, we could instead compute normals using the eigenvector of each Gaussian's covariance matrix corresponding to the smallest eigenvalue, and sign them to point away from the camera. In Figure 25, we show examples using Gaussian objects from the ShapeSplat dataset [Ma et al. 2024] and Kobranov [2024].

Signed distance to neural implicits. Using our method, more general implicit functions can easily be redistanced. In Figure 26, we

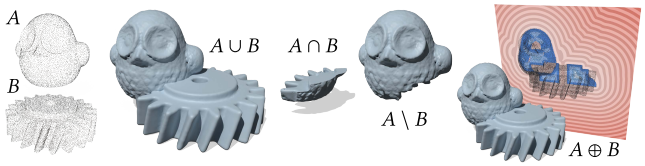


Figure 23. We apply Boolean operations to two point clouds (self-intersecting meshes 87043 and 1146170 from Thingi10K).

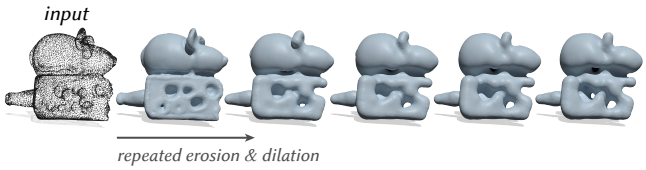


Figure 24. We can directly apply morphological operations to the surfaces underlying point clouds, without meshing.

sample a neural implicit into an oriented point cloud using the ray-casting procedure of Sharp and Jacobson [2022], and compute signed distance to this point cloud. One could also use standard ray-casting to sample implicit functions [Ling et al. 2025].

Signed distance from multiview geometry. Computer vision models such as COLMAP [Schonberger and Frahm 2016], GLOMAP [Pan et al. 2024], MASt3R [Leroy et al. 2025], and others have achieved impressive large-scale point cloud reconstruction from photos. These point clouds often have large holes, noise, and spurious "floaters", making signed distance computation extremely challenging. In Figure 16, we demonstrate our method on the dense point cloud output of COLMAP prepared by Chen et al. [2024a], who use multiview stereo data from the BlendedMVS [Yao et al. 2020] and DTU datasets [Jensen et al. 2014]. Figure 21 shows more examples using the output of GLOMAP. We extract contours using the marching cubes implementation in `scikit-image` [van der Walt et al. 2014], masking out voxels not within 8 voxels of the contour, and use a grid resolution of 1024^3 after scaling point clouds to a $[-1, 1]^3$ box. We do not apply any preprocessing to the point cloud data.

Orienting inconsistently oriented point clouds. Like most other signed distance methods, ours requires at least some orientation information hinting at the given shape's inside and outside, and does not do well without consistently oriented normals.

Our method optionally provides a lightweight way to actively orient inconsistently oriented point clouds, using an iterative procedure: for each point $\mathbf{p}_i$, we compute a weighted average $\widetilde{\mathbf{n}}_i$ of the SDF gradient evaluated at each of its neighbors $\mathbf{p}_j$. The weights are anisotropic Laplacian weights $\exp\left(-M_i(\mathbf{p}_i - \mathbf{p}_j)\right)$, where

$$M_i(\mathbf{r}) = \left(\kappa_{\min}^2 \langle \mathbf{r}, \widehat{\mathbf{v}}_{\min}\rangle^2 + \kappa_{\max}^2 \langle \mathbf{r}, \widehat{\mathbf{v}}_{\max}\rangle^2 + \varepsilon^2 \langle \mathbf{r}, \mathbf{n}_i\rangle^2\right)^{1/2}$$

is a Mahalanobis distance, and $\kappa_{\min}, \kappa_{\max}, \widehat{\mathbf{v}}_{\min}, \widehat{\mathbf{v}}_{\max}$ are the principal curvatures and directions defined in Section 4.1 that act as natural length scales and directions. We flip $\mathbf{n}_i$ if $\langle \mathbf{n}_i, \widetilde{\mathbf{n}}_i\rangle < 0$. We then recompute the fitted tori and repeat the process until convergence. Figure 27 shows the result of this process after 10 iterations,

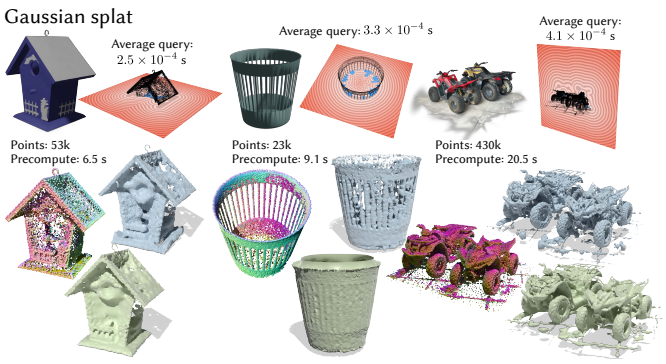


Figure 25. Scenes of 3D Gaussians can be converted to point clouds, to which we compute signed distance. Raw 3D Gaussians have inconsistent geometry and orientations that are challenging to reconstruct (*bottom row*); nevertheless, our SDF behaves well in the far-field. Query times are measured without GPU acceleration. The results of SPSR [Kazhdan and Hoppe 2013] are shown for comparison (green).

Figure 26. 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.

used to orient a point cloud whose normals are aligned with the correct direction, but 30% have the wrong orientation. However, for completely unoriented point clouds or ambiguous cases, a more sophisticated strategy is likely needed.

6 Limitations and future work

We showed that using a neural network to learn local kernel parameters is a powerful approach for signed distance estimation. Rather than use expensive, end-to-end learning in an attempt to solve a nonlinear, nonlocal eikonal problem, we decompose signed distance estimation into a problem dominated by local surface behavior. At the same time, our method yields better results than fast methods based on classical closed-form kernels. Rather than manually tweak kernel functions, bandwidths, hierarchy levels, etc. of one particular surface model, we simply use fixed-size neighborhoods with a data-driven approach that can fit to an extremely large set of easy-to-generate examples, using a far more sophisticated data-fitting model than we could ever craft purely by hand.

However, there is plenty of room for improvement. For instance, our current predictions might not be robust for point clouds whose sampling characteristics are significantly different from those seen in training (Figures 28, 29). This fact is not surprising: rather than simply learn a map from (nice) neighborhoods to kernel parameters,

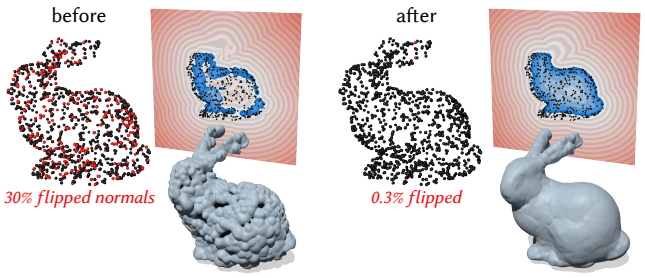


Figure 27. Our method can optionally be used to orient inconsistently oriented point clouds, by iteratively updating the signs of the normals based on the gradient of our SDF.

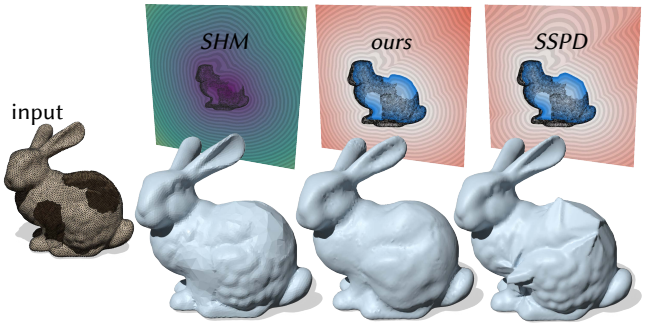


Figure 28. Our method was not trained on point clouds with large regions of completely missing data, such as this shape from Figure 27 of Feng and Crane [2024]. Their SHM fills in holes more smoothly, though our method still produces better results than a naive convolutional method (Equation 29). SHM was solved on a tet mesh with 32k vertices.

our network must now also learn how to implicitly fix, denoise, or subsample the input point cloud, effectively increasing the dimension of our feature space to include different point distributions. Making our predictions more robust, whether through a model that considers additional global context, a hierarchical approach, neighborhood size estimation, point cloud subsampling, or simply more extensive training and fine-tuning, is an interesting challenge for future work. Recent point-based methods also improve initial reconstructions through per-example image-based optimization [Chen et al. 2024b; Huang et al. 2024], something we do not explore here.

Our method in principle also allows scaling of λ to balance reconstruction detail vs. noise or sparsity; 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 30). 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].

For 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). In

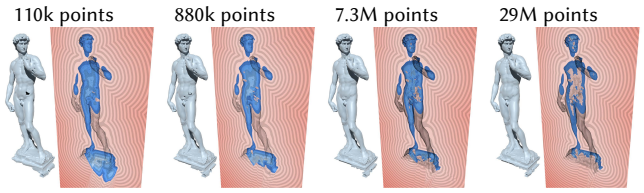
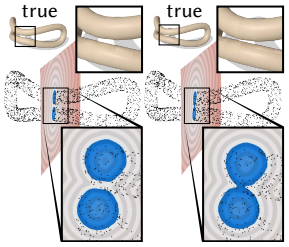


Figure 29. We visualize contoured zero level sets and an SDF slice resulting from increasingly dense samplings of the shape in Figure 20. Though non-negative level sets always behave well, the SDF interior can paradoxically get worse as the sampling gets denser. We hypothesize that as neighborhoods of 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, or simply train on point clouds of varying densities — we train on point clouds within only a small range of densities.

practice, the error goes away with high λ, and any potential overestimation is insignificant compared to the uncertainty inherent to point cloud data. Formulas in the form of Equation 16, including LogSumExp-style distance approximations, can be made conservative [Madan and Levin 2022] only under the assumption that the input geometry is *exactly* the true surface, which is almost never the case (and an especially poor approximation for point clouds).

The simplicity of our method and its roots in classical point-based methods mean it is ripe for extension. We list several possibilities.



Constraints and priors. Given the huge amount of work on meshless methods across statistics, geometry processing, engineering, and simulation, there might 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 try to enforce physical constraints tied to knowledge about the point cloud acquisition process: for example, if one knows that the point cloud came from a noiseless sensor, then the zero level set of the final SDF cannot enclose other points. Our method also cannot guarantee exact interpolation. Exact interpolation is typically undesirable — real point clouds have non-zero noise — but nonetheless could be a useful option that may avoid unintentional merging (inset).

Sharp feature extraction. Using the convolutional formula in Equation 17 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 11, 12, 22 dragon whiskers) or planar features (Figure 16). Future work can try extracting true sharp features, or fitting extra primitives.

Increasing shape coverage. Our method uses local functions strictly associated with points in the input point cloud. One might optimize for an additional point set to improve coverage of point clouds with large holes or uneven sampling patterns. Doing so might involve,

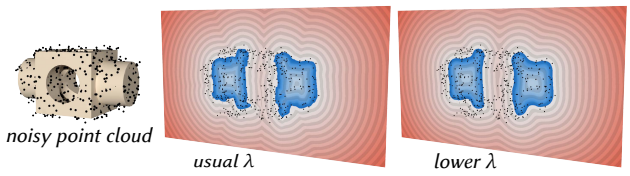


Figure 30. Though we use a simple, fixed heuristic, further tuning λ can lead to better reconstruction even using the same fitted tori. For example, if noise is high, setting λ lower leads to smoother reconstructions. Alternatively, if noise is low, setting λ higher might yield better detail recovery.

for example, sampling the zero level set of the reconstructed surface where the SDF is highly non-eikonal (where $\|\nabla\phi\|$ is far from 1).

Advanced acceleration. To accelerate SDF evaluations, for each evaluation point we simply sum over points within a fixed radius, falling back to k nearest neighbors if no points are found. This strategy can lead to artifacts if the group of points one sums over changes very discontinuously. One could instead consider a more advanced hierarchical acceleration scheme based on (potentially unbiased) Barnes-Hut summation [Madan et al. 2025] to improve accuracy while maintaining the same asymptotic performance. Using the fast multipole method [Sun et al. 2014] could provide further acceleration when performing multiple queries. Guennebaud et al. [2008]; Mercier et al. [2022] also provide performance and robustness improvements for fitting algebraic spheres to point clouds, which may inspire improvements for our method.

Our method currently involves a non-negligible precomputation cost for each new point cloud. Although our implementation already uses the fairly performant `MultiHeadAttention` function of `Flax NNX` [Heek et al. 2024], 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 has not been extensively engineered, and performance may be 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.

Compression. Our algorithm encodes a shape as a set of tori, one for each point in the input point cloud. However, some shapes could be accurately represented using fewer tori than points. One might achieve a more compressed representation by using a coarse-to-fine approach to torus-fitting, or by adapting quadric error simplification [Garland and Heckbert 1997] to merge tori after fitting.

Torus signed distance as a differentiable shape representation. Our method 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.

Generalizations to other data-fitting tasks. We showed that the two convolutional distance formulas in Equations 16 and 17 not only unify decades of geometry processing research in surface reconstruction and distance computation, but also cut to the heart of modern generative models and manifold learning techniques. We hope that our geometric insights can improve the accuracy and generalization of statistical and machine learning tasks on the whole, following works like that of Bamberger et al. [2025].

Acknowledgments

This work was funded by NSF awards 2047341, 2212290 and 2504890, and a gift from nTopology. The authors thank Alexander Belyaev and Pierre Fayolle for inspiring conversations about distance computation; Chris Wojtan, Christian Hafner, Samara Ren, Aleksei Kalinov, and Mikaël Ly for helpful discussion; Rohan Sawhney and Josua Sassen for discussion of early ideas; and Hanyu Chen and Benran Hu for providing point cloud data.

References

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

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. doi:10.1145/1108590.1108592

Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. 2001. Point set surfaces. In *Proceedings Visualization, 2001. VIS '01.* 21–29, 537. doi:10.1109/VISUAL.2001.964489

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

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

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

Jakob Andreas Bærentzen and Henrik Aanæs. 2005. Signed distance computation using the angle weighted pseudonormal. *I E E E Transactions on Visualization and Computer Graphics* 11, 3 (2005), 243 – 253. doi:10.1109/TVCG.2005.49 Copyright: 2005 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

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

Gavin Barill, Neil Dickson, Ryan Schmidt, David I.W. Levin, and Alec Jacobson. 2018. Fast Winding Numbers for Soups and Clouds. *ACM Transactions on Graphics* (2018).

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

Alexander G Belyaev and Pierre-Alain Fayolle. 2015. On variational and PDE-based distance function approximations. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 104–118.

Ted Belytschko, Y. Krongauz, Daniel Organ, Mark Fleming, and Petr Krysl. 1996. Meshless methods: An overview and recent developments. *Computer Methods in Applied Mechanics and Engineering* 139, 1 (1996), 3–47. doi:10.1016/S0045-7825(96)01078-X

Yizhak Ben-Shabat and Stephen Gould. 2020. DeepFit: 3D Surface Fitting via Neural Network Weighted Least Squares. arXiv:2003.10826 [cs.CV] https://arxiv.org/abs/2003.10826

Yizhak Ben-Shabat, Chamin Hewa Koneputugodage, and Stephen Gould. 2023. DiGS : Divergence guided shape implicit neural representation for unoriented point clouds. arXiv:2106.10811 [cs.CV] https://arxiv.org/abs/2106.10811

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. doi:ttps://doi.org/10.1007/978-1-4757-3069-2

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. doi:10.1109/34.3881

Giulio Biroli, Tony Bonnaire, Valentin de Bortoli, and Marc Mézard. 2024. Dynamical Regimes of Diffusion Models. arXiv:2402.18491 [cs.LG] https://arxiv.org/abs/2402.18491

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

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. doi:10.1093/imanum/draa038

Jose Luis Blanco and Pranjal Kumar Rai. 2014. nanoflann: a C++ header-only fork of FLANN, a library for Nearest Neighbor (NN) with KD-trees. https://github.com/jlblancoc/nanoflann.

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

Jules Bloomenthal. 1997. Bulge Elimination in Convolution Surfaces. *Computer Graphics Forum* 16, 1 (1997), 31–41. doi:10.1111/1467-8659.114

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. doi:10.1145/122718.122757

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

Alexandre Boulch and Renaud Marlet. 2022. POCO: Point Convolution for Surface Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 6302–6314.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. *JAX: composable transformations of Python+NumPy programs.* http://github.com/jax-ml/jax

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. doi:10.1111/j.1467-8659.2011.02058.x

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. doi:10.1016/0167-8655(94)90006-X

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. arXiv:2301.13188 [cs.CR] https://arxiv.org/abs/2301.13188

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. doi:10.1145/383259.383266

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

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. doi:10.1145/3687914

Jiong Chen, Florian Schaefer, and Mathieu Desbrun. 2024b. Lightning-fast Method of Fundamental Solutions. *ACM Trans. Graph.* (2024).

Julian Chibane, Aymen Mir, and Gerard Pons-Moll. 2020. Neural Unsigned Distance Fields for Implicit Function Learning. arXiv:2010.13938 [cs.CV] https://arxiv.org/abs/2010.13938

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

Guillaume Coiffier and Louis Béthune. 2024. 1-Lipschitz Neural Distance Fields. In *Computer Graphics Forum.* Wiley Online Library, e15128.

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

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

Martin Costabel. 1987. Principles of boundary element methods. *Computer Physics Reports* 6, 1-6 (1987), 243–274.

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.

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

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

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).*

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.

Casey Duncan. 2018. noise. https://github.com/caseman/noise.

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

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Michael Wimmer, and Niloy J. Mitra. 2020. Points2Surf: Learning Implicit Surfaces from Point Cloud Patches. *CoRR* abs/2007.10453 (2020). arXiv:2007.10453 https://arxiv.org/abs/2007.10453

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

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

Sergei Fedotov. 1999. Wave front for a reaction-diffusion system and relativistic Hamilton-Jacobi dynamics. *Phys. Rev. E* 59 (May 1999), 5040–5044. Issue 5. doi:10.1103/PhysRevE.59.5040

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

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

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. doi:10.1145/1186822.1073227

Wendell H Fleming and Panagiotis E Souganidis. 1986. PDE-viscosity solution approach to some problems of large deviations. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze* 13, 2 (1986), 171–192.

Mark I Freidlin. 1986. Geometric Optics Approach to Reaction-Diffusion Equations. *SIAM J. Appl. Math.* 46, 2 (1986), 222–232. doi:10.1137/0146016

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00).* ACM Press/Addison-Wesley Publishing Co., USA, 249–254. doi:10.1145/344779.344899

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

Weiguo Gao and Ming Li. 2024. How Do Flow Matching Models Memorize and Generalize in Sample Data Subspaces? arXiv:2410.23594 [cs.LG] https://arxiv.org/abs/2410.23594

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. doi:10.1145/258734.258849

Mark Gillespie, Denise Yang, Mario Botsch, and Keenan Crane. 2024. Ray Tracing Harmonic Functions. *ACM Trans. Graph.* 43, 4, Article 99 (July 2024), 18 pages. doi:10.1145/3658201

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

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. doi:10.1145/2451236.2451238

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

Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. 2018. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).*

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

Gaël Guennebaud, Marcel Germann, and Markus Gross. 2008. Dynamic Sampling and Rendering of Algebraic Point Set Surfaces. *Computer Graphics Forum* 27, 2 (2008), 653–662. doi:10.1111/j.1467-8659.2008.01163.x

Gaël Guennebaud and Markus Gross. 2007. Algebraic point set surfaces. *ACM Trans. Graph.* 26, 3 (July 2007), 23–es. doi:10.1145/1276377.1276406

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.

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.

Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. 2024. *Flax: A neural network library and ecosystem for JAX.* http://github.com/google/flax

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

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *ACM SIGGRAPH Conference Papers*.

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*. 4369–4379.

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

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

Anubhav Jain, Yuya Kobayashi, Takashi Shibuya, Yuhta Takida, Nasir Memon, Julian Togelius, and Yuki Mitsufuji. 2024. Classifier-Free Guidance inside the Attraction Basin May Cause Memorization. arXiv:2411.16738 [cs.CV] https://arxiv.org/abs/2411.16738

Wenzel Jakob. 2022. nanobind: tiny and efficient C++/Python bindings. https://github.com/wjakob/nanobind.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. 2014. Large scale multi-view stereopsis evaluation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 406–413.

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. doi:10.1145/3658215

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

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

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. doi:10.1016/0031-3203(95)00137-9

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

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

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.

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. doi:10.1109/34.745731

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Transactions on Graphics* 36, 4 (2017).

Vladimir Kobranov. 2024. splats. https://huggingface.co/VladKobranov/splats/tree/main. (2024).

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)*.

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

Yury Aleksandrovich Kravtsov and Yury Ilich Orlov. 1990. *Geometrical optics of inhomogeneous media*. Vol. 38. Springer.

Gerhard Kreisselmeier and Reinhold 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. doi:10.1016/B978-0-08-024488-4.50022-X

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

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. doi:10.1109/GMAP.2002.1027497

Vincent Leroy, Yohann Cabon, and Jerome Revaud. 2025. Grounding Image Matching in 3D with MASt3R. In *Computer Vision – ECCV 2024*, Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (Eds.). Springer Nature Switzerland, 71–91.

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

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. doi:10.1145/344779.344849

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. doi:10.1145/3727142

Selena Ling, Abhishek Madan, Nicholas Sharp, and Alec Jacobson. 2025. Uniform Sampling of Surfaces by Casting Rays. 44, 5 (2025).

Yaron Lipman. 2021. Phase Transitions, Distance Functions, and Implicit Neural Representations. arXiv:2106.07689 [cs.LG] https://arxiv.org/abs/2106.07689

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. doi:10.1109/CVPR46437.2021.00183

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

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. doi:10.1016/j.cagd.2009.01.004

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. doi:10.1007/s11044-010-9220-0

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

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. doi:10.1109/TPAMI.2014.2302451

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

Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. 2021. Neural-Pull: Learning Signed Distance Functions from Point Clouds by Learning to Pull Space onto Surfaces. arXiv:2011.13495 [cs.CV] https://arxiv.org/abs/2011.13495

Qi Ma, Yue Li, Bin Ren, Nicu Sebe, Ender Konukoglu, Theo Gevers, Luc Van Gool, and Danda Pani Paudel. 2024. ShapeSplat: A Large-scale Dataset of Gaussian Splats and Their Self-Supervised Pretraining.

Abhishek Madan and David IW Levin. 2022. Fast evaluation of smooth distance constraints on co-dimensional geometry. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–17.

Abhishek Madan, Nicholas Sharp, Francis Williams, Ken Museth, and David I.W. Levin. 2025. Stochastic Barnes-Hut Approximation for Fast Summation on the GPU. 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 142, 10 pages. doi:10.1145/3721238.3730725

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

Corentin Mercier, Thibault Lescoat, Pierre Roussillon, Tamy Boubekeur, and Jean-Marc Thiery. 2022. Moving level-of-detail surfaces. *ACM Trans. Graph.* 41, 4, Article 130 (July 2022), 10 pages. doi:10.1145/3528223.3530151

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. doi:10.1145/1198555.1198645

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. doi:10.1145/122718.122743

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. doi:10.1016/j.matcom.2008.01.003

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. doi:10.1145/1198555.1198649

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. doi:10.1109/SMI.2003.1199611

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

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. doi:10.1111/j.1467-8659.2009.01388.x

Linfei Pan, Daniel Barath, Marc Pollefeys, and Johannes Lutz Schönberger. 2024. Global Structure-from-Motion Revisited. In *European Conference on Computer Vision (ECCV)*.

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.

Jakiw Pidstrigach. 2022. Score-Based Generative Models Detect Manifolds. arXiv:2206.01018 [stat.ML] https://arxiv.org/abs/2206.01018

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.

Inigo Quilez. [n. d.]. distance functions. https://iquilezles.org/articles/distfunctions/.

Inigo Quilez. 2019. 2D distance and gradient functions - 2019. https://iquilezles.org/articles/distgradfunctions2d/.

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

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.

Rohan Sawhney, Ruihao Ye, Johann Korndoerfer, and Keenan Crane. 2020. FCPW: Fastest Closest Points in the West.

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

Johannes L. Schonberger and Jan-Michael Frahm. 2016. Structure-From-Motion Revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Daniel Scrivener, Daniel Cui, Ellis Coldren, S. Mazdak Abulnaga, Mikhail Bessmeltsev, and Edward Chien. 2025. Faraday Cage Estimation of Normals for Point Clouds and Ribbon Sketches. *ACM Trans. Graph.* 44, 4, Article 49 (2025), 13 pages. doi:10.1145/3731212

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.

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. doi:10.1145/3528223.3530155

Nicholas Sharp, Yousuf Soliman, and Keenan Crane. 2019. The Vector Heat Method. *ACM Trans. Graph.* 38, 3 (2019).

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. doi:10.1145/1186562.1015816

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.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 7462–7473. https://proceedings.neurips.cc/paper_files/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf

Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2022. Diffusion Art or Digital Forgery? Investigating Data Replication in Diffusion Models. arXiv:2212.03860 [cs.LG] https://arxiv.org/abs/2212.03860

Timothy Sun, Papoj Thamjaroenporn, and Changxi Zheng. 2014. Fast multipole representation of diffusion curves and points. *ACM Trans. Graph.* 33, 4, Article 53 (July 2014), 12 pages. doi:10.1145/2601097.2601187

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

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

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

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

user484. 2015. How to make a blob in 3D? https://mathematica.stackexchange.com/a/99761.

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. 2014. scikit-image: image processing in Python. *PeerJ* 2 (6 2014), e453. doi:10.7717/peerj.453

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.

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

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. doi:10.1109/SMI.2008.4547957

Zimo Wang, Cheng Wang, Taiki Yoshino, Sirui Tao, Ziyang Fu, and Tzu-Mao Li. 2025. HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition. In *CVPR*.

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. doi:10.1145/3618311

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. doi:10.1111/cgf.14752

Samuel Weidemaier, Florine Hartwig, Josua Sassen, Sergio Conti, Mirela Ben-Chen, and Martin Rumpf. 2025. SDFs from Unoriented Point Clouds using Neural Variational Heat Distances. arXiv:2504.11212 [math.NA] https://arxiv.org/abs/2504.11212

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18500–18510.

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

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

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.

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.

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.

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. doi:10.1145/3731210

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. doi:10.1016/S0010-4485(99)00042-1

Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. 2020. BlendedMVS: A Large-scale Dataset for Generalized Multi-view Stereo Networks. *Computer Vision and Pattern Recognition (CVPR)* (2020).

Lior Yariv, Omri Puny, Natalia Neverova, Oran Gafni, and Yaron Lipman. 2023. Mosaic-SDF for 3D Generative Models. *arXiv* (2023).

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

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. doi:10.1109/TVCG.2011.276

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

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 *CVPR*. https://arxiv.org/abs/2206.03087v1

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

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

A The signed Hopf-Cole transformation

We give a derivation of the signed Hopf-Cole transformation, inspired by the formulation of the viscous signed eikonal equation in Theorems 4 & 5 of Lipman [2021]. We pay special attention to the boundary conditions of the resulting screened Laplace equation, and the validity of the transformation when Ω is open. Appendix A.2 uses perturbative methods to provide another perspective on the Hopf-Cole transformation. Lastly, Appendix A.3 discusses a link between our theory and the theory of phase fields.

A.1 From the (signed) eikonal equation to a (jump) screened Laplace equation

We start with the signed viscous eikonal equation

$$\begin{aligned} \tfrac{1}{\lambda}\Delta u(\mathbf{x}) - \operatorname{sign}_\Omega(\mathbf{x})\big(\|\nabla u(\mathbf{x})\|^2 - 1\big) &= 0 \quad \mathbf{x} \notin \Omega, \\ u(\mathbf{x}) &= 0 \quad \mathbf{x} \in \Omega, \\ \tfrac{\partial u}{\partial \mathbf{n}}(\mathbf{x}) &= 1 \quad \mathbf{x} \in \Omega \end{aligned} \tag{30}$$

where $\operatorname{sign}_\Omega : \mathbb{R}^d \to \{-1, 1\}$ is a function associated with the $(d-1)$-dimensional oriented surface Ω, defined to be piecewise constant with boundary conditions

$$\operatorname{sign}_\Omega\left(\lim_{s\to 0} \mathbf{x} \pm s\mathbf{n}(\mathbf{x})\right) = \lim_{s\to 0} \operatorname{sign}_\Omega(\mathbf{x} \pm s\mathbf{n}(\mathbf{x})) = \pm 1, \quad \mathbf{x} \in \Omega \tag{31}$$

where $\mathbf{n}$ is the outward-facing normal direction to Ω.

We consider the following function of $u(\mathbf{x})$, which we refer to as the *signed Hopf-Cole transformation*,

$$w(\mathbf{x}) \coloneqq \operatorname{sign}_\Omega(\mathbf{x}) \exp\big(-\lambda \operatorname{sign}_\Omega(\mathbf{x}) u(\mathbf{x})\big). \tag{32}$$

Equation 32 can be considered a signed variant of Varadhan's formula [Varadhan 1967].

For all points $\mathbf{x} \in \mathbb{R}^d$ not on the locus of points where $\operatorname{sign}_\Omega$ changes sign, we have $0 < |w(\mathbf{x})| \leq 1$ for all $\lambda > 0$. So for all such $\mathbf{x}$, we can define the inverse transformation of Equation 32 as

$$u(\mathbf{x}) = -\frac{1}{\lambda} \operatorname{sign}_\Omega(\mathbf{x}) \log\big(\operatorname{sign}_\Omega(\mathbf{x}) w(\mathbf{x})\big). \tag{33}$$

To proceed, we need to consider the definition of the function $\operatorname{sign}_\Omega$ away from Ω, and its relationship with the well-defined function $\operatorname{sign}_w(\mathbf{x}) \coloneqq \operatorname{sign}(w(\mathbf{x}))$. We distinguish between two cases for Ω, which will eventually lead to the same conclusion.

Closed surface. If Ω is simple and closed, then Ω bounds a well-defined region A, and the only reasonable definition of $\operatorname{sign}_\Omega(\mathbf{x})$ on $\{\mathbf{x} \mid \mathbf{x} \notin \Omega\}$ is such that

$$\operatorname{sign}_\Omega(\mathbf{x}) = \begin{cases} -1 & \mathbf{x} \in A, \\ +1 & \mathbf{x} \in \mathbb{R}^d \setminus \overline{A}. \end{cases}$$

As Ω separates the domain $\mathbb{R}^d$, the solution of the viscous signed eikonal equation in Equation 30 is equivalent to the union of the solutions of two independent viscous signed eikonal equations: one defined on A with boundary Ω (the "interior"), and one defined on $\mathbb{R}^d \setminus \overline{A}$ with boundary $-\Omega$ (the "exterior"), where $-\Omega$ denotes Ω with opposite orientation.

Let $u^+ : \mathbb{R}^d \setminus \overline{A} \to \mathbb{R}$ be the solution to the exterior problem, and $u^- : A \to \mathbb{R}$ the solution to the interior problem (and similarly for $w^\pm(\mathbf{x})$). Using the signed Hopf-Cole transformation, we obtain:

$$\begin{aligned} \nabla u^\pm(\mathbf{x}) &= \mp\tfrac{1}{\lambda}\tfrac{\nabla w}{w}, \\ \Delta u^\pm(\mathbf{x}) &= \mp\tfrac{1}{\lambda}\left(\tfrac{\Delta w}{w} - \tfrac{\|\nabla w\|^2}{w^2}\right). \end{aligned}$$

Applying these expressions to the viscous signed eikonal equation in Equation 30, we obtain a screened Laplace equation in $w(\mathbf{x})$ for both the interior and exterior regions:

$$\Delta w^\pm(\mathbf{x}) - \lambda^2 w^\pm(\mathbf{x}) = 0. \tag{34}$$

The Dirichlet boundary conditions of $w^+(\mathbf{x})$ and $w^-(\mathbf{x})$ are

$$\begin{aligned} & w^\pm(\mathbf{x}) = \operatorname{sign}_\Omega(\mathbf{x}) \exp\big(-\lambda \operatorname{sign}_\Omega(\mathbf{x}) u^\pm(\mathbf{x})\big), && \mathbf{x} \in \Omega, \\ \Rightarrow\ & w^\pm(\mathbf{x}) = \operatorname{sign}_\Omega(\mathbf{x}) && \text{since } u^\pm(\mathbf{x}) = 0 \text{ for } \mathbf{x} \in \Omega. \end{aligned} \tag{35}$$

The Neumann boundary conditions for both $w^+(\mathbf{x})$ and $w^-(\mathbf{x})$ are identical w.r.t. the normals of Ω:

$$\frac{\partial w^\pm(\mathbf{x})}{\partial \mathbf{n}(\mathbf{x})} = -\lambda.$$

Putting together the solutions $w^+(\mathbf{x})$ and $w^-(\mathbf{x})$, we obtain the following jump screened Laplace equation for $w(\mathbf{x})$:

$$\begin{aligned} \Delta w(\mathbf{x}) - \lambda^2 w(\mathbf{x}) &= 0 && \mathbf{x} \notin \Omega, \\ w^\pm(\mathbf{x}) &= \operatorname{sign}^\pm_\Omega(\mathbf{x}) && \mathbf{x} \in \Omega, \\ \tfrac{\partial w}{\partial \mathbf{n}}(\mathbf{x}) &= -\lambda && \mathbf{x} \in \Omega. \end{aligned} \tag{36}$$

where $w^\pm(\mathbf{x}) \coloneqq w(\mathbf{x}^\pm)$ and $\operatorname{sign}^\pm_\Omega(\mathbf{x}) \coloneqq \operatorname{sign}_\Omega(\mathbf{x}^\pm)$ (where $\mathbf{x}^\pm \coloneqq \lim_{s\to 0} \mathbf{x} \pm s\mathbf{n}(\mathbf{x})$). Equation 36 implies that $\operatorname{sign}_\Omega = \operatorname{sign}_w$.

Open surface. When Ω is open, the function $\operatorname{sign}_\Omega(\mathbf{x})$ is no longer easily defined away from Ω. However, we still know the Dirichlet boundary conditions of $\operatorname{sign}_\Omega$ (Equation 31), which from Equation 35 equal those of $w(\mathbf{x})$. Though we have not yet introduced any constraints between $\operatorname{sign}_\Omega(\mathbf{x})$ and $\operatorname{sign}_w(\mathbf{x})$ at points $\mathbf{x} \notin \Omega$, we make the ansatz $\operatorname{sign}_\Omega = \operatorname{sign}_w$, as we proved in the case of closed Ω. We re-write the signed Hopf-Cole transformation in Equation 32 as

$$w(\mathbf{x}) = \operatorname{sign}_w(\mathbf{x}) \exp\big(-\lambda \operatorname{sign}_w(\mathbf{x}) u(\mathbf{x})\big) \tag{37}$$

with inverse

$$u(\mathbf{x}) = -\frac{1}{\lambda} \operatorname{sign}_w(\mathbf{x}) \log\big(\operatorname{sign}_w(\mathbf{x}) w(\mathbf{x})\big) = -\frac{1}{\lambda} \operatorname{sign}_w(\mathbf{x}) \log|w(\mathbf{x})|.$$

Defining the signed Hopf-Cole transformation essentially necessitates $\operatorname{sign}_\Omega = \operatorname{sign}_w$, otherwise the argument of the logarithm in the inverse transformation can become negative. Using our ansatz guarantees that the inverse transformation is well-defined, as above we

established by Equation 32 that $|w| > 0$ always. Using the updated signed Hopf-Cole transformation, we obtain the expressions:

$$\begin{aligned}\nabla u &= -\frac{1}{\lambda}\big(\log|w|\nabla\,\mathrm{sign}_w + \mathrm{sign}_w\nabla\log|w|\big),\\ \Delta u &= -\frac{1}{\lambda}\big(2\nabla\log|w|\cdot\nabla\,\mathrm{sign}_w + \log|w|\Delta\,\mathrm{sign}_w + \mathrm{sign}_w\Delta\log|w|\big).\end{aligned} \tag{38}$$

Computing the derivatives on the right-hand side of these expressions is possible only at $\mathbf{x}$ not on the locus ∂w of points where w changes sign. At such $\mathbf{x}$, $|w(\mathbf{x})|$ and $\mathrm{sign}_w(\mathbf{x})$ are continuous, and we may safely take their gradients. The boundary conditions of w — established above by Equations 31 and 35 — imply that ∂w is a superset of Ω.

For all $\mathbf{x}$ not on ∂w, $\nabla\,\mathrm{sign}_w(\mathbf{x}) = 0$ and $\Delta\,\mathrm{sign}_w(\mathbf{x}) = 0$, and we can continue from Equation 38:

$$\begin{aligned}\nabla u(\mathbf{x}) &= -\frac{1}{\lambda}\,\mathrm{sign}_w(\mathbf{x})\nabla\log|w(\mathbf{x})|\\ &= -\frac{1}{\lambda}\underbrace{\mathrm{sign}_w(\mathbf{x})w(\mathbf{x})}_{|w(\mathbf{x})|}\frac{\nabla w(\mathbf{x})}{|w(\mathbf{x})|^2}\\ &= -\frac{1}{\lambda}\nabla w(\mathbf{x})/|w(\mathbf{x})|,\\ \Delta u(\mathbf{x}) &= -\frac{1}{\lambda}\frac{|w(\mathbf{x})|\Delta w(\mathbf{x}) - \nabla w(\mathbf{x})\cdot\frac{w(\mathbf{x})\nabla w(\mathbf{x})}{|w(\mathbf{x})|}}{|w(\mathbf{x})|^2}\\ &= -\frac{1}{\lambda}\left(\frac{\Delta w(\mathbf{x})}{|w(\mathbf{x})|} - \frac{w(\mathbf{x})\|\nabla w(\mathbf{x})\|^2}{|w(\mathbf{x})|^3}\right)\end{aligned}$$

$$\begin{aligned}&\Rightarrow \frac{1}{\lambda}\Delta u(\mathbf{x}) - \mathrm{sign}_w(\mathbf{x})\Big(\|\nabla u(\mathbf{x})\|^2 - 1\Big) = 0\\ &\Rightarrow -\frac{1}{\lambda^2}\left(\frac{\Delta w(\mathbf{x})}{|w(\mathbf{x})|} - \frac{w(\mathbf{x})\|\nabla w(\mathbf{x})\|^2}{|w(\mathbf{x})|^3}\right)\\ &\qquad - \mathrm{sign}_w(\mathbf{x})\left(\frac{1}{\lambda^2}\frac{\|\nabla w(\mathbf{x})\|^2}{|w(\mathbf{x})|^2} - 1\right) = 0\\ &\Rightarrow -\frac{1}{\lambda^2}\,\mathrm{sign}_w(\mathbf{x})\left(\frac{\Delta w(\mathbf{x})}{|w(\mathbf{x})|} - \frac{w(\mathbf{x})\|\nabla w(\mathbf{x})\|^2}{|w(\mathbf{x})|^3}\right)\\ &\qquad -\left(\frac{1}{\lambda^2}\frac{\|\nabla w(\mathbf{x})\|^2}{|w(\mathbf{x})|^2} - 1\right) = 0\\ &\Rightarrow -\frac{1}{\lambda^2}\,\mathrm{sign}_w(\mathbf{x})\frac{\Delta w(\mathbf{x})}{|w(\mathbf{x})|}\\ &\qquad +\frac{1}{\lambda^2}\underbrace{\mathrm{sign}_w(\mathbf{x})w(\mathbf{x})}_{|w(\mathbf{x})|}\frac{\|\nabla w(\mathbf{x})\|^2}{|w(\mathbf{x})|^3} - \frac{1}{\lambda^2}\frac{\|\nabla w(\mathbf{x})\|^2}{|w(\mathbf{x})|^2} + 1 = 0\\ &\Rightarrow -\frac{1}{\lambda^2}\,\mathrm{sign}_w(\mathbf{x})\frac{\Delta w(\mathbf{x})}{|w(\mathbf{x})|} + 1 = 0\\ &\Rightarrow -\frac{1}{\lambda^2}\frac{\Delta w(\mathbf{x})}{w(\mathbf{x})} + 1 = 0\\ &\Rightarrow \Delta w(\mathbf{x}) - \lambda^2 w(\mathbf{x}) = 0.\end{aligned}$$

We arrive at the same screened Laplace equation for $w(\mathbf{x})$ as before. This screened Laplace equation is valid for all $\mathbf{x} \notin \Omega$.

We already established the Dirichlet boundary conditions of $w(\mathbf{x})$, so all that remains is to establish the Neumann boundary conditions:

$$\begin{aligned}1 &= \nabla u(\mathbf{x})\cdot\mathbf{n} = -\tfrac{1}{\lambda|w(\mathbf{x})|}(\nabla w(\mathbf{x})\cdot\mathbf{n}), && \mathbf{x}\in\Omega\\ \Rightarrow \tfrac{\partial w}{\partial\mathbf{n}}(\mathbf{x}) &= -\lambda && \text{using } |w(\mathbf{x})| = 1,\end{aligned}$$

which matches the Neumann boundary conditions we obtained in Equation 36 when we assumed Ω was closed.

A.2 WKB approximation

We show an alternative derivation that links the screened Laplace equation to the eikonal equation through the *Wentzel-Kramers-Brillouin (WKB) approximation*. The WKB approximation is a classical methodology for obtaining a global approximation to the solution of a linear differential equation whose highest derivative is multiplied by a small constant — such as our screened Laplace equation in Equation 36, which we rewrite equivalently as:

$$\frac{1}{\lambda^2}\Delta w(\mathbf{x}) - w(\mathbf{x}) = 0, \tag{39}$$

and consider its limit behavior as $\lambda \to \infty$. We paraphrase here the derivation given in an example in Bender and Orszag [1999, §10].

The WKB approximation seeks approximate solutions of the form:

$$w(\mathbf{x}) \sim \exp\left(S(\mathbf{x})/\delta\right), \qquad \delta \to 0^+$$

motivated by the exponential behavior of dissipative and diffusive effects. Using series expansions of the phase $S(\mathbf{x})$ in powers of δ yields the power series

$$w(\mathbf{x}) \sim \exp\left(\frac{1}{\delta}\sum_{n=0}^{\infty}\delta^n S_n(\mathbf{x})\right), \qquad \delta \to 0^+.$$

This WKB approximation has derivatives (as $\delta \to 0^+$)

$$\begin{aligned}\nabla w(\mathbf{x}) &\sim \left(\frac{1}{\delta}\sum_{n=0}^{\infty}\delta^n\nabla S_n(\mathbf{x})\right)\exp\left(\frac{1}{\delta}\sum_{n=0}^{\infty}\delta^n S_n(\mathbf{x})\right)\\ \Delta w(\mathbf{x}) &\sim \left[\left(\frac{1}{\delta}\sum_{n=0}^{\infty}\delta^n\nabla S_n(\mathbf{x})\right)^2 + \frac{1}{\delta}\sum_{n=0}^{\infty}\delta^n\Delta S_n(\mathbf{x})\right]\exp\left(\frac{1}{\delta}\sum_{n=0}^{\infty}\delta^n S_n(\mathbf{x})\right)\end{aligned}$$

and substituting into the screened Laplace equation yields

$$\begin{aligned}&\frac{1}{\lambda^2}\left[\left(\frac{1}{\delta}\sum_{n=0}^{\infty}\delta^n\nabla S_n(\mathbf{x})\right)^2 + \frac{1}{\delta}\sum_{n=0}^{\infty}\delta^n\Delta S_n(\mathbf{x})\right] - 1 = 0\\ &\Rightarrow \frac{1}{\lambda^2\delta^2}\left(\sum_{n=0}^{\infty}\delta^n\nabla S_n(\mathbf{x})\right)^2 + \frac{1}{\lambda^2\delta}\sum_{n=0}^{\infty}\delta^n\Delta S_n(\mathbf{x}) = 1.\end{aligned}$$

The largest term on the left-hand side is the first term $1/\lambda^2\delta^2\|\nabla S_0(\mathbf{x})\|^2$ of the leftmost sum — all further terms are multiplied by higher powers of the small scale parameter δ. By dominant balance, this term must have the same order of magnitude as the right-hand side; hence we determine that $\delta \sim 1/\lambda$. Taking $\delta = 1/\lambda$ yields

$$\left(\sum_{n=0}^{\infty}\frac{1}{\lambda^n}\nabla S_n(\mathbf{x})\right)^2 + \sum_{n=0}^{\infty}\frac{1}{\lambda^{n+1}}\Delta S_n(\mathbf{x}) = 1.$$

By equating powers of $1/\lambda$, we obtain a recursive formula for S_n:

$$\begin{cases} \|\nabla S_0(\mathbf{x})\|^2 = 1, \\ \Delta S_{n-1}(\mathbf{x}) + \sum_{j=0}^{n} \nabla S_j(\mathbf{x}) \cdot \nabla S_{n-j}(\mathbf{x}) = 0, \quad n \geq 1. \end{cases}$$

It is possible that $\delta \sim c/\lambda$ for some constant c, though asymptotic analysis is only interested in the order of magnitude of δ.

We see that the first equation is an eikonal equation, whose boundary conditions are determined by those of the original screened Laplace equation in Equation 39. This analysis implies that the leading term of $w(\mathbf{x})$ in the large-λ regime is the solution to an eikonal equation — equivalent to what is implied by the Hopf-Cole transformation and Varadhan's formula.

Making these types of approximations is common in optics to analyze the Helmholtz equation — effectively Equation 39 but with an imaginary λ, corresponding to the frequency of light. For example, using only the leading term corresponds to making a geometric optics assumption, from which the eikonal equation is derived in what is called the *high-frequency limit*. An alternative derivation is possible by starting from the boundary integral representation of the solution to the screened Laplace equation (Equation 14), then invoking the so-called *stationary phase approximation* to keep only the leading contribution of the exponential term in the Yukawa potential, arriving again at the eikonal equation. Kravtsov and Orlov [1990] provide details about the use of these approximations in geometric optics, including in cases of heterogeneous optical media (giving rise to an eikonal equation with a spatially-varying metric). For more theory connecting viscosity solutions, Hamilton-Jacobi equations, and general reaction-diffusion equations, we refer to Freidlin [1986], Fleming and Souganidis [1986], and Fedotov [1999].

A.3 Phase fields

The boundary-layer perspective also yields a connection between viscous eikonal equations and phase fields, also noted by Lipman [2021]. In particular, consider the optimization

$$\begin{array}{ll} \min_{v:M\to\mathbb{R}} & \int_M \frac{1}{\lambda^2}\|\nabla v(\mathbf{x})\|^2 + (|v(\mathbf{x})| - 1)^2 \, \mathrm{d}\mathbf{x} \\ \text{s.t.} & v(\mathbf{x}) = 0 \qquad\qquad \mathbf{x} \in \partial M \end{array}$$

whose objective is a variant of the *Modica-Mortola functional* common in image processing. We assume for simplicity that M is a closed region — we discuss the case of an open region in detail in Appendix A.1. Expanding the first term of the objective (and using angle brackets to denote the L^2 inner product) yields

$$\begin{aligned} \int_M \frac{1}{\lambda^2}\|\nabla v(\mathbf{x})\|^2 \, \mathrm{d}\mathbf{x} &= \frac{1}{\lambda^2}\langle \nabla v, \nabla v\rangle_M \\ &= \frac{1}{\lambda^2}\left(-\langle \Delta v, v\rangle_M + \langle \mathbf{n}\cdot\nabla v, v\rangle_{\partial M}\right) \quad \text{(Stokes' theorem)} \end{aligned}$$

where $\mathbf{n}$ is the unit normal to ∂M. To derive the necessary conditions for optimality, we group the interior and boundary terms of the objective, differentiate the expressions w.r.t. v, and set to 0, to obtain

$$\begin{array}{rcll} \frac{1}{\lambda^2}\Delta v(\mathbf{x}) - v(\mathbf{x}) & = & -\operatorname{sign}_v(\mathbf{x}) & \mathbf{x} \in M \\ v(\mathbf{x}) & = & 0 & \mathbf{x} \in \partial M \\ \frac{\partial v}{\partial \mathbf{n}}(\mathbf{x}) & = & \mu(\mathbf{x}) & \mathbf{x} \in \partial M \end{array}$$

where $\mu : \partial M \to \mathbb{R}$ is a Lagrange multiplier. If $\operatorname{sign}_v(\mathbf{x}) = 1$ everywhere in M, then applying the change of variable $w(\mathbf{x}) = 1 - v(\mathbf{x})$ yields the jump screened Laplace equation in Equation 36 where $\operatorname{sign}_w(\mathbf{x}) = 1$, except with Neumann boundary conditions equal to $-\mu(\mathbf{x})$. If $\operatorname{sign}_v(\mathbf{x}) = -1$ everywhere in M, applying the change of variable $w(\mathbf{x}) = -1 - v(\mathbf{x})$ yields Equation 36, again with Neumann boundary conditions equal to $-\mu(\mathbf{x})$. Lastly, we can rule out changes of $\operatorname{sign}_v(\mathbf{x})$ inside M using arguments similar to those in Appendix A.1. We note that, though the Neumann boundary conditions are different from those in Equation 36, the integral expressions for distance in Section 2 need only that they stay the same for both $\operatorname{sign}_v(\mathbf{x}) = -\operatorname{sign}_w(\mathbf{x}) = \pm 1$.

B The jump screened Laplace equation as a Poisson equation

We prove we can rewrite the jump screened Laplace equation in Equation 9,

$$\begin{array}{rcll} \Delta w(\mathbf{x}) - \lambda^2 w(\mathbf{x}) & = & 0 & \mathbf{x} \notin \Omega \\ w^{\pm}(\mathbf{x}) & = & \pm 1 & \mathbf{x} \in \Omega \\ \frac{\partial w^+}{\partial \mathbf{n}}(\mathbf{x}) & = & \frac{\partial w^-}{\partial \mathbf{n}}(\mathbf{x}) & \mathbf{x} \in \Omega, \end{array}$$

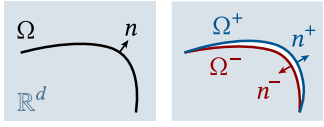
as the screened Poisson equation without boundary in Equation 11,

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

In general, the solution $u(\mathbf{x})$ to a screened Laplace equation on a domain $M \subset \mathbb{R}^d$, with Dirichlet and Neumann boundary conditions $u(\mathbf{x}) = g(\mathbf{x})$ and $\frac{\partial u}{\partial \mathbf{n}} = h(\mathbf{x})$ (resp.) on ∂M, admits the following boundary integral representation:

$$u(\mathbf{x}) = \int_{\partial M} G^\lambda(\mathbf{x}, \mathbf{z}) h(\mathbf{z}) - \frac{\partial G^\lambda(\mathbf{x}, \mathbf{z})}{\partial \mathbf{n}_\mathbf{z}} g(\mathbf{z}) \, \mathrm{d}A(\mathbf{z}), \quad \mathbf{x} \in M$$

where G^λ is Green's function for the screened Laplace operator (Equation 12), and A is the area measure. If there is a source term $f(\mathbf{x})$ (that is, one is solving a screened Poisson rather than screened Laplace equation) then the above representation has an additional term $\int_M G^\lambda(\mathbf{x}, \mathbf{y}) f(\mathbf{y}) \, \mathrm{d}\mathbf{y}$ — a volume integral — that incorporates the contribution of $f(\mathbf{x})$ over the domain M.



In our case, the domain is $\mathbb{R}^d \setminus \Omega$, and the domain boundary is the oriented curve or surface Ω. The boundary can be thought of as $\Omega = \Omega^+ \cup \Omega^-$, where Ω^+ and Ω^- are the positive and negative "sides" of Ω (see Section 2). In this way, even when Ω is open, Ω can be thought of as a "slit" in $\mathbb{R}^d$ that still acts as a closed boundary to $M \setminus \Omega$ (inset). Then for points

$\mathbf{x} \notin \Omega$, the solution to Equation 9 can be expressed as

$$\begin{aligned} w(\mathbf{x}) &= \int_{\Omega^+} \frac{\partial G^\lambda(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}^+} w^+(\mathbf{y}) + G^\lambda(\mathbf{x},\mathbf{y}) \frac{\partial w^+(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}^+} \,\mathrm{d}A(\mathbf{y}) \\ &\quad + \int_{\Omega^-} \frac{\partial G^\lambda(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}^-} w^-(\mathbf{y}) + G^\lambda(\mathbf{x},\mathbf{y}) \frac{\partial w^-(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}^-} \,\mathrm{d}A(\mathbf{y}) \\ &= \int_{\Omega^+} \frac{\partial G^\lambda(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}^+} w^+(\mathbf{y}) + G^\lambda(\mathbf{x},\mathbf{y}) \frac{\partial w^+(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}^+} \,\mathrm{d}A(\mathbf{y}) \\ &\quad - \int_{\Omega^+} \frac{\partial G^\lambda(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}^+} w^-(\mathbf{y}) + G^\lambda(\mathbf{x},\mathbf{y}) \frac{\partial w^-(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}^+} \,\mathrm{d}A(\mathbf{y}) \\ &= \int_{\Omega^+} \frac{\partial G^\lambda(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} \left(w^+(\mathbf{y}) - w^-(\mathbf{y})\right) \,\mathrm{d}A(\mathbf{y}) \\ &\quad + \int_{\Omega^+} G^\lambda(\mathbf{x},\mathbf{y}) \cancel{\left(\frac{\partial w^+(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} - \frac{\partial w^-(\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} \right)} \,\mathrm{d}A(\mathbf{y}) \\ &= 2 \int_{\Omega^+} \frac{\partial G^\lambda(\mathbf{x},\mathbf{y})}{\partial \mathbf{n}_\mathbf{y}} \,\mathrm{d}A(\mathbf{y}) \end{aligned}$$

at interior points $\mathbf{x}$. In summary, the integrals with Neumann boundary data cancel, leaving just an integral with Dirichlet data. As an alternative to our "slit" argument, we can derive the same result using the boundary integral representation for linear elliptic PDEs with double-sided boundary conditions [Costabel 1987, Section 3].

Meanwhile, the solution to Equation 11 using the earlier representation formula is

$$\begin{aligned} w(\mathbf{x}) &= \int_{\mathbb{R}^d} G^\lambda(\mathbf{x},\mathbf{y}) \left(-2\nabla \cdot \mathbf{n}_\mathbf{y}\right) \mu_\Omega(\mathbf{y}) \,\mathrm{d}\mathbf{y} \\ &= -2 \int_\Omega G^\lambda(\mathbf{x},\mathbf{z}) \left(\nabla \cdot \mathbf{n}_\mathbf{z}\right) \,\mathrm{d}A(\mathbf{z}) \\ &= -2 \left(\cancel{\int_{\partial\Omega} G^\lambda(\mathbf{x},\mathbf{z}) (\mathbf{n}_\mathbf{z} \cdot \mathbf{t}_\mathbf{z}) \,\mathrm{d}l(\mathbf{z})} - \int_\Omega \nabla G^\lambda(\mathbf{x},\mathbf{z}) \cdot \mathbf{n}_\mathbf{z} \,\mathrm{d}A(\mathbf{z}) \right) \\ &\quad \text{(integration by parts)} \\ &= 2 \int_\Omega \frac{\partial G^\lambda(\mathbf{x},\mathbf{z})}{\partial \mathbf{n}_\mathbf{z}} \,\mathrm{d}A(\mathbf{z}) \end{aligned}$$

where $\mathbf{t}_\mathbf{z}$ is the co-normal vector at $\mathbf{z} \in \partial\Omega$ — tangent to Ω and orthogonal to $\partial\Omega$, thus $\mathbf{n}_\mathbf{z} \cdot \mathbf{t}_\mathbf{z} = 0$ — and l is the arc-length measure. The change from the volume to the area measure in the first step of the above sequence requires a careful definition of $\mu_\Omega(\mathbf{x})$, for example as in Osher et al. [2004, §1.5, Equations 1.20–1.21] for an implicit representation of Ω.

C Derivation of Principal Curvatures

We derive the expressions for the principal curvatures and directions of a polynomial surface (Section 4.1), which amount to standard Monge patch calculations.

We denote the first and second fundamental forms as

$$\mathrm{I} = \begin{bmatrix} E & F \\ F & G \end{bmatrix}, \mathrm{II} = \begin{bmatrix} e & f \\ f & g \end{bmatrix}.$$

The polynomial surface associated with point $\mathbf{p}_i$ is parameterized by s and t as

$$Q_i^*(s,t) = \mathbf{p}_i + s \cdot \mathbf{s}_i + t \cdot \mathbf{t}_i + Q_i(s,t) \cdot \mathbf{n}_i$$

where $Q_i(s,t)$ is the polynomial defined in the local coordinate system at point $\mathbf{p}_i$. The unit normal $\mathbf{n}_i^*(s,t)$ to the surface is

$$\mathbf{n}_i^*(s,t) = \frac{\frac{\partial Q_i^*}{\partial s} \times \frac{\partial Q_i^*}{\partial t}}{\left\| \frac{\partial Q_i^*}{\partial s} \times \frac{\partial Q_i^*}{\partial t} \right\|} = \frac{\mathbf{n}_i - \mathbf{s}_i \frac{\partial Q_i}{\partial s} - \mathbf{t}_i \frac{\partial Q_i}{\partial t}}{\sqrt{1 + \left(\frac{\partial Q_i}{\partial s}\right)^2 + \left(\frac{\partial Q_i}{\partial t}\right)^2}}.$$

Using the basis $\{{}^{\partial Q_i^*}/_{\partial s}, {}^{\partial Q_i^*}/_{\partial t}\}$ of the supporting plane, we have

$$\begin{aligned} E &= \frac{\partial Q_i^*}{\partial s} \cdot \frac{\partial Q_i^*}{\partial s} = 1 + \left(\frac{\partial Q_i}{\partial s}\right)^2 \\ F &= \frac{\partial Q_i^*}{\partial s} \cdot \frac{\partial Q_i^*}{\partial t} = \frac{\partial Q_i}{\partial s} \frac{\partial Q_i}{\partial t} \\ G &= \frac{\partial Q_i^*}{\partial t} \cdot \frac{\partial Q_i^*}{\partial t} = 1 + \left(\frac{\partial Q_i}{\partial t}\right)^2 \end{aligned}$$

and

$$\begin{aligned} e &= \mathbf{n}_i^* \cdot \frac{\partial^2 Q_i^*}{\partial s^2} = \frac{\frac{\partial^2 Q_i}{\partial s^2}}{\sqrt{1 + \left(\frac{\partial Q_i}{\partial s}\right)^2 + \left(\frac{\partial Q_i}{\partial t}\right)^2}} \\ f &= \mathbf{n}_i^* \cdot \frac{\partial^2 Q_i^*}{\partial s \partial t} = \frac{\frac{\partial^2 Q_i}{\partial s \partial t}}{\sqrt{1 + \left(\frac{\partial Q_i}{\partial s}\right)^2 + \left(\frac{\partial Q_i}{\partial t}\right)^2}} \\ g &= \mathbf{n}_i^* \cdot \frac{\partial^2 Q_i^*}{\partial t^2} = \frac{\frac{\partial^2 Q_i}{\partial t^2}}{\sqrt{1 + \left(\frac{\partial Q_i}{\partial s}\right)^2 + \left(\frac{\partial Q_i}{\partial t}\right)^2}}. \end{aligned}$$

At $(s,t) = (0,0)$, we have

$$\begin{aligned} &\frac{\partial Q_i}{\partial s}(0,0) = a_{1,0}, \qquad \frac{\partial Q_i}{\partial t}(0,0) = a_{0,1}, \\ &\frac{\partial^2 Q_i}{\partial s^2}(0,0) = 2a_{2,0}, \quad \frac{\partial^2 Q_i}{\partial t^2}(0,0) = 2a_{0,2}, \quad \frac{\partial^2 Q_i}{\partial s \partial t}(0,0) = a_{1,1}. \end{aligned}$$

so the fundamental forms simplify to

$$\mathrm{II} = \frac{1}{A} \begin{bmatrix} 2a_{2,0} & a_{1,1} \\ a_{1,1} & 2a_{0,2} \end{bmatrix}, \quad \mathrm{I} = \begin{bmatrix} 1 + a_{1,0}^2 & a_{1,0}a_{0,1} \\ a_{1,0}a_{0,1} & 1 + a_{0,1}^2 \end{bmatrix}$$

where $A \coloneqq \sqrt{1 + a_{0,1}^2 + a_{1,0}^2}$. The curvatures κ_+, κ_- satisfy

$$\kappa_\pm = H \pm \sqrt{H^2 - K}$$

with

$$H = \frac{a_{0,2}(1 + a_{1,0}^2) + a_{2,0}(1 + a_{0,1}^2) - a_{1,1}a_{1,0}a_{0,1}}{A^3}, \quad K = \frac{4a_{0,2}a_{2,0} - a_{1,1}^2}{A^4}.$$

Solving $(\mathrm{II} - \kappa \mathrm{I})\,\mathbf{w} = 0$ gives a system of two equations for the eigenvectors, which are parallel to the vectors

$$\mathbf{w}_\pm = \left[\kappa_\pm a_{1,0}a_{0,1}A - a_{1,1} \quad 2a_{2,0} - \kappa_\pm \left(1 + a_{1,0}^2\right) A \right].$$

The vectors $\mathbf{w}_\pm$ are expressed in the local coordinate system. The corresponding 3D vectors are

$$\begin{aligned} \mathbf{v}_\pm &= [\mathbf{w}_\pm]_x \cdot \frac{\partial Q_i^*}{\partial s}(0,0) + [\mathbf{w}_\pm]_y \cdot \frac{\partial Q_i^*}{\partial t}(0,0) \\ &= [\mathbf{w}_\pm]_x \cdot \left(\mathbf{s}_i + a_{1,0}\mathbf{n}_i\right) + [\mathbf{w}_\pm]_y \cdot \left(\mathbf{t}_i + a_{0,1}\mathbf{n}_i\right). \end{aligned}$$

The principal directions can be found by normalizing $\mathbf{v}_\pm$.

D Gradient expressions

We give gradient expressions for the self-normalized convolutional distance formula in Equation 17, which we use to estimate the eikonal loss in Equation 27. We also give an expression for the Laplacian, which we do not use but may be helpful for future work.

The gradient is given by

$$\nabla_{\mathbf{x}}\widehat{d^{\lambda}}(\mathbf{x}) = \frac{\int_{\Omega} \exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\left(\nabla_{\mathbf{x}} g(\mathbf{x},\mathbf{z}) - \lambda\frac{\mathbf{x}-\mathbf{z}}{\|\mathbf{x}-\mathbf{z}\|} g(\mathbf{x},\mathbf{z})\right)\,\mathrm{d}A(\mathbf{z})}{\int_{\Omega}\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\,\mathrm{d}A(\mathbf{z})} + \widehat{d^{\lambda}}(\mathbf{x})\frac{\lambda\int_{\Omega}\frac{\mathbf{x}-\mathbf{z}}{\|\mathbf{x}-\mathbf{z}\|}\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\,\mathrm{d}A(\mathbf{z})}{\int_{\Omega}\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\,\mathrm{d}A(\mathbf{z})}.$$

For convenience, let $\nabla_{\mathbf{x}}\widehat{d^{\lambda}}(\mathbf{x}) = \mathbf{A} + \widehat{d^{\lambda}}(\mathbf{x})\mathbf{B}$ where

$$\mathbf{A} \coloneqq \frac{\int_{\Omega} \exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\left(\nabla_{\mathbf{x}} g(\mathbf{x},\mathbf{z}) - \lambda\frac{\mathbf{x}-\mathbf{z}}{\|\mathbf{x}-\mathbf{z}\|} g(\mathbf{x},\mathbf{z})\right)\,\mathrm{d}A(\mathbf{z})}{\int_{\Omega}\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\,\mathrm{d}A(\mathbf{z})}$$

$$\mathbf{B} \coloneqq \frac{\lambda\int_{\Omega}\frac{\mathbf{x}-\mathbf{z}}{\|\mathbf{x}-\mathbf{z}\|}\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\,\mathrm{d}A(\mathbf{z})}{\int_{\Omega}\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\,\mathrm{d}A(\mathbf{z})}.$$

The Laplacian can be found by taking the divergence of the gradient. The divergence of the first term $\mathbf{A}$ is

$$\frac{\int_{\Omega}\exp(-\lambda\|\mathbf{x}-\mathbf{z}\|)\left(\Delta g(\mathbf{x},\mathbf{z}) - 2\lambda\left\langle\frac{\mathbf{x}-\mathbf{z}}{\|\mathbf{x}-\mathbf{z}\|}, \nabla_{\mathbf{x}} g(\mathbf{x},\mathbf{z})\right\rangle + \lambda g(\mathbf{x},\mathbf{z})\left(\lambda - \frac{d-1}{\|\mathbf{x}-\mathbf{z}\|}\right)\right)\mathrm{d}A(\mathbf{z})}{\int_{\Omega}\exp(-\lambda\|\mathbf{x}-\mathbf{z}\|)\,\mathrm{d}A(\mathbf{z})} + \langle\mathbf{A},\mathbf{B}\rangle$$

where d is the dimension ($d = 2$ in $\mathbb{R}^2$, $d = 3$ in $\mathbb{R}^3$). The divergence of the second term is

$$\nabla_{\mathbf{x}}\widehat{d^{\lambda}}(\mathbf{x})\cdot\mathbf{B} + \widehat{d^{\lambda}}(\mathbf{x})\left(\frac{\lambda\int_{\Omega}\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\left(\frac{d-1}{\|\mathbf{x}-\mathbf{z}\|} - \lambda\right)\,\mathrm{d}A(\mathbf{z})}{\int_{\Omega}\exp\left(-\lambda\|\mathbf{x}-\mathbf{z}\|\right)\,\mathrm{d}A(\mathbf{z})} + \langle\mathbf{B},\mathbf{B}\rangle\right).$$

Adding up these two divergence expressions gives us $\Delta\widehat{d^{\lambda}}(\mathbf{x})$.

E Analytical signed distance function parameters

As part of evaluation, we use fourteen 2D SDFs from Quilez [2019]: circle, pie, arc, segment, vesica, box, cross, pentagon, hexagon, triangle, quad, ellipse, moon, and trapezoid. We sample instances of each 2D SDF using parameters in the following ranges, chosen such that shapes remain intersection-free:

circle	$r \in [r_{\min}, r_{\max}]$
pie	$r \in [r_{\min}, r_{\max}]$, $t \in [t_{\min}, t_{\max}]$
arc	$r_a \in [r_{\min}, r_{\max}]$, $r_b \in [r_a \cdot z_{\min}, r_a \cdot z_{\max}]$, $t \in [t_{\min}, t_{\max}]$
vesica	$r \in [r_{\min}, r_{\max}]$, $d \in [2r(z_{\min}) - r, 2r(z_{\max}) - r]$
box	$b_x \in [r_{\min}, r_{\max}]$, $b_y \in [r_{\min}, r_{\max}]$
cross	$b_x \in [r_{\min}, r_{\max}]$, $b_y \in [r_{\min}, r_{\max}]$
pentagon	$r \in [r_{\min}, r_{\max}]$
hexagon	$r \in [r_{\min}, r_{\max}]$
ellipse	$a \in [r_{\min}, r_{\max}]$, $b \in [r_{\min}, r_{\max}]$
moon	$r_a \in [r_{\min}, r_{\max}]$, $r_b \in [r_a \cdot z_{\min}, r_a \cdot z_{\max}]$, $d \in [(r_a - r_b)/z_{\max}, (r_a + r_b) \cdot z_{\max}]$
trapezoid	$r_a \in [r_{\min}, r_{\max}]$, $r_b \in [r_{\min}, r_{\max}]$, $h \in [2r_{\min}, 2r_{\max}]$

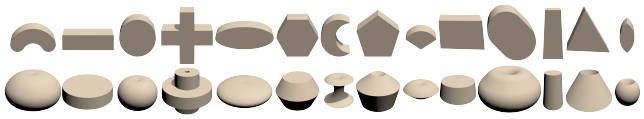


Figure 31. A few of the analytical SDFs used for evaluation in Section 5.2.

For the segment shape, we select its two endpoints as $a = [-h, 0]$ and $b = [h, 0]$, where $h \in [r_{\min}, r_{\max}]$, and choose $r \in [h \cdot z_{\min}, h \cdot z_{\max}]$. For the triangle shape, we start with an equilateral triangle of height $h \in [r_{\min}, r_{\max}]$, and perturb its three vertices using three different randomly sampled vectors of magnitude $h \cdot z_{\max}$. For the quad shape, we start with a rectangle with width $w \in [r_{\min}, r_{\max}]$ and height $h \in [r_{\min}, r_{\max}]$, and perturb its four vertices using four different randomly sampled vectors of magnitude $\min(w \cdot z_{\max}, h \cdot z_{\max})$.

We set $r_{\min} = 0.1$ and $r_{\max} = 0.9$ so shapes stay roughly within $[-1, 1]^2$. We set $z_{\min} = 0.1$ and $z_{\max} = 0.9$, which control the minimum and maximum ratio of smaller radii to larger radii. We set $t_{\min} = \pi/6$ and $t_{\max} = 5\pi/6$, which control the minimum and maximum angles of certain shapes. All parameters are chosen uniformly at random within their ranges.