Points as Tori: Fast Pointwise Signed Distance for Point Clouds

Nicole Feng
Ioannis Gkioulekas
Keenan Crane
Carnegie Mellon University

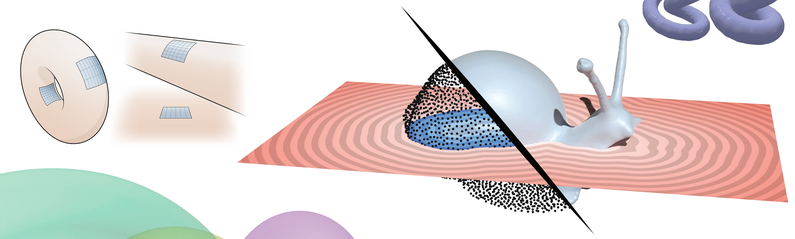
Points as tori (PAT) allows **fast pointwise evaluation** of **signed distance** from **point clouds,** at **arbitrary spatial resolution.**

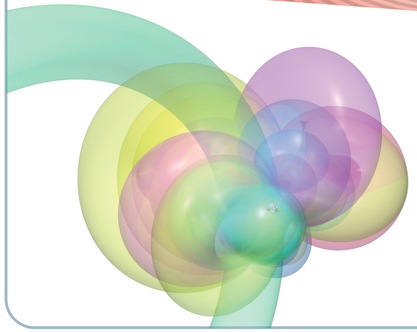
Given a point cloud with normals $P = \{\mathbf{p}_i, \mathbf{n}_i\}_{i=1}^{|P|}$, PAT evaluates signed distance $\phi(\mathbf{x})$ at a given point $\mathbf{x} \in \mathbb{R}^3$ simply using the *kernel density estimator*

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

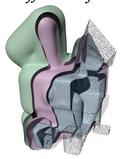
The formula implements closest-point interpolation of functions g_i.

We define g_i as signed distance functions (SDFs) of tori, because torus SDFs have efficient closed-form expressions, and tori can locally approximate surfaces up to second order.



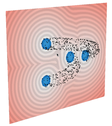


offset surfaces



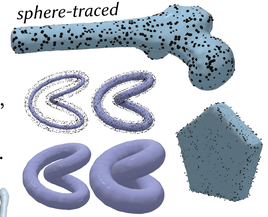
Booleans

shader visualizations

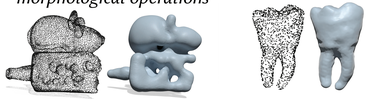


signed distance reconstructions

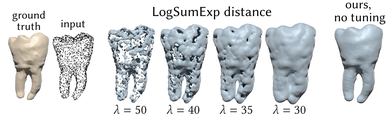
sphere-traced

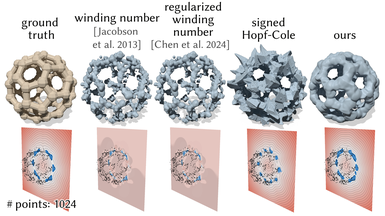


morphological operations

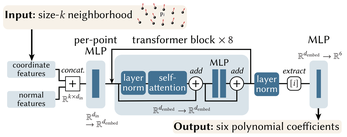


better reconstruction than winding numbers and other point-based methods





We determine these tori by pre-training a neural network. Once this network is trained, one can infer signed distance from **any** point cloud.



The algorithm is
- trivially parallelizable
- output-sensitive (evaluation at specific query points)
- doesn't require global solves or spatial discretization
- differentiable

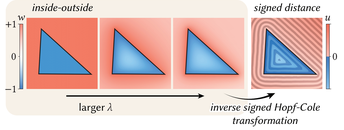
Why this formulation?

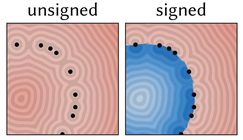
There are two *convolutional distance formulas* for approximating distance to a surface Ω.

Formula 1: $$\widetilde{d}^{\lambda}(\mathbf{x}) = -\frac{1}{\lambda} \log\left(\int_{\Omega} h(\mathbf{x}, \mathbf{z}) \exp\left(-\lambda \|\mathbf{x} - \mathbf{z}\|\right) \, dA(\mathbf{z})\right)$$

Many special cases:
- *Kreisselmeier-Steinhauser function*
  - [Kreisselmeier & Steinhauser 1980]
- *Varadhan's formula*
  - [Varadhan 1967]
- *Schrödinger distance transform*
  - [Gurumoorthy & Rangarajan 2009]
- *LogSumExp distance*
  - [Madan & Levin 2022]
- ... and more

Unifies signed distance with **(regularized) winding numbers** and **Poisson surface reconstruction**!





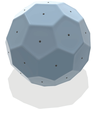
However, Formula 1 cannot generalize signed distance to point clouds!
And any regularization is futile.

Formula 2: $$\widehat{d}^{\lambda}(\mathbf{x}) = \frac{\int_{\Omega} g(\mathbf{x}, \mathbf{z}) \exp\left(-\lambda \|\mathbf{x} - \mathbf{z}\|\right) \, dA(\mathbf{z})}{\int_{\Omega} \exp\left(-\lambda \|\mathbf{x} - \mathbf{z}\|\right) \, dA(\mathbf{z})}$$

Many special cases:
- *kernel regression*
- *partition-of-unity methods*
- *induced exponential function*
- *softmax function*
- ... and more

Formula 2 can control the estimated surface around via the functions g.

Many past works use simple linear functions, leading to ugly/brittle shape completion:

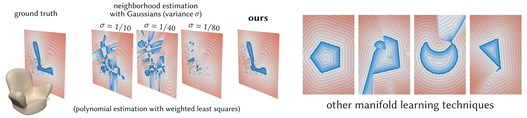


We instead use the SDFs of tori, which admit inexpensive, closed-form signed distance queries (unlike other quadratic surfaces).

Why learning?

Classic methods for (local) surface reconstruction are brittle. Even just estimating local neighborhood size is difficult.

So we simply use *fixed-size* neighborhoods with a *data-driven* approach that can fit to extremely large sets of (easy-to-generate) examples — a far more sophisticated data-fitting model than we could ever craft purely by hand.



Why not end-to-end learning?

Rather than try to directly learn SDFs from entire point clouds, **we intentionally invoke the minimum amount of learning needed**. We rely on classical results for convergence to true signed distance, and using learning only for the one component with no easy analytical solution.

Learning SDFs end-to-end is difficult because the eikonal equation is nonlinear; many existing methods involve costly optimization or per-shape training.

Connection to other kernel methods

We discuss our method's connection to other kernel methods, including past convolutional distance formulas, implicit modeling, and manifold learning techniques. A theme is that *exponential asymptotics are useful for clustering problems, but are often detrimental in regression problems.*